Tesis Doctoral

Sobre grafos arco-circulares propios y helly

Soulignac, Francisco Juan

2010

UNIVERSIDAD DE BUENOS AIRES

Facultad de Ciencias Exactas y Naturales

Departamento de Computación

SOBRE GRAFOS ARCO-CIRCULARES PROPIOS Y HELLY

Tesis presentada para optar al título de
Doctor de la Universidad de Buenos Aires
en el área Ciencias de la Computación

Francisco Juan Soulignac

Director de tesis: Dr. Min Chih Lin

Consejero de estudios: Dr. Min Chih Lin

Buenos Aires, 2010

Sobre grafos arco-circulares propios y Helly

Un modelo arco-circular es un par $\mathcal{M} = (C, \mathcal{A})$ donde C es un círculo y $\mathcal{A}$ es una familia de arcos de C. Si ningún arco se encuentra contenido en otro arco entonces decimos que $\mathcal{M}$ es propio, mientras que si $\mathcal{A}$ satisface la propiedad de Helly entonces decimos que $\mathcal{M}$ es Helly. Un grafo G es arco-circular si es el grafo de intersección de los arcos de un modelo arco-circular $\mathcal{M}$. Si además $\mathcal{M}$ es propio (resp. Helly) entonces decimos que G es un grafo arco-circular propio (resp. Helly). Los grafos arco-circulares y sus subclases son estudiados con especial atención desde fines de la década de 1960, y al día de hoy la literatura al respecto es muy vasta. Esto se debe a la gran cantidad de aplicaciones que poseen en áreas tan diversas como las bases de datos, la genética, la arqueología, la psicología, la economía, etc., y a las propiedades de su estructura combinatoria. El problema de reconocimiento de grafos arco-circulares, y de varias de sus subclases, puede ser resuelto en tiempo lineal. Más aún, un modelo arco-circular puede ser generado en tiempo lineal. En esta tesis estudiamos la clase de grafos arco-circulares desde una perspectiva estructural y algorítmica, concentrándonos principalmente en las subclases de grafos arco-circulares propios y Helly.

Palabras clave: *grafos arco-circulares propios, grafos arco-circulares Helly, grafos de intervalos, potencias de caminos, potencias de ciclos, algoritmos de reconocimiento, algoritmos de transformación, algoritmos de reconocimiento dinámicos, problema de isomorfismo, grafos clique, comportamiento del operador clique iterado.*

On proper and Helly circular-arc graphs

A circular-arc model $\mathcal{M} = (C, \mathcal{A})$ is a circle C together with a collection $\mathcal{A}$ of arcs of C. If no arc is contained in any other, then $\mathcal{M}$ is a proper circular-arc model, and if $\mathcal{A}$ satisfies the Helly Property, then $\mathcal{M}$ is a Helly circular-arc model. A graph G is a circular-arc graph if it is the intersection graph of the arcs of a circular-arc model $\mathcal{M}$. If in addition $\mathcal{M}$ is a proper (resp. Helly) circular-arc model then G is a proper (resp. Helly) circular-arc graph. Circular-arc graphs and their subclasses have been the object of a great deal of attention in the literature since the late 1960's. This is because of their applications in areas as diverse as databases, genetics, archeology, psychology, economics, among others, and because of their nice combinatorial structure. Linear time recognition algorithms have been described both for the general class and for some of its subclasses. Moreover, a circular-arc model can be obtained within the same amount of time. In this thesis we study circular-arc graphs from a structural and algorithmic point of view, with our focus on the proper and Helly subclasses.

Keywords: *proper circular-arc graphs, Helly circular-arc graphs, interval graphs, powers of paths, powers of cycles, recognition algorithms, transformation algorithms, dynamic recognition algorithms, isomorphism problem, clique graphs, K-behavior.*

Para yeye.
Este logro no es mío,
es nuestro.

(*To my mother.*
This achievement is not mine,
is ours.)

Acknowledgements

Throughout the time I spent working on this Thesis, I learned quite a lot about graphs, algorithms, circular-arcs, the Helly property, and many more things. The consequence of all this learning is somehow reflected in the document you are reading now. But there are two other people that are responsible of this work: Oscar Lin (formally known as Min Chih Lin) and Jayme Luiz Szwarcfiter[1]. Without their advise, their ideas, and their teaching, this thesis would have zero chapters! I will never forget the breakfast and lunch talks with Oscar, or the seaside walks with Jayme at Puerto Varas. Neither Google will forget about the tons of chat sessions, mails, and drafts that we exchanged. However, what I have humanly learned from them in these years broadly surpasses the value of this thesis. Both Oscar and Jayme shared with me more than what an advisor is required to, asking nothing in exchange. It is wonderful to experiment that there are advisors who don't think of their students as investments. Thanks a lot!

Two other people had collaborated with us. Dieter Rautenbach worked with us in the results about powers of cycles and paths, and Ross McConnell worked with us in the algorithm for computing the cliques of a Helly circular-arc model. I want to thank both of them for their ideas. I also want to thank the juries, Profs. Martin C. Golumbic, Marisa Gutiérrez, and Yoshiko Wakabayashi, for taking the time to read the thesis; for their cooperation; and for their invaluable reports and comments.

Late, but not least, I want to express my gratitude to the members of the Operations Research, Combinatorial Optimization and Graphs Group, and to my office friends. Their help, enthusiasm, comradeship, and affection (mixed with some beautiful conversations, snacks, coffees, and bondiolas) is one of the most rewarding aspects of my work. You can bet that the good mood that they share with me is a key ingredient of this thesis. Furthermore, is a key ingredient of my life.

Finally, I want to thank the *tus papis son hermanos* group for the nickname (now you can call me dr.oopy instead of droopy) and to all the other friends that accompanied me in these long ten years of studies.

March 29, 2010.

[1] Though there were some bureaucratic issues that prevented its formal inclusion, Jayme is my (informal) co-adviser.

Contents

List of Figures

List of Tables

List of Algorithms

graph G encodes a semiorder $\prec$ such that $v \prec w$ if and only if the interval corresponding to v is completely to the left of the interval corresponding to w. For this reason, proper interval graphs are also known by the name of *indifference* graphs.

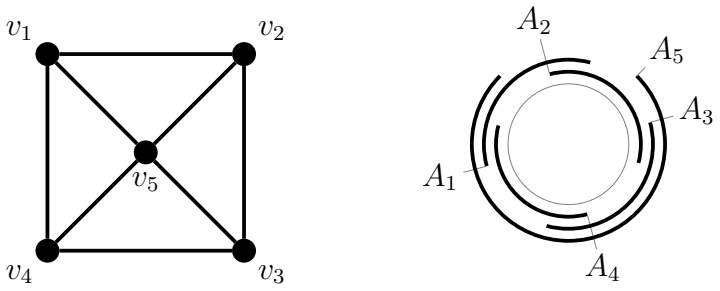
It is rather easy to transform an interval model of a graph G into a proper interval model of G, whenever possible (see *e.g.* [BW99]). So, we can recognize if a graph G is actually a PIG graph with a two step procedure. First, find some interval model of the input graph and, next, transform this model into a PIG model. This algorithm has one major drawback because, as we mentioned before, the classic algorithms that compute the initial interval model are hard. So, many linear-time algorithms to directly recognize if a given graph is a proper interval graph were designed [Cor04, CKN+95, DHH96, HH05, HdFMPdM95]. As for interval graphs, these algorithms are able to produce a proper interval model of the input graph. Furthermore, Corneil et al. [CKN+95] also show how to obtain a unit interval model for the input graph. On the other hand, a minimal forbidden induced subgraph can be obtained, also in linear time, when the input graph is not a PIG graph [HH05]. The family of such minimal forbidden induced subgraphs was first described by Wegner [Weg67].

1.2 Circular-arc graphs

A graph is a *circular-arc graph* if its vertices are in a one-to-one correspondence with a family of arcs of some circle in such a way that two vertices of the graph are adjacent if and only if their corresponding arcs have nonempty intersection (see Figure 1.2). The circle together with the family of arcs is called a *circular-arc model* or *circular-arc representation* of the graph. The relation between circular-arc graphs and interval graphs is clear from the definition. If we bend the real line into a circle, then any family of intervals of the real line is transformed into a family of arcs of the circle. Therefore, every interval graph is a circular-arc graph.

The appearance of circular-arc graphs in the graph theory scene is not so clear as it is for interval graphs. The first reference to the recognition problem of circular-arc graphs seems to be the one in the English translation of a combinatorial geometry book [HD64]. In that book there is a section by Klee, who writes (Page 54):

> "Graphs are also useful in describing certain intersection properties of a family F of sets, where we associate a node with every set in the family and join two nodes by an edge if and only if the corresponding sets intersect. The graph G does not describe all of the intersection properties of F, but only those associated with pairwise intersections. However, the one-dimensional form of Helly's theorem (Proposition 16) shows that G does describe all of F's intersection properties when F is a finite family of segments in the line R^1. The graphs that are obtainable in this way were characterized by



A circular-arc graph G A circular-arc model of G

Figure 1.2: A circular-arc graph and one of its circular-arc models, where the arc A_i corresponds to the vertex v_i for $1 \leq i \leq 5$.

> C. G. Lekkerkerker — J. C. Boland [171], but the corresponding problem is unsolved for other classes of families of sets — arcs in a circle, circular disks in E^2, convex sets in R^n, and so forth."

The strong relationship between interval graphs and circular-arc graphs is present in this paragraph since, to motivate the study of circular-arc graphs (*i.e.*, "graphs that are obtainable" from a family of "arcs in a circle" by modeling "pairwise intersection"), Klee briefly introduces the family of interval graphs. A few years later, Klee devotes a research problem paper to ask what are the circular-arc graphs [Kle69].

Besides the theoretical interest that circular-arc graphs awake in the graph theory community, there are also many applications of circular-arc graphs in disciplines as diverse as genetics, traffic light scheduling, computer networks, and allocation problems, among others [CDG01, Gol04, Rob78, Sta67, Sto68, SE04, Tuc78]. As for interval graphs, a lot of NP-hard problems can be efficiently solved when restricted to circular-arc graphs, *e.g.* the maximum (weight) independent set problem [Spi03], the minimum dominating set problem [HT91], the maximum (weight) clique problem [Asa91] and others. But, as usual, there is a price to pay for the more generality of circular-arc graphs over interval graphs. First, the algorithms for circular-arc graphs are much more complex than those for interval graphs. Second, some problems that are easy for interval graphs, such as the coloring problem, are hard for the class of circular-arc graphs [GJMP80].

Klee's original question about circular-arc graphs has been answered in many ways, by translating many of the interval graph characterizations to circular-arc graphs (see *e.g.* [Gol04]). However, from an algorithmic viewpoint, these characterizations were not strong enough to provide a polynomial-time recognition algorithm. In fact, the tractability of the recognition problem remained open until 1980, when Tucker devised an $O(n^3)$ time algorithm to compute a circular-arc model of a graph [Tuc80]. As usual, n and m denote the numbers of vertices and edges of the graph, respectively. Since then, the time complexity of the recognition problem was improved to $O(nm)$ by Hsu [Hsu95],

to $O(n^2)$ by Eschen and Spinrad [ES93], and finally, in 2003, the optimal $O(n + m)$ bound was obtained by McConnell [McC03]. Although its time complexity is optimal, McConnell's algorithm is really difficult to understand and implement. A simplified version of this algorithm was developed by Kaplan and Nussbaum [KN06], but it is still quite complex. Hence, the problem of finding a simple linear-time recognition algorithm for circular-arc graphs remains open.

All the recognition algorithms described above produce a circular-arc model of the input graph, when such a model exists. However, none of them is able to produce a piece of evidence that indicates why the input graph does not admit a circular-arc model when the algorithm fails. This has to do, in part, with the fact that the family of forbidden induced graphs is yet unknown. Of course, some forbidden graphs are known, such as those obtained by inserting an isolated vertex to a non-interval graph. Bonomo et al. [BDGS09] made some partial contributions to this problem, by restricting their analysis to some classes and showing all the forbidden subgraphs for each class.

As for interval graphs, many subclasses of circular-arc graphs can be obtained by restricting the way in which arcs can intersect in a circular-arc model. By restricting the containment of arcs, we obtain the generalizations of PIG and UIG graphs. A circular-arc graph is a *proper circular-arc* (PCA) graph if it admits a circular-arc model in which no arc is properly contained in any other arc. If in addition all the arcs in the model have the same length, then the graph is a *unit circular-arc* (UCA) graph. *Proper circular-arc models* and *unit circular-arc models* are defined as usual. As before, every UCA graph is a PCA graph by definition. But in this case, not all the PCA graphs admit a UCA model, thus UCA graphs form a proper subclass of PCA graphs. Both the PCA and UCA classes were characterized by Tucker, who showed their lists of minimal induced forbidden subgraphs [Tuc74].

A graph can be tested to be a PCA graph in linear time. The first linear-time recognition algorithm for PCA graphs is due to Deng et al. [DHH96], and it relies on an incremental PIG recognition algorithm. This algorithm exploits an important similarity between PIG and PCA graphs, namely, PIG graphs admit a straight orientation while PCA graphs admit a round orientation. Deng et al. show how to obtain such a round orientation of the input PCA graph, which can then be transformed into a PCA model in $O(n)$ time. Recently, Kaplan and Nussbaum [KN09] devised an algorithm that provides an evidence of failure when the input graph admits no PCA model. Such an evidence is either a forbidden induced subgraph or an odd cycle of an incompatibility graph. Also recently, Nussbaum [Nus08] showed a new recognition algorithm whose input is a circular-arc model $\mathcal{M}$ of a graph G. This algorithm works in $O(n)$ time and it provides either a PCA model of G or a forbidden induced submodel of $\mathcal{M}$. The problem of finding a forbidden induced subgraph of G is still open when the input graph is not known to be a circular-arc graph.

A graph can also be tested to be a UCA graph in linear time. The first algorithm to recognize UCA graphs is due to Tucker [Tuc74]. This algorithm works by shrinking and lengthening the arcs of an input PCA model, and it can be proved that the number of operations is polynomial. However, as it was noted in [DGM+06, Spi03], the integers involved in such shrinks and lengthens are not of polynomial size with respect to the input PCA model, thus the algorithm cannot be claimed to terminate in polynomial time. The first polynomial-time algorithm for the recognition of UCA graphs was devised by Durán et al. [DGM+06] and it runs in $O(n^2)$ time. The major inconvenience of this algorithm is that it provides no evidence for its claims, so there is no way to obtain a UCA model even when the graph is known to be UCA. Lin and Szwarcfiter [LS08] gave the first efficient algorithm to actually build a UCA model from a PCA model, whenever possible. Their algorithm runs in $O(n)$ time and the extremes of the arcs are values bounded by $O(n^2)$, which is asymptotically the best possible. Concurrently, Kaplan and Nussbaum [KN09] improved the algorithm by Durán et al. so as to find, in $O(n)$ time, a forbidden induced subgraph when the input graph is not UCA.

Only two types of intersections are feasible on interval models: overlap and containment. When we study the intersections between intervals, it makes sense to define those interval models in which one of the intersection types is not possible. By doing so, we obtain the class of proper interval graphs. (Restricting overlaps is not a clever thing to do.) Circular-arc models allow two other types of intersections that define two other subclasses of circular-arc graphs. A graph is a *normal circular-arc* (NCA) graph if it admits a circular-arc model in which no two arcs together cover the circle. As usual, such a model is called a *normal circular-arc* model of the graph. In other words, an NCA model is a model in which no pair of intersecting arcs share two maximal overlapping segments. As we already said, this type of intersection is not possible on interval models, thus all interval graphs are NCA graphs.

Normal circular-arc graphs play an important role in the recognition of UCA graphs. Their importance is due to the following results proven by Tucker in [Tuc74]. First, every PCA model admits a normal PCA model, *i.e.*, every PCA graph is also an NCA graph. Second, every normal PCA model of a UCA graph can be transformed into a UCA model without affecting the order of its extremes. All the UCA recognition algorithms that we mentioned before depend on these two properties. Despite their importance in the recognition of UCA graphs, NCA graphs were ignored for many years as an important class by itself. In the last years, NCA graphs gained more status thanks to Hell and Huang [HH04], who proved that interval bigraphs are precisely those bipartite graphs whose complement is an NCA graph. In [LS09], Lin and Szwarcfiter ask for a forbidden induced subgraph characterization and a polynomial-time recognition algorithm for this class.

The last type of restriction on the intersections of the arcs that we shall consider in this document has to do with the Helly property. The Helly Theorem for intervals asserts that

for every family $\mathcal{I}$ of pairwise intersecting intervals there is a point of the real line that belongs to all the intervals in $\mathcal{I}$. When Klee writes in [HD64] that "the one-dimensional form of Helly's theorem" for an interval graph G with a model F "describes all of F's intersection properties", he actually states that by knowing pairwise intersections we can find out the global intersection of a family of intervals. In terms of graphs, every clique K of G can be associated with a point p on the real line such that p belongs to all the intervals of F corresponding to K. The Helly property is present, at least implicitly, in a great number of algorithms that deal with interval graphs. Just to name an example, the PQ-tree data structure maintains all the feasible permutations of the cliques of an interval graph as if these cliques were points of the real line.

The Helly property is completely lost when the intervals of the real line are generalized to the arcs of a circle. It is not hard to generate a circular-arc model with three pairwise intersecting arcs that share no common point. A graph is a *Helly circular-arc* (HCA) graph if it admits a circular-arc model in which every family of pairwise intersecting arcs share a common point. Such a circular-arc model is called a *Helly circular-arc* model. As the class of HCA graphs lies between the classes of interval and circular-arc graphs, there are many properties that can be generalized from interval graphs to HCA graphs, but that cannot be generalized to general circular-arc graphs. These properties usually involve the cliques of the graphs, *i.e.*, the sets of vertices that correspond to families of pairwise intersecting arcs of the graph.

Helly circular-arc graphs can also be recognized in linear time. The first polynomial-time recognition algorithm runs in $O(n^3)$ time and is due to Gavril [Gav74]. Lin and Szwarcfiter [LS06] developed an $O(n+m)$ time algorithm that transforms every circular-arc model of an HCA graph into an HCA model. They also described all those circular-arc graphs that are not HCA, and showed how to find a forbidden induced subgraph in $O(n + m)$ time for these graphs. This algorithm was recently improved by Joeris et al. [JLM+09] so as to run in $O(n)$ time. A complete characterization by forbidden induced subgraphs, *i.e.* the family of graphs that are neither circular-arc nor HCA, is still not known for this class.

1.3 Our contributions

Circular-arc graphs are a natural generalization of interval graphs, where the intersection of segments of a line (*i.e.* intervals) is replaced with the intersection of segments of a circle (*i.e.* arcs). Despite the replacement of a line with a circle can seem like a petty modification, circular-arc graphs have a much more complex structure than interval graphs. One of the main reasons behind this complexity is that arcs can intersect in different ways than intervals do. In an interval model, each family $\mathcal{I}$ of pairwise intersecting intervals is associated with a unique nonempty segment (s, t) of the real line such that

(s,t) is contained in every interval of $\mathcal{I}$, and (s,t) is inclusion-maximal. The situation for circular-arc models is quite different since for some families of pairwise intersecting arcs there could be zero or more than one of such segments. Let $\mathcal{C}$ be the class of circular-arc models such that every family $\mathcal{A}$ of arcs is associated with a unique nonempty segment (s,t) of the circle such that (s,t) is contained in every arc of $\mathcal{A}$, and (s,t) is inclusion-maximal. It is not hard to prove that $\mathcal{C}$ is precisely the class of circular-arc models that are simultaneously normal and Helly. In this thesis, we define the *normal Helly circular arc* (NHCA) graphs as those graphs that admit a model in $\mathcal{C}$. If in addition the model is proper (resp. unit), then the graph is a *proper (resp. unit) Helly circular-arc* (PHCA, UHCA) graph. There are many properties of (proper) interval graphs that do not hold for general circular-arc graphs, but that can be generalized to (PHCA) NHCA graphs. In Chapters 3 and 8 we prove some of these properties that show the importance of these subclasses of circular-arc graphs. Also, throughout this thesis we show some algorithms that work on interval graphs and can be easily generalized so as to work on NHCA graphs. These algorithms are non-trivial (or unknown) for the general class of circular-arc graphs. So, there is a motivation to generalize those algorithms that work only for interval graphs to the more general class of NHCA graphs.

In Chapter 3 we present partial characterizations of NHCA, PHCA, and UHCA graphs by forbidden induced subgraphs. Such characterizations always state that a graph that belongs to a class of circular-arc graphs also belongs to a restricted class of circular-arc graphs if and only if it contains no graph from a family $\mathcal{F}$ as an induced subgraph. For instance, we prove which circular-arc graphs are also NHCA, which proper (resp. Helly) circular-arc graphs are also PHCA, and so on. As a corollary of these characterizations we obtain an important property of NHCA, PHCA, and UHCA graphs: every circular-arc model of a non-interval NHCA (resp. PHCA, UHCA) graph is an NHCA (resp. a PHCA, a UHCA) model of the graph. Consequently, we obtain $O(n+m)$ time algorithms to test if a graph is NHCA, PHCA, or UHCA. In Chapter 5 we improve many of these algorithms so as to run in $O(n)$ time (see below).

Interval and proper circular-arc graphs are also studied from the point of view of oriented graphs. For instance, PCA graphs are precisely those graphs that can be oriented in such a way that the outset and the inset of every vertex is a transitive tournament. In Chapter 3 we study NCA, NHCA, and PHCA graphs from this perspective as well. In particular, we prove that there is a strong relationship between (proper) interval graphs and (PHCA) NHCA graphs in the sense that (PHCA) NHCA graphs are those locally (proper) interval graphs.

In the process of finding an independent set of a circular-arc graph, Golumbic and Hammer [GH88] proved that non-complete powers of cycles are precisely those circular-arc graphs that contain no dominated vertices. In Chapter 4 we review this equivalence and prove, similarly, that non-complete powers of paths are those interval graphs that contain no dominated "interior" vertices. Furthermore, we prove that every UCA (resp.

UIG) graph is an induced subgraph of a powers of a cycle (resp. path). In Chapter 4 we also give yet another constructive proof of Roberts' "proper = unit" theorem, that says that every PIG graph admits a UIG model. The idea for our proof is to generate a power of path supergraph H of a PIG graph G, then compute a UIG model of H, and, finally, remove all the intervals corresponding to vertices in $H \setminus G$. Our proof is harder than previous ones, but it comes with the guaranty that the obtained model has linear size. This proof is also interesting because it shows an example of how to use the power of a path characterization of UIG graphs.

In [Gar07], Gardi asks for a "linear-time and space" algorithm to transform a PIG model into a UIG model, without building a tree. Our "proper = unit" proof yields a linear space and quadratic time algorithm to solve this problem without using any tree. In Chapter 5, we show how to improve this algorithm so that it runs in linear time, solving Gardi's problem. In the same chapter we also improve the algorithms of Chapter 3 that transform, whenever possible, a circular-arc model into a restricted circular-arc model of some class. When such transformations are not possible, we show how to obtain a forbidden induced submodel of the input model.

The first linear-time algorithm for the recognition of PCA graphs is due to Deng et al. [DHH96]. As a part of their algorithm, Deng et al. have to decide whether a connected graph is a PIG graph or not. To solve this problem, they propose a linear-time incremental recognition algorithm for connected PIG graphs. That is, they show how to augment a connected PIG model in $O(d)$ time so as to insert a new interval that represents a new vertex with degree d. Later, Hell et al. [HSS01] generalized this algorithm to allow insertions and removals of both vertices and edges, even when the graph is not connected. In [DHH96], Deng et al. remark that they "believe there is an incremental linear time algorithm to directly compute" a PCA model of an input PCA graph, "along the lines" of their PIG recognition algorithm. In Chapter 6 we confirm this belief by further generalizing the algorithm by Hell et al. so as to insert vertices into a PCA graph. Our algorithm runs in $O(d)$ time, where d is the degree of the inserted vertex. We also show an algorithm that allows both insertions and removals of vertices from a PCA graph, and an algorithm that allows the insertion and removal of edges when the graph is PHCA. The former runs in $O(d + \log n)$ time while the latter runs in $O(\log n)$ time. It is worth to mention that the time complexities of all our algorithms are the same as those for the algorithms by Hell et al.

Huang proved in [Hua92] that connected and co-connected PCA graphs admit a unique PCA model up to full reversal. As a part of our dynamic PCA recognition algorithm, we have to compute all the co-components of a PCA graph. When a PCA model is given, we can find the unique PCA models of all its co-components in $O(n)$ time. In Chapter 7 we develop a simple linear-time algorithm for the isomorphism problem of PCA graphs that exploits the uniqueness of these models. The idea is to compute a canonical PCA model of the input graphs, and then test the equality of these representations. To compute

such a canonical representation of a graph G, we first compute all the co-components of G. After that, we sort the co-components and we compute a canonical machine representation of the unique model of each co-component. Finally, we combine these representations into a canonical representation of G. This chapter is a good example of how a divide and conquer algorithm can be designed for PCA graphs.

The *clique graph* $K(G)$ of a graph G has the cliques of G as its vertices and two vertices of $K(G)$ are adjacent when their corresponding cliques intersect. In other words, $K(G)$ is the intersection graph of the cliques of G. Deogun and Gopalakrishnan proved in [DG99] that there is a strong relationship between clique graphs of interval graphs and the consecutive retrieval property. In [Hed84], Hedman characterized the clique graphs of interval and proper interval graphs. He proved that the clique graph of an interval graph is always a PIG graph, and that every PIG graph is the clique graph of some PIG graph. These characterizations yield linear-time algorithms for the recognition of clique graphs of interval and PIG graphs. Clique graphs of HCA graphs were studied by Durán and Lin [DL01]. They proved that the clique graph of an HCA graph is always a PCA graph. Also, they showed some characterizations of clique graphs of HCA graphs, but these characterizations are not strong enough to directly yield a polynomial-time algorithm for the recognition problem. In Chapter 8 we provide such a characterization, and we also show a series of characterizations for clique graphs of NHCA and PHCA graphs that resemble the results by Hedman. Specifically, we prove that: 1) the clique graph of an HCA is either a PHCA graph or is obtained by inserting at least two universal vertices to a co-bipartite PHCA graph, 2) the clique graph of an NHCA graph is a PHCA graph, and 3) every PHCA graph is the clique graph of some PHCA graph. As a consequence, we obtain linear-time algorithms for the associated recognition problems.

In Chapter 8 we also show an $O(n)$ time algorithm to find all the cliques of an HCA graph. The first algorithm to solve this problem is due to Durán et al. [DLMS06], and it runs in $O(n^2)$ time.

Finally, we also consider the K-*behavior* of circular-arc graphs. Define the *iterated clique graph* as $K^0(G) = G$ and $K^{i+1}(G) = K(K^i(G))$. A graph G is K-*null* if $K^i(G)$ has only one vertex, for some $i \geq 0$. Say that G is K-*periodic* if $K^i(G) = G$, for some $i > 0$. A graph is K-*convergent* when it is K-null or $K^i(G)$ is K-periodic for some $i \geq 0$. If G is not K-convergent, then $|V(K^i(G))|$ is unbounded when $i \longrightarrow \infty$; in this case G is K-*divergent*. To determine the K-behavior of a graph G means to decide if G is K-null, K-convergent, or K-divergent. No algorithm is known to decide the K-behavior of a general graph. In Chapter 8 we combine some known results to show when does a circular-arc graph K-converge. Moreover, we show a linear-time algorithm to compute the unique graph to which a circular-arc graph K-converges, when it does. In the process of proving the correctness of the algorithm, we show that non-interval NHCA graphs are precisely those graphs that are K-convergent but not K-null.

In the last chapter we discuss some research problems and present further remarks.

2 Preliminaries

2.1 Basic graph notions

A *graph* is an ordered pair $G = (V(G), E(G))$, where $V(G)$ is a nonempty finite set and $E(G)$ is a set of unordered pairs vw with $v, w \in V(G)$ and $v \neq w$. The set $V(G)$ is the *vertex set* of G and its elements are the *vertices* of G, while the set $E(G)$ is the *edge set* of G and its elements are the *edges* of G. The number of vertices of G is called the *order* of G. As it is usual in the literature, we denote $n = |V(G)|$ and $m = |E(G)|$, unless otherwise stated. The unique graph of order 1 is called the *trivial* graph. Observe that in our graph definition, $vv \notin E(G)$ for $v \in V(G)$, $vw \in E(G)$ if and only if $wv \in E(G)$, and $|\{vw \in E(G) \mid v, w \in V(G)\}| \leq 1$. In the literature, these conditions on graphs are referred to as *loopless*, *undirected*, and *simple*, respectively.

A vertex v is *adjacent* to a vertex w when vw is an edge of the graph. We also say that v is a *neighbor* of w when v is adjacent to w. The *neighborhood* of v is the set $N_G(v)$ of all the neighbors of v, and the *complement neighborhood* of v is the set $\overline{N_G}(v)$ of all the non-neighbors of v. The cardinality of $N_G(v)$ is the *degree* of v and is denoted by $d_G(v)$. The minimum and maximum values among the degrees of all the vertices are respectively denoted by $\delta(G)$ and $\Delta(G)$. If $\delta(G) = \Delta(G)$ then G is a *regular* graph. The *closed neighborhood* of v is the set $N_G[v] = N_G(v) \cup \{v\}$. If $N_G[v] = V(G)$ then v is a *universal vertex* while if $N_G(v) = \emptyset$ then v is an *isolated vertex*. We will omit the subscripts in N and d when there is no ambiguity about G.

Two vertices v and w are *true twins*, or simply *twins*, when $N[v] = N[w]$. We refer to v and w as *false twins* when $N(v) = N(w)$. Vertex v is a *dominator* of w if $N[w] \subseteq N[v]$. In this case we also say that w is *dominated by* v. If in addition $N[w] \subset N[v]$, then v is a *proper* dominator of w and w is *properly* dominated by v. A *dominator sequence* is a maximal sequence of vertices $v_1, \ldots, v_k$ such that v_i is a proper dominator of v_{i+1}, for $1 \leq i < k$.

A graph H is a *subgraph* of the graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. If also $E(H) = \{vw \in E(G) \mid v, w \in V(H)\}$, then H is an *induced* subgraph of G. For each $V \subseteq V(G)$, the subgraph of G *induced by* V is the unique induced subgraph of G whose vertex set is V. We denote by $G[V]$ the subgraph of G induced by V. A subgraph of G whose vertex set is $V(G)$ is called a *spanning subgraph*. For each $V \subset V(G)$, we denote

by $G \setminus V$ the subgraph of G induced by $V(G) \setminus V$. Similarly, for each $E \subseteq E(G)$, we denote by $G \setminus E$ the spanning subgraph of G with edge set $E(G) \setminus E$.

Two graphs G and H are *isomorphic* if there is a one-to-one mapping f between $V(G)$ and $V(H)$ such that $vw \in E(G)$ if and only if $f(v)f(w) \in E(H)$. The mapping f is referred to as an *isomorphism* between G and H. We say that G and H are *disjoint graphs* when $V(G) \cap V(H) = \emptyset$. Graph G is a *copy* of H when G and H are disjoint and isomorphic. For each graph H, the graph G is *H-free* if no induced subgraph of G is isomorphic to H.

The *complement* of a graph G, denoted by $\overline{G}$, is the graph that has the same vertices as G and such that two vertices are adjacent in $\overline{G}$ if and only if they are not adjacent in G. The *disjoint union* of two graphs G and H is the graph $G \cup H$ whose vertex set is $V(G') \cup V(H)$ and whose edge set is $E(G') \cup E(H)$, where G' is a copy of G disjoint from H. The *disjoint join* of G and H is the graph $G + H = \overline{\overline{G} \cup \overline{H}}$. Note that if G and H are disjoint graphs then $G + H$ is the graph obtained from $G \cup H$ by inserting all the edges vw, for $v \in V(G)$ and $w \in V(H)$. For each $k \in \mathbb{N}_{>0}$, we will denote by kG the graph $\bigcup_{i=1}^{k} G$.

A *walk* in a graph G is a sequence $v_1, \ldots, v_k$ of vertices such that v_i is adjacent to v_{i+1}, for every $1 \leq i < k$. Such a walk is said to be a walk *between* v_1 and v_k (or *joining* v_1 with v_k). The vertices v_i and v_{i+1} are said to be *consecutive* in the walk and $v_i v_{i+1}$ is said to be an *edge of the walk*. The *length* of the walk is the number $k - 1$ of edges of the walk. A *closed walk* is a walk that joins a vertex with itself. A *path* is a walk formed by pairwise distinct vertices. A *cycle* is a closed walk $v_1, \ldots, v_k, v_1$ where $k \geq 2$ and $v_1, \ldots, v_k$ is a path. For simplicity, we will say that $v_1, \ldots, v_k$ is a *cycle* for $v_1 \neq v_k$ when $v_1, \ldots, v_k, v_1$ is a cycle.

A graph is *connected* if it contains a path between any two of its vertices. A *disconnected* graph is a graph that is not connected while a *co-connected* graph is a graph whose complement is connected. A *connected component*, or simply a *component*, is a maximal connected subgraph. Similarly, a *co-connected component*, or simply a *co-component*, is a maximal co-connected subgraph.

The *distance* of two vertices v and w in a graph G, denoted by $d_G(v, w)$, is the minimum among the lengths of all the paths between v and w. The distance of v and w is infinity when there is no path joining v with w. As before, we omit the subscript when there is no ambiguity about G. For each $k \in \mathbb{N}_0$, the *k-th power* of G, denoted by G^k, is the graph that has the same vertices as G and such that two vertices are adjacent in G^k if and only if their distance in G is at most k.

A *chord* of a path or cycle is an edge that joins two non-consecutive vertices of the path or cycle. Those paths and cycles that have no chords are called *chordless*. The chordless

path of order n is denoted by P_n and the chordless cycle of order n is denoted by C_n. A *hole* is a chordless cycle of order at least 4.

A *complete set* in a graph G is a set of pairwise adjacent vertices and a *clique* is a maximal complete set. (Note that this definition is not standard; many authors use the term clique to refer to complete sets.) We also use the terms *complete* and *clique* to refer to the subgraphs induced by a complete set and a clique, respectively. An *independent set* is a set of pairwise non-adjacent vertices. The complete graph of order n is denoted by K_n and K_3 is referred to as a *triangle*.

A graph G is *bipartite* when there is a partition V_1, V_2 of $V(G)$ such that both V_1 and V_2 are independent sets. Contrary to the usual definition of a partition, we allow one of the sets V_1 and V_2 to be empty. So, the trivial graph is bipartite for us. The partition V_1, V_2 is called a *bipartition* of $V(G)$, and we denote it by $\langle V_1, V_2 \rangle$. If each vertex in V_1 is adjacent to all the vertices in V_2, then G is a *complete bipartite* graph. The complete bipartite graph with bipartition $\langle V_1, V_2 \rangle$ is denoted by $K_{|V_1|,|V_2|}$. We call G *co-bipartite* when $\overline{G}$ is bipartite, and we refer to each bipartition of $\overline{G}$ as a *co-bipartition* of G.

Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ be a family of sets. The *intersection graph* of $\mathcal{S}$ is the graph G with vertices $v_1, \ldots, v_n$ such that v_i is adjacent to v_j if and only if S_i and S_j have nonempty intersection, for $1 \leq i < j \leq n$. The family $\mathcal{S}$ is called a *model* or *representation* of G. The *global intersection* of $\mathcal{S}$ is the set $S_1 \cap \ldots \cap S_n$. We say that $\mathcal{S}$ is *intersecting* if its intersection graph is a complete graph. Family $\mathcal{S}$ *satisfies the Helly property*, or simply $\mathcal{S}$ is *Helly*, when every intersecting subfamily of $\mathcal{S}$ has nonempty global intersection. For instance, $\{\{1,2\},\{2,3\},\{1,3\}\}$ is not Helly.

A *class* of graphs is a family $\mathcal{C}$ whose members are graphs. The class $\mathcal{C}$ is *hereditary* if, for every $G \in \mathcal{C}$, all the induced subgraphs of G also belong to $\mathcal{C}$.

2.2 Digraphs

A *digraph* is an ordered pair $D = (V(D), E(D))$ where $V(D)$ is a nonempty finite set and $E(D)$ is a set of ordered pairs (v, w) with $v, w \in V(G)$ and $v \neq w$. The set $V(D)$ is the *vertex set* of D and its elements are the *vertices* of D, while the set $E(D)$ is the *edge set* of D and its elements are the *directed edges* of D. As for graphs, we denote $n = |V(D)|$ and $m = |E(D)|$ unless otherwise stated. In our definition, digraphs are *loopless* and *simple*, *i.e.*, $(v, v) \notin E(D)$ for $v \in V(D)$ and $|\{(v, w) \in E(D) \mid v, w \in V(D)\}| \leq 1$.

We write $v \longrightarrow w$ to indicate that (v, w) is an edge of a digraph D. When (v, w) is not an edge of D, we write $v \not\longrightarrow w$. A vertex v is *adjacent* to a vertex w in D if either $v \longrightarrow w$ or $w \longrightarrow v$. The *underlying graph* of D is the graph $G(D)$ with vertex set $V(D)$ such that v is adjacent to w in G if and only if v is adjacent to w in D. When $v \longrightarrow w$ we say that v is an *in-neighbor* of w and that w is an *out-neighbor* of v. The *inset* of v

is the set $N_D^-(v)$ of all the in-neighbors of v, and the *outset* of v is the set $N_D^+(v)$ of all the out-neighbors of v. The *closed inset* is $N_D^-[v] = N_D^-(v) \cup \{v\}$, and the *closed outset* is $N_D^+[v] = N_D^+(v) \cup \{v\}$. The cardinality of $N_D^-(v)$, denoted by $d_D^-(v)$, is the *indegree* of v and the cardinality of $N_D^+(v)$, denoted by $d_D^+(v)$, is the *outdegree* of v. As for graphs, we omit the subscripts in N and d when there is no ambiguity about D.

A digraph D is an *oriented graph* when either $v \not\to w$ or $w \not\to v$, for every $v, w \in V(D)$. In other words, D is an oriented graph if it can be obtained from a graph G by choosing an orientation for each edge vw of G. In such case, we call D an *orientation* of G, *i.e.*, D is an *orientation* of G if D is an oriented graph whose underlying graph is isomorphic to G.

2.3 Circular-arc models

A *circular-arc model* $\mathcal{M}$ is an ordered pair $(C(\mathcal{M}), \mathcal{A}(\mathcal{M}))$ where $C(\mathcal{M})$ is a circle and $\mathcal{A}(\mathcal{M})$ is a finite family of open arcs of C. The arcs in $\mathcal{A}(\mathcal{M})$ are said to be *arcs* of $\mathcal{M}$, and $C(\mathcal{M})$ is said to be the *circle* of $\mathcal{M}$. Unless explicitly stated, we always choose the clockwise direction for traversing $C(\mathcal{M})$. For $s, t \in C(\mathcal{M})$, write (s, t) to mean the open arc of $C(\mathcal{M})$ defined by traversing the circle from s to t. Call s, t the *extremes* of (s, t), while s is the *beginning point* and t is the *ending point* of the arc. For each $A \in \mathcal{A}(\mathcal{M})$, represent by $s(A)$ the beginning point of A and by $t(A)$ the ending point of A. The *extremes* of $\mathcal{M}$ are those of all the arcs $A \in \mathcal{A}(\mathcal{M})$.

A circular-arc model is *well-formed* when no two extremes of distinct arcs of $\mathcal{M}$ coincide and no single arc entirely covers the circle of $\mathcal{M}$. Throughout this thesis we will assume that every circular-arc model is well-formed (see Proposition 2.4.3). The *complement* of $A \in \mathcal{A}(\mathcal{M})$ is the arc $\overline{A} = (t(A), s(A))$. The *complement* of $\mathcal{M}$ is the circular-arc model $\overline{\mathcal{M}} = (C(\mathcal{M}), \{\overline{A} \mid A \in \mathcal{A}(\mathcal{M})\})$. Note that $\overline{\mathcal{M}}$ is well-formed when $\mathcal{M}$ is well-formed.

Two extremes e_1 and e_2 of a circular-arc model $\mathcal{M}$ are *consecutive* if there is no extreme of $\mathcal{M}$ in (e_1, e_2). In this case, the arc (e_1, e_2) is called a *segment* of $C(\mathcal{M})$, and e_1 and e_2 are respectively called the *left* and *right* extremes of the segment. We also say that e_1 *immediately precedes* (or *is immediately to the left* of) e_2 and that e_2 *immediately succeeds* (or is *immediately to the right* of) e_2. More generally, a sequence $e_1, \ldots, e_k$ of extremes is *consecutive* if e_{i+1} immediately succeeds e_i, for every $1 \leq i \leq k$. Say that $\epsilon > 0$ is *small enough* if $\epsilon < \frac{1}{4}\ell$, where ℓ is the minimum among the lengths of all the segments of $\mathcal{M}$. *Duplicating the arc* A means inserting the arc $(s(A) + \epsilon, t(A) + \epsilon)$ into $\mathcal{M}$, for some small enough ϵ.

We classify the segments (e_1, e_2) of $\mathcal{M}$ into four types according to the nature of e_1 and e_2. If both e_1 and e_2 are beginning points (respectively ending points) of $\mathcal{M}$ then the segment is of type *s-s* (respectively *t-t*), while if e_1 is a beginning point (respectively

an ending point) of $\mathcal{M}$ and e_2 is an ending point (respectively a beginning point) of $\mathcal{M}$ then the segment is of type s-t (respectively t-s). An *s-sequence* is a maximal sequence of consecutive beginning points. Similarly, a *t-sequence* is a maximal sequence of consecutive ending points. In general, an *extreme sequence* means either an s-sequence or a t-sequence.

There are five elementary classes of circular-arc models that are of special interest (see [LS09]). Let $\mathcal{M}$ be a circular-arc model. If $\mathcal{A}(\mathcal{M})$ is a Helly family, then $\mathcal{M}$ is a *Helly* circular-arc (HCA) model. That is, $\mathcal{M}$ is an HCA model if every family of pairwise intersecting arcs share a common point. If $\mathcal{M}$ has no pair of arcs that together cover the circle, then $\mathcal{M}$ is a *normal* circular-arc (NCA) model. When there is no arc contained in any other arc, we say that $\mathcal{M}$ is a *proper* circular-arc (PCA) model. If in addition all the arcs have the same length, then $\mathcal{M}$ is a *unit* circular-arc (UCA) model. Finally, $\mathcal{M}$ is an *interval* circular-arc (ICA) model when some point of $C(\mathcal{M})$ is covered by no arcs.

An *interval model* is a finite family $\mathcal{I}$ of finite open intervals on the real line. Every interval circular-arc model $\mathcal{M}$ is in correspondence with an interval model $\mathcal{I}$ as follows. Take the point p of $C(\mathcal{M})$ that is not crossed by any arc. By removing the arc $[p-\epsilon, p+\epsilon]$ from $C(\mathcal{M})$ (for some appropriate value of ϵ), we obtain a segment of the circle that contains all the arcs of $\mathcal{M}$. This segment can be put into the real line so as to obtain an interval model $\mathcal{I}$. Similarly, every interval model $\mathcal{I}$ can be transformed into $\mathcal{M}$ as follows. Take two points ℓ and r of the real line such that all the intervals of $\mathcal{I}$ lie in the segment (ℓ, r). Extract the segment (ℓ, r) from the real line and bend this segment into a circle in such a way that ℓ gets identified with r. Consequently, ICA models can be put in a one-to-one correspondence with interval models. Furthermore, two arcs of an ICA model intersect if and only if the corresponding intervals intersect in the interval model. For this reason we will use interval models instead of ICA models, and every definition on circular-arc models translates to interval models with the above correspondence.

The five elementary classes of circular-arc models can be combined so as to generate a total of 32 classes of circular-arc models, as follows. Let $X \subseteq \{\text{N, P, U, H, I}\}$. Say that $\mathcal{M}$ is an XCA model if $\mathcal{M}$ is an xCA model, for every $x \in X$. For instance, $\mathcal{M}$ is an {N,H}CA model if $\mathcal{M}$ is both an NCA and an HCA model. Clearly, if $\mathcal{M}$ is an XCA model then $\mathcal{M}$ is also an YCA for every $Y \subseteq X$. Not all the 32 classes of circular-arc models are different, because some of the properties are implied by others. For example, every UCA model is proper, thus every UCA model is a {U,P}CA model as well. Similarly, every interval model is Helly and normal. This leaves us with 15 different classes of circular-arc models that are listed in Table 2.1. To avoid the ugly set notation to name a class of circular-arc models, we choose a better acronym for each class of circular-arc models.

Name of the model	Acronym	I	U	P	H	N
Unit interval	UIG	1	1	1	1	1
Proper interval	PIG	1	0	1	1	1
Interval	IG	1	0	0	1	1
Normal unit Helly circular-arc	NUHCA	0	1	1	1	1
Unit Helly circular-arc	UHCA	0	1	1	1	0
Normal unit circular-arc	NUCA	0	1	1	0	1
Unit circular-arc	UCA	0	1	1	0	0
Normal proper Helly circular-arc	NPHCA	0	0	1	1	1
Proper Helly circular-arc	PHCA	0	0	1	1	0
Normal proper circular-arc	PNCA	0	0	1	0	1
Proper circular-arc	PCA	0	0	1	0	0
Normal Helly circular-arc	NHCA	0	0	0	1	1
Helly circular-arc	HCA	0	0	0	1	0
Normal circular-arc	NCA	0	0	0	0	1
Circular-arc	CA	0	0	0	0	0

Table 2.1: Subclasses of circular-arc models. The five columns on the right show the properties that each class of model satisfies.

2.4 Circular-arc graphs

A graph G is a *circular-arc graph* if there is a one-to-one correspondence between the vertices of G and a family $\mathcal{A}$ of arcs of a circle C such that two vertices are adjacent if and only if their corresponding arcs have nonempty intersection. The circular-arc model $(C, \mathcal{A})$ is called a *model* or *representation* of G, and G is said to *admit* the model $(C, \mathcal{A})$. In other words, G is a circular-arc graph if it is isomorphic to the intersection graph of $\mathcal{A}$. For simplicity, we also say that G is an intersection graph of $(C, \mathcal{A})$. Say that two circular-arc models are *equivalent* when their intersection graphs are isomorphic.

By restricting the attention to a subclass of circular-arc models, we obtain a special class of circular-arc graphs, formed by those graphs that admit a model of the subclass. For $X \subseteq \{\text{N, P, U, H, I}\}$, say that G is an XCA graph when G admits an XCA model. As before, these 32 classes of circular-arc graphs are not all different. Not even the fifteen classes defined by the special models in Table 2.1 are all different, because graphs may admit many circular-arc models.

Theorem 2.4.1 ([Rob69], see Theorem 3.1.1)**.** *Every PIG model is equivalent to a UIG model.*

Theorem 2.4.2 ([Tuc74], see Theorem 2.4.8)**.** *Every PCA model is equivalent to an NPCA model.*

Name of the class	Acronyms	Admitted models
Proper interval (unit interval)	PIG (UIG)	UIG and PIG
Interval	IG	interval
Unit Helly circular-arc	UHCA	UHCA and NUHCA
Unit circular-arc	UCA	UCA and NUCA
Proper Helly circular-arc	PHCA	PHCA and NPHCA
Proper circular-arc	PCA	PCA and NPCA
Normal Helly circular-arc	NHCA	NHCA
Helly circular-arc	HCA	HCA
Normal circular-arc	NCA	NCA
Circular-arc	CA	CA

Table 2.2: Subclasses of circular-arc graphs, according to the class of models that they admit.

In total, ten different subclasses of circular-arc graphs are obtained by combining the normal, proper, unit, Helly, and interval properties of a circular-arc model. (In Chapter 3 we prove that these classes are all pairwise different.) These classes are enumerated in Table 2.2. As before, we avoid the ugly set notation by defining appropriate acronyms for each subclass. For historic reasons, proper interval graphs are also called unit interval graphs and the acronym UIG is also used for PIG graphs.

In view of Table 2.2, we say that a PIG model is *strong* if it is a UIG model, and that a PCA model is *strong* if it is an NPCA model. Non-strong models are referred to as *weak*. The following proposition shows that there is no loss of generality in considering only well-formed circular-arc models.

Proposition 2.4.3 (see *e.g.* [Gol04])**.** *Every XCA graph admits a well-formed XCA model, for* $X \subseteq \{N,\ P,\ U,\ H,\ I\}$.

Let ϵ be a small enough value for a circular-arc model $\mathcal{M}$. If we replace an arc $A \in \mathcal{A}(\mathcal{M})$ with the arc $(s(A) + \epsilon, t(A) + \epsilon)$, we obtain a model $\mathcal{M}'$ equivalent to $\mathcal{M}$. This implies that any circular-arc graph admits an infinite number of equivalent models. However, $\mathcal{M}$ and $\mathcal{M}'$ are essentially the same model, since all we are interested in is in how do the arcs of $\mathcal{M}$ intersect. Say that two circular-arc models $\mathcal{M}$ and $\mathcal{M}'$ have *equal extremes*, or simply that $\mathcal{M}$ and $\mathcal{M}'$ are *equal*, if there is a one-to-one mapping f from the extremes of $\mathcal{M}$ to the extremes of $\mathcal{M}'$ such that:

(i) (e, e') is a segment of $\mathcal{M}$ if and only if $(f(e), f(e'))$ is a segment of $\mathcal{M}'$ and

(ii) (s, t) is an arc of $\mathcal{M}$ if and only if $(f(s), f(t))$ is an arc of $\mathcal{M}'$.

In other words, $\mathcal{M}$ and $\mathcal{M}'$ are equal if their extremes appear in the same order. (An alternative definition of equality using strings is given in Chapter 7.) With this definition,

every circular-arc graph admits a finite number of non-equal models. When we informally say that a graph G *admits* k *models*, we mean that G admits k non-equal models.

We will use the same terminology used for graphs and vertices when talking about circular-arc models and arcs. For example, we say that an arc is *universal* to mean that its corresponding vertex is universal in the intersection graph. Similarly, we call a model *connected* when its intersection graph is connected.

For each circular-arc model $\mathcal{M}$, we will represent by $\mathcal{M}^{-1}$ the *reverse* model of $\mathcal{M}$. That is, $\mathcal{M}^{-1}$ is the model $(C, \mathcal{A})$ where C is obtained by reflecting $C(\mathcal{M})$ with respect to some chord and $\mathcal{A} = \{(t, s) \mid (s, t) \in \mathcal{A}(\mathcal{M})\}$. Note that $\mathcal{M}$ and $\mathcal{M}^{-1}$ are always equivalent. For $k \in \mathbb{N}_0$, define $U_k(\mathcal{M})$ to be the model obtained from $\mathcal{M}$ by removing all but k of its universal arcs, if existing. Clearly, $\mathcal{M} = U_k(\mathcal{M})$ if and only if $\mathcal{M}$ has at most k universal arcs.

In a circular-arc model, all the arcs that cover some point p of the circle form a complete set. If this complete set is also a clique $\mathcal{Q}$, then we call p a *clique point* and we say that $\mathcal{Q}$ is *represented* by p. Recall that, by definition, a circular-arc model is Helly precisely when every family of pairwise intersecting arcs share a common point. In other words, an HCA model is a circular-arc model in which every clique is represented by a clique point.

A family of arcs $\mathcal{A}$ of a circular-arc model $\mathcal{M}$ is *twin-consecutive* when both the set of beginning points and the set of ending points of $\mathcal{A}$ correspond to consecutive sequences. Clearly, every family of twin-consecutive arcs is formed by twin arcs, but the converse is not necessarily true. We say that $\mathcal{M}$ is *twin-consecutive* if every maximal family of twin arcs is twin-consecutive. The following lemma shows that every XCA graph admits a twin-consecutive XCA model.

Lemma 2.4.4. *Let $\mathcal{M}$ be an XCA model for $X \subset \{I, U, P, H, N\}$, and A be an arc of $\mathcal{M}$. Then the model $\mathcal{M}'$ that is obtained by duplicating the arc A is an XCA model.*

Corollary 2.4.5. *Let v and w be two twin vertices of a graph G. Then G is an XCA graph if and only if $G \setminus \{v\}$ is an XCA graph, for every $X \subset \{I, U, P, H, N\}$. Furthermore, $\mathcal{M}$ is an XCA model of $G \setminus \{v\}$ where the arc A corresponds to w if and only if the model obtained by duplicating A in $\mathcal{M}$ is an XCA model of G.*

Tucker [Tuc74] defined the class of $CI(n, k)$ graphs so as to characterize those PCA graphs that are not UCA. For relative prime values n and k such that $n > 2k$, define $CI(n, k)$ as circular-arc model that is built as follows (see also Figure 2.1). Let C be a circle of length $4n$. Draw n arcs $A_0, \ldots, A_{n-1}$ of length $4k+1$ such that each A_i begins at $4ki$ and ends at $4k(i+1)+1$. Afterwards, draw n arcs $B_0, \ldots, B_{n-1}$ of length $4k-1$ such that each B_i begins at $4ki + 2k + 1$ and ends at $4k(i + 1) + 2k$. The intersection graph of the model $CI(n, k)$ is called the $CI(n, k)$ graph. Note that every $CI(n, k)$ model is an NPCA model by definition.

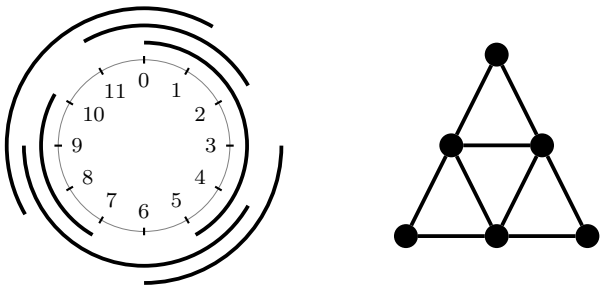


Figure 2.1: The circular-arc model $CI(3,1)$ and its intersection graph.

Theorem 2.4.6 ([Tuc74])**.** *Let G be a PCA graph. Then G is a UCA graph if and only if it contains no $CI(n,k)$ as an induced subgraph, with n and k relative primes and $n > 2k$.*

As we already mentioned, Tucker [Tuc74] (see also [Gol04]) proved that every PCA graph admits an NPCA model. We now give a rather elementary proof of this fact which, we believe, is much simpler than the one in [Gol04]. For this, we need to use the following simple lemma.

Lemma 2.4.7. *Let $\mathcal{M}$ be a PCA model of a graph G. If A_1, A_2 are two arcs of $\mathcal{M}$ that together cover the circle, then both A_1 and A_2 are universal.*

Proof. The order in which the extremes of A_1 and A_2 appear in a traversal of $C(\mathcal{M})$ is $s(A_1), t(A_2), s(A_2), t(A_1)$ because A_1 and A_2 cover the circle. If some arc A has its ending point $t(A)$ outside A_1, then $t(A) \in A_2$. If in addition $s(A) \in A_2 \setminus A_1$, then either $A \subset A_2$ or $A \supset A_1$, which contradicts the fact that $\mathcal{M}$ is a PCA model. Consequently, every arc has either its ending or its beginning point inside A_1, *i.e.*, A_1 is universal. The proof for A_2 is analogous. □

Tucker's theorem is now a simple corollary.

Theorem 2.4.8 ([Gol04, Tuc74])**.** *Every PCA model is equivalent to an NPCA model.*

Proof. Let $\mathcal{M}$ be a PCA model with k universal arcs. By Lemma 2.4.7, $U_1(\mathcal{M})$ is an NPCA model because it has only one universal arc. If $k > 1$ then duplicate $k-1$ times the universal arc of $U_1(\mathcal{M})$ so as to include all the universal arcs that were possibly removed from $\mathcal{M}$ in the process of building $U_1(\mathcal{M})$. The resulting model is NPCA by Lemma 2.4.4. □

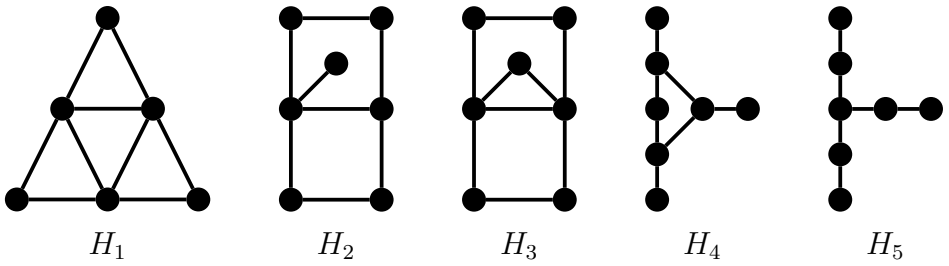


Figure 2.2: Complements of the forbidden induced subgraphs for PCA graphs

The strength of Lemma 2.4.7 lies not only on its simplicity and the fact that it implies Theorem 2.4.8. It also helps us to transform a PCA model of a graph G into an NPCA model of G, in $O(n)$ time (see Chapter 3). And, it can be used to find all the universal arcs of $\mathcal{M}$ as follows.

Lemma 2.4.9. *An arc A of a PCA model $\mathcal{M}$ is universal if and only if A contains at least $n-1$ extremes of $\mathcal{M}$, where $n = |\mathcal{A}(\mathcal{M})|$.*

Proof. Let k be the number of extremes that appear inside A. If $k < n-1$ then there is at least one arc A' that has both of its extremes outside A. Since A is not contained in any other arc, then it follows that $A \cap A' = \emptyset$, *i.e.*, A is not universal. If $k \geq n-1$ then either all the arcs have at least one extreme inside A or there is one arc whose both extremes lie inside A. In the first case A is universal by definition, while in the second case A is universal by Lemma 2.4.7. □

For a graph G, denote by G^* the graph that is obtained from G by inserting an isolated vertex. PCA graphs were also characterized by forbidden subgraphs by Tucker [Tuc74], as follows.

Theorem 2.4.10 ([Tuc74])**.** *A graph G is a PCA graph if and only if it does not contain as induced subgraphs any of the following graphs: C_n^* for $n \geq 4$, $\overline{C_{2n}}$ for $n \geq 3$, $\overline{C_{2n+1}^*}$ for $n \geq 1$, and the graphs $\overline{H_1}$, $\overline{H_2}$, $\overline{H_3}$, $\overline{H_4}$, $\overline{H_5}$ and H_1^* (see Figure 2.2).*

2.5 Orderings

For a linear ordering $X = x_1, \ldots, x_n$, we will say that x_i is *before* (or *to the left*) of x_j, for every $1 \leq i < j \leq n$. The elements x_1 and x_n are respectively called the *first* (or *leftmost*) and *last* (or *rightmost*) of X. If $Y = y_1, \ldots, y_m$ is a linear ordering, we denote by X, Y the linear ordering $x_1, \ldots, x_n, y_1, \ldots, y_m$. We also consider each single element

y a linear ordering, thus X, y is the linear ordering $x_1, \ldots, x_n, y$, and y, X is the linear ordering $y, x_1, \ldots, x_n$.

In this thesis we deal with a lot of collections that are of a circular (cyclic) nature, such as circles (viewed as a collection of points), circular lists, etc. Generally, the objects in a collection are labeled with some kind of index that identifies the position of the object inside the collection. Unless otherwise stated, we assume that all the operations on these indices are taken modulo the length of the collection. Furthermore, we may refer to negative indices and to indices greater than the length of the collection. In these cases, indices should also be understood modulo the length of the collection. For instance, if C is a circle of length 1, then the point at the position 2 is the same as the point at position 0. Similarly, if $x_1, \ldots, x_n$ is a circular ordering of objects, then $x_{n+1} = x_1$.

Let $X = x_1, \ldots, x_n$ be a finite ordering of objects (it is not important whether X is a linear or circular ordering). For $x_i, x_j \in X$, the *range* $[x_i, x_j]$ is defined as the linear ordering $x_i, x_{i+1}, \ldots, x_{j-1}, x_j$ where, as said before, all the operations are calculated modulo n. Similarly, the range $[x_i, x_j)$ is obtained by removing the last element from $[x_i, x_j]$, the range $(x_i, x_j]$ is obtained by removing the first element from $[x_i, x_j]$, and (x_i, x_j) is obtained by removing both the first and last elements from $[x_i, x_j]$. We introduce a new notation to work with linear orderings. For $1 \leq i \leq j \leq n$, define the *linear range* $\textsc{Lin}[x_i, x_j]$ as $[x_i, x_j]$, while for $i > j$ define $\textsc{Lin}[x_i, x_j]$ as the empty ordering. The linear ranges $\textsc{Lin}[x_i, x_j)$, $\textsc{Lin}(x_i, x_j]$ and $\textsc{Lin}(x_i, x_j)$ are defined analogously.

Sometimes we want to traverse all the elements in $[x_i, x_j]$ but going through x_i twice when $x_i = x_j$. Note that with our range notation this is impossible since $[x_i, x_i] = x_i$, while $[x_i, x_i)$, $(x_i, x_i]$ and (x_i, x_i) are all empty. We introduce a new notation for these cases. For $x_i, x_j \in X$, define the $\circ$-*range* as

$$[x_i \circ x_j] = \begin{cases} [x_i, x_{i-1}], x_i & \text{if } x_i = x_j \\ [x_i, x_j] & \text{otherwise} \end{cases}$$

As before, the $\circ$-range $(x_i \circ x_j]$ is obtained by removing the first element from $[x_i \circ x_j]$, the $\circ$-range $[x_i \circ x_j)$ is obtained by removing the last element from $[x_i \circ x_j]$, and the $\circ$-range $(x_i \circ x_j)$ is obtained by removing both the first and last elements from $[x_i \circ x_j]$.

2.6 Certifying algorithms

Recall that a *decision problem* Π consists of determining whether some instance I of Π satisfies a given property. The possible outputs for Π are YES, if I satisfies the property, or NO, otherwise. Thus, Π partitions the set of instances into YES-instances and NO-instances. A *positive certificate* for a YES-instance I is a piece of evidence proving that I is a YES-instance, while a *negative certificate* for a NO-instance I is a

piece of evidence proving that I is a NO-instance. In general, a *certificate* for I is either a positive or negative certificate. A *certifying algorithm* for Π is an algorithm that, for every instance I, outputs the result of Π on I together with a certificate for I. An algorithm that tests the validity of a certificate is called an *authentication algorithm*. That is, an authentication algorithm takes the instance and a certificate as its input, and outputs YES if and only if the certificate is correct for the instance.

There are many reasons why certifying algorithms are preferred over decision algorithms. From an end user perspective, accepting a YES-NO answer with no evidence from a black box seems like an act of faith. Moreover, the YES-NO answer is sometimes not enough for solving the user problems, *e.g.* when the user requires a UIG model of a graph. From a software engineering point of view, certificates can be used for debugging and testing purposes because an algorithm's implementation may hide bugs even though the algorithm is proven correct. In [KMMS06] there is a much deeper analysis about the importance of having certifying algorithms. We will follow the principles described in that paper for our certifying algorithms.

3 Subclasses of normal Helly circular-arc graphs

One of the most important algorithmic properties about interval models is the fact that they are Helly. The first linear-time recognition algorithm [BL76] is based on this fact, as well as are some of the newer recognition algorithms [HMPV00]. When interval models are generalized to circular-arc models, the Helly property is completely lost. The class of Helly circular-arc graphs lies between the CA and IG classes and, for this reason, HCA graphs preserve a lot of nice properties of interval graphs that are lost even for UCA graphs. Usually, these properties involve the cliques of the graphs. Just to give one of the many examples, consider the two theorems below.

Theorem 3.0.1 ([GH64, FG65])**.** *Let G be a graph. Then G is an interval graph if and only if there is a linear ordering of the cliques of G such that, for every vertex v, the cliques containing v appear consecutively in the ordering.*

Theorem 3.0.2 ([Gav74])**.** *Let G be a graph. Then G is an HCA graph if and only if there is a circular ordering of the cliques of G such that, for every vertex v, the cliques containing v appear consecutively in the ordering.*

It is not hard to find a counterexample to Theorem 3.0.2 when HCA graphs are replaced with circular-arc graphs. Even if we restrict ourselves to UCA graphs, we can still find counterexamples, as the $\overline{3K_2}$ graph whose only circular-arc model is depicted in Figure 3.1. Since UCA graphs are not Helly, we can expect the jump from UIG graphs to UCA graphs to be as big as it is the jump from interval graphs to circular-arc graphs. The same can be said about the jump from UIG graphs to PCA graphs. To see how far are UIG graphs from UCA and PCA graphs, we can study the classes of UHCA and PHCA graphs.

There is a second property of interval graphs that is lost for circular-arc graphs. This property has to do with the neighborhood of each vertex v. In an interval model $\mathcal{I}$, the submodel of $\mathcal{I}$ induced by the closed neighborhood of I is an interval model, for every $I \in \mathcal{I}$. This property holds neither for general HCA nor for general PCA models. Even more, there are some graphs, such as the 4-wheel graph (see Figure 3.3), for which no HCA model and no PCA model satisfies this property. However, if we restrict our attention to an NHCA model $\mathcal{M}$, we can see that the submodel of $\mathcal{M}$ induced by the

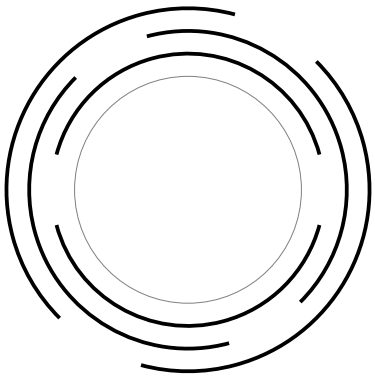


Figure 3.1: The unique circular-arc model of $3\overline{K_2}$. There is no circular ordering of the cliques of $\overline{3K_2}$ such that all the vertices belong to consecutive cliques in the ordering

closed neighborhood of A is an interval model for every $A \in \mathcal{A}(\mathcal{M})$. Hence, NHCA graphs will retain many properties of interval graphs that deal with the "intervality" of the neighborhood of a vertex. The local intervality property of NHCA graphs follows from the next theorem by Lin and Szwarcfiter.

Theorem 3.0.3 ([JLM+09, LS06])**.** *A circular-arc model $\mathcal{M}$ is HCA if and only if*

(i) if three arcs of $\mathcal{M}$ cover $C(\mathcal{M})$, then two of them also cover it, and

(ii) the intersection graph of $\overline{\mathcal{M}}$ contains no induced hole.

Corollary 3.0.4. *If an NCA model $\mathcal{M}$ is not HCA then three arcs of $\mathcal{M}$ cover $C(\mathcal{M})$.*

Proof. Suppose that no three arcs of $\mathcal{M}$ cover $C(\mathcal{M})$ and yet $\mathcal{M}$ is not HCA. Then, by Theorem 3.0.3, the intersection graph of $\overline{\mathcal{M}}$ has an induced hole $v_1, \ldots, v_k$, for some $k \geq 4$. Thus, the arcs $\overline{A_1}$, $\overline{A_3} \in \mathcal{A}(\overline{\mathcal{M}})$ corresponding to vertices v_1 and v_3 do not intersect and, therefore, A_1 and A_3 are arcs of $\mathcal{A}(\mathcal{M})$ that cover $C(\mathcal{M})$, *i.e.*, $\mathcal{M}$ is not NCA. □

As a corollary we obtain Theorem 3.0.5. For the sake of simplicity, we will take this property as an alternative definition of NHCA, PHCA and UHCA graphs. Hence, sometimes we omit the reference to this theorem.

Theorem 3.0.5. *A graph is an XHCA graph if and only if it admits an XCA model in which there are no two nor three arcs covering the circle, for $X \in \{N, P, U\}$.*

In this chapter we study these HCA subclasses. We first present a summary, without all the proofs, of many nice properties of interval and UIG graphs that are preserved

for NHCA and PHCA graphs, respectively. This summary is presented in Section 3.1 as a motivation for the study of these HCA subclasses. Of course, there is a completely theoretical motivation, which is to study all the circular-arc subclasses that are obtained by the intersection of the most well known subclasses of circular-arc models. Next, in Section 3.2, we show forbidden induced subgraphs characterizations of NHCA, PHCA and UHCA graphs. For the class of NHCA graphs this characterization is partial, we only show which HCA graphs are not NHCA. These characterizations immediately imply $O(n+m)$ time recognition algorithms for the three classes. We postpone the improvements of these algorithms until Chapter 5. Finally, in Section 3.3, we prove some of the results that are used as motivation in Section 3.1, and we study the HCA subclasses from the point of view of vertex enumerations and graph orientations.

3.1 Why study subclasses of NHCA graphs?

In this section we motivate the study of the Helly subclasses that are defined in this thesis. For this purpose, we show several properties that hold in these subclasses and are not easy to generalize to the more general classes of circular-arc graphs. These properties are generalizations or slight modifications of properties that hold for the interval graph subclasses. Some of these properties have algorithmic implications that can perhaps be used to generalize some of the algorithms for interval graphs.

We begin studying the forbidden induced subgraph structure of NHCA graphs and its relation with the structure of interval graphs. For interval graphs the structure is described by the following theorem by Roberts.

Theorem 3.1.1 ([Rob69])**.** *Let G be an interval graph. Then the following are equivalent:*

(i) *G does not contain $K_{1,3}$ as an induced subgraph.*

(ii) *G is a proper interval graph.*

(iii) *G is a unit interval graph.*

For NHCA graphs we obtain a similar result.

Theorem 3.1.2. *Let G be an NHCA graph. Then G is a PHCA graph if and only if G contains no $K_{1,3}$ as an induced subgraph.*

Proof. Clearly, the $K_{1,3}$ graph is not PCA, so it is neither PHCA. For the converse, suppose that G is an NHCA graph with an NHCA model $\mathcal{M}$. Sort every extreme sequence of $\mathcal{M}$ in such a way that no arc with an extreme in the sequence is properly contained in some other arc with an extreme in the sequence. This sorting does not

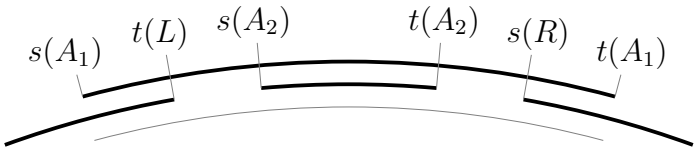


Figure 3.2: The submodel of $\mathcal{M}'$ induced by A_1, L, A_2, R in Theorem 3.1.2. From left to right, we show each case according to whether $L = R$, $L \neq R$ and $L \cap R \neq \emptyset$, and $L \cap R = \emptyset$

change the intersections between the arcs of $\mathcal{M}$, thus the sorted model $\mathcal{M}'$ is also an NHCA model of G. Now, if some arc $A_1 \in \mathcal{A}(\mathcal{M}')$ is contained in some other arc $A_2 \in \mathcal{A}(\mathcal{M}')$, it is because there is some ending point between $s(A_1)$ and $s(A_2)$, and there is some beginning point between $t(A_2)$ and $t(A_1)$. In other words, there are two arcs L and R of $\mathcal{M}'$ such that $s(A_1), t(L), s(A_2), t(A_2), s(R)$ and $t(A_1)$ appear in this order in $\mathcal{M}'$ (see Figure 3.2). Since $\mathcal{M}'$ is NHCA then $L \neq R$ and $L \cap R = \emptyset$, *i.e.*, the intersection graph of A_1, L, A_2, R is isomorphic to $K_{1,3}$. □

Implication $(ii) \Longrightarrow (iii)$ of Theorem 3.1.1 is lost, because the graph $CI(n, k)$ is PHCA but not UHCA for every $n > 3k$. Indeed, by definition, every $CI(n, k)$ graph admits a PCA model in which no family with at most $\lceil n/k \rceil > 3$ arcs cover the circle. To retain the "proper=unit" property with a similar definition as the one in Theorem 3.0.5, we should ask that no set of arcs cover the circle. This is because the unit length property of UIG graphs is global, and it does not depend only on the neighborhood of each vertex.

Next, we consider clique graphs of HCA graphs and NHCA graphs. The *clique graph* $K(G)$ of a graph G is the intersection graph of the cliques of G, and for a class $\mathcal{C}$ of graphs we denote $K(\mathcal{C}) = \{K(G) \mid G \in \mathcal{C}\}$. Clique graphs of interval graphs were studied by Hedman in [Hed84], where he proved that the PIG and K(IG) classes are equal. On the other hand, clique graphs of HCA graphs were first studied by Durán and Lin in [DL01], where they proved that clique graphs of HCA graphs are both PCA and HCA. The question that motivated us to study PHCA graphs first, and NHCA graphs later, was the following: is the class of PHCA graphs equal to the K(HCA) class?. The answer is no, but almost, as it is shown in the next theorem. The proof of this theorem is postponed to Theorem 8.1.6.

Theorem 3.1.3 (Theorem 8.1.6)**.** *Let H be a graph and U the set of universal vertices of H. Then H is the clique graph of some HCA graph G if and only if:*

(i) H is a PHCA graph or

(ii) $H \setminus U$ is a co-bipartite PHCA graph and $|U| \geq 2$.

However, an analogous result to the one obtained by Hedman holds for NHCA graphs (see Theorem 8.1.7). That is, the K(NHCA) and the PHCA classes of graphs are equal. Hedman [Hed84] proved also that for every PIG graph G there is a PIG graph H such that $K(H)$ is isomorphic to G. That is, the PIG and the K(PIG) classes are also equal. The same result holds for PHCA graphs, *i.e.*, for every PHCA graph G there is a PHCA graph H such that $K(H)$ and G are isomorphic (see Theorem 8.1.5).

Now we move into the third property that can be preserved by restricting the attention to PHCA graphs. A classic characterization of interval graphs is that interval graphs are those graphs whose clique matrix has the consecutive-ones property for columns [GH64]. Similarly, a graph is HCA if and only if its clique matrix has the circular-ones property for columns [Gav74]. The definitions of clique matrix, consecutive and circular-ones properties are given in Section 3.3. We can think that the consecutive and circular-ones properties for columns are due to the Helly property of the interval and HCA graphs, respectively. On the other hand, it is well known that a graph is a proper interval graph if and only if its clique matrix has the consecutive-ones property for both its rows and its columns (see *e.g.* [DG99, Fis85, Gar07]). An analogous theorem for the circular-ones property can be proved for PHCA graphs, as we shall see in Section 3.3. That is, the clique matrix of a graph has the circular-ones properties for both rows and columns if and only if the graph is a PHCA graph.

Finally, the "local interval" property of NHCA graphs implies that a lot of simple algorithms that work for interval graphs also work for NHCA graphs. Consider for example the problem of finding every clique of an interval graph. A *clique segment* in an interval model $\mathcal{M}$ is a segment (s, t) where s is a beginning point and t is an ending point. In $\mathcal{M}$, every clique point belongs to a clique segment, and all the points in a clique segment are clique points. Thus, the set of intervals that contain a given clique segment induce a clique and each clique is represented by exactly one clique segment. It is trivial to find every clique of an interval graph by computing the clique segments in one of its interval models. For HCA graphs the situation is quite different. First, not every segment of the form (s, t) is formed by clique points (with s and t being two consecutive beginning and ending point, respectively). Second, not every clique of the model is represented by exactly one clique segment, *i.e.*, there are several clique segments whose containing arcs induce the same clique. Thus, the algorithm to find every clique point is not so trivial (see Section 8.2). However, NHCA graphs share the same property as interval graphs: every segment (s, t) represents a clique, and every clique is represented by exactly one of such clique segments. Thus, the same algorithm for interval graphs works as well for NHCA graphs, after some minor changes. Another example for which the local interval property has some advantages is the dynamic recognition. It is somehow easy to insert a new edge into a PHCA graph, but it is not so easy for PCA graphs (see Section 6.6). In Section 3.3.3 we further develop these intervality concepts for NHCA and PHCA graphs.

3.2 The structure of the HCA subclasses

In this section we present characterizations by forbidden induced subgraphs for the classes of NHCA, PHCA, and UHCA graphs. These characterizations follow the same spirit of Theorem 3.1.2, in the sense that they show when a graph from some class belongs to a subclass of it. The characterizations shown in this section immediately yield $O(n+m)$ time recognition algorithms for all the classes. In Chapter 5 we further discuss the algorithmic implications of these characterizations.

The following proposition is used several times throughout this section. We include it here for this reason.

Proposition 3.2.1. *Let $\mathcal{M}$ be a circular-arc model and $B_1, \ldots, B_k$ be a hole in $\mathcal{M}$. If $A \in \mathcal{A}(\mathcal{M})$ is an arc which is not contained in any other arc, then either:*

(i) *A and B_i cover the circle for some $1 \leq i \leq k$,*

(ii) *A, B_i and B_{i+1} cover the circle for some $1 \leq i \leq k$,*

(iii) *$A \subset (B_i \cup B_{i+1}) \setminus (B_{i+2} \cup \ldots \cup B_{i-1})$ for some $1 \leq i \leq k$, or*

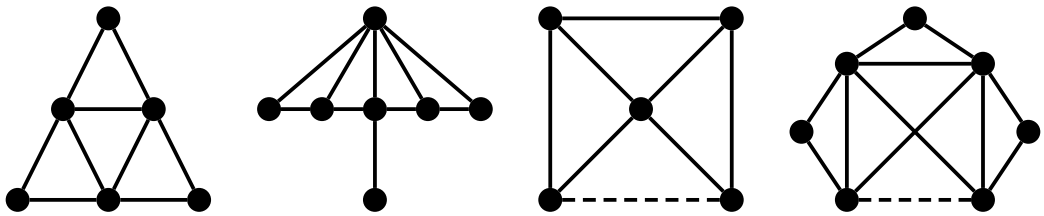
(iv) *$A, B_i, \ldots, B_j$ is an induced hole of $\mathcal{M}$, for some $1 \leq i, j \leq k$.*

Proof. If $A = B_i$ for some $1 \leq i \leq n$, then *iv* follows. Suppose then that A is not an arc of the hole. Traverse $C(\mathcal{M})$ from $t(A)$ and let B_i be the arc whose beginning point appears first. If $s(B_i) \in A$ then A and B_{i-1} must cover the circle. Otherwise, if $t(B_i) \in A$ then A, B_{i-1} and B_i cover the circle. Finally, suppose that neither $s(B_i)$ nor $t(B_i)$ are points of A, and let B_j be the arc whose ending point appears first in a counterclockwise traversal of $C(\mathcal{M})$ from $s(A)$. If $i-1 = j+1$, then it follows that $A \subset B_{i-1}$, which is a contradiction to the fact that A is not properly contained in any other arc. Otherwise, $A, B_{i-1}, \ldots, B_{j+1}$ induce a hole whenever $i-1 \neq j+2$ or $A \subset (B_{i-2} \cup B_{i-1}) \setminus (B_i \cup \ldots \cup B_{i-3})$ whenever $i-1 = j+2$. □

3.2.1 Normal Helly circular-arc graphs

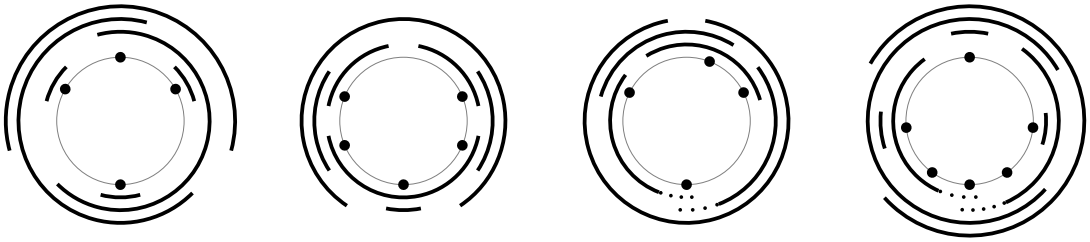
We begin with the problem of determining when does an HCA graph admit an NHCA model. Wheels, 3-suns, rising suns, and umbrellas are the forbidden subgraphs involved in the characterization. The 3-sun and the umbrella are the graphs depicted in Figure 3.3 (a) and (b), respectively. The *n-wheel*, for $n \geq 4$, is the graph obtained by inserting one universal vertex into a hole of length n (see Figure 3.3 (c)). Finally, the *n-rising sun*, for $n \geq 4$, is the graph that is obtained from a path $v_2, \ldots, v_{n-1}$ by first adding two universal vertices v_1 and v_n, and then inserting three vertices w_1, w_{n-1}, w_n such that w_i is adjacent only to v_i and to v_{i+1}, for $i \in \{1, n-1, n\}$ (see Figure 3.3 (d)). The 3-sun

graph is denoted by S_3, the umbrella is denoted by U, the n-wheel is denoted by W_n, and the n-rising sun is denoted by R_n.



(a) 3-sun (b) Umbrella (c) Wheels (d) Rising suns

Figure 3.3: HCA graphs that are not NHCA. Wheels have at least 5 vertices and rising suns have at least 7 vertices.



(a) 3-sun (b) Umbrella (c) Wheels (d) Rising suns

Figure 3.4: HCA models of the graphs in Figure 3.3. Points in the circles are used to mark the clique points.

Interval graphs can be partially characterized as HCA graphs using the following result by Lekkerkerker and Boland.

Theorem 3.2.2 ([LB63])**.** *An HCA graph is an interval graph if and only if it does not contain holes, 3-suns, rising suns, nor umbrellas as induced subgraphs.*

The only difference between the characterization by Lekkerkerker and Boland and the characterization of NHCA graphs is that holes are replaced by wheels. We analyze first how do HCA models of NHCA graphs look like.

Theorem 3.2.3. *Let $\mathcal{M}$ be an HCA model of a graph G. Then $\mathcal{M}$ is equivalent to an NHCA model if and only if $\mathcal{M}$ contains no wheels, 3-suns, rising suns, nor umbrellas as induced submodels.*

Proof. Let $\mathcal{M}'$ be any HCA model of a wheel W_k, for $k \geq 4$. Such a model exists as it is depicted in Figure 3.4 (c). Model $\mathcal{M}'$ has at least one clique point for each clique of W_k; the universal arc covers all of these clique points, while each of the other arcs of the

submodel covers exactly two of them. Consequently, there are two arcs covering $C(\mathcal{M}')$, *i.e.*, $\mathcal{M}'$ is not normal. Thus, whenever $\mathcal{M}$ is equivalent to an NHCA model, $\mathcal{M}$ does not contain an induced submodel of W_k. The proofs for the 3-sun, the umbrella and the rising suns follow analogously.

For the converse, suppose that $\mathcal{M}$ contains none of the forbidden submodels and yet $\mathcal{M}$ has two arcs A_1 and A_2 that cover the circle which, w.l.o.g., are not contained in other arcs of $\mathcal{M}$. Suppose also, to obtain a contradiction, that $\mathcal{M}$ has k arcs $B_1, \ldots, B_k$ that induce a hole in that order. If $A_1 = B_i$ for some $1 \leq i \leq k$, then A_2 is adjacent to all the arcs of the hole, *i.e.*, $A_2, B_1, \ldots, B_k$ induce a wheel. Otherwise, we ought to consider three cases by Proposition 3.2.1:

Case 1: A_1 and B_i cover the circle for some $1 \leq i \leq k$. In this case, A_1 intersects all the arcs of the hole, *i.e.*, $A_1, B_1, \ldots, B_k$ induce a wheel. Therefore, this case cannot happen.

Case 2: A_1 is contained in $B_i \cup B_{i+1}$ for some $1 \leq i \leq k$. In this case, A_2 intersects all the arcs of $B_1, \ldots, B_k$, thus $A_2, B_1, \ldots, B_k$ induce a wheel. This case is also impossible.

Case 3: $A_1, B_i, \ldots, B_j$ is a hole of $\mathcal{M}$ for some $1 \leq i, j \leq k$. As in Case 2, A_2 intersects all the arcs of this new hole. Thus, $A_2, A_1, B_i, \ldots, B_j$ induce a wheel, again a contradiction.

Since none of the cases can occur, it follows that $\mathcal{M}$ has no holes. Thus, G is hole-free and it contains no rising suns, 3-suns, nor umbrellas, which implies that G is an interval graph by Theorem 3.2.2. Consequently, $\mathcal{M}$ is equivalent to some interval model of G. □

The characterization by minimal forbidden induced subgraphs then follows easily.

Corollary 3.2.4. *An HCA graph is NHCA if and only if it contains no wheels, 3-suns, rising suns, nor umbrellas as induced subgraphs.*

There is also a strong consequence for circular-arc models that can be used for negative certification and which is also useful for the characterization of NHCA graphs in terms of NCA graphs.

Corollary 3.2.5. *Every circular-arc model of a non-interval NHCA graph is NHCA.*

Proof. Let $\mathcal{M}$ be any circular-arc model of a non-interval NHCA graph G. Since G is NHCA then we can apply the algorithm in [LS06] with input $\mathcal{M}$, to obtain an HCA model $\mathcal{M}'$ of G. By definition, $\mathcal{M}'$ is equivalent to some NHCA model, because G is NHCA. If $\mathcal{M}'$ has two arcs that cover the circle, then we can use the same arguments as in Theorem 3.2.3 to prove that G is an interval graph, a contradiction. Otherwise, $\mathcal{M}'$ is NHCA, *i.e.*, there are neither two nor three arcs that cover $C(\mathcal{M}')$.

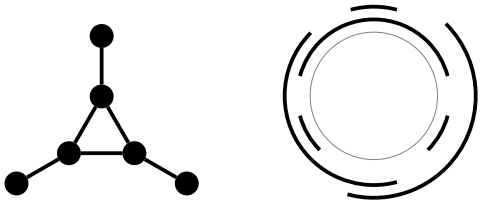


Figure 3.5: The tent graph and one of its NCA models

The algorithm in [LS06] works in such a way that every arc of $\mathcal{M}$ is included in some arc of $\mathcal{M}'$. Thus, in $\mathcal{M}$ there could neither be two nor three arcs covering the circle, *i.e.*, $\mathcal{M}$ is NHCA. □

Now we proceed with the characterization of NHCA graphs in terms of NCA graphs. In this case, the forbidden induced subgraphs are the wheels, the 3-sun, the rising suns, the umbrella, and the *tent graph* $\overline{S_3}$ (see Figure 3.5). The proof follows the same scheme as before, we analyze the NCA models of NHCA graphs and the characterization is obtained as a corollary.

Theorem 3.2.6 ([LB63])**.** *An NCA graph is an interval graph if and only if it does not contain holes,* 3*-suns, rising suns, umbrellas, nor tents as induced subgraphs.*

Theorem 3.2.7. *Let $\mathcal{M}$ be an NCA model of a graph G. Then $\mathcal{M}$ is equivalent to an NHCA model if and only if $\mathcal{M}$ contains no wheels,* 3*-suns, rising suns, umbrellas, nor tents, as induced submodels.*

Proof. Wheels, 3-suns, rising suns, umbrellas, and tents admit circular-arc models that are not NHCA (see Figures 3.4 and 3.5). Hence they are not NHCA, by Corollary 3.2.5.

The converse is somehow similar to the converse of Theorem 3.2.3, but it needs a few tweaks. Suppose that $\mathcal{M}$ contains none of the forbidden submodels and yet $\mathcal{M}$ has three arcs A_1, A_2 and A_3 that cover the circle. We may assume that none of these three arcs is contained in any other arc because $\mathcal{M}$ has no two arcs that cover the circle. To obtain a contradiction, suppose that $\mathcal{M}$ has k arcs $B_1, \ldots, B_k$ that induce a hole $\mathcal{H}$ in this order.

Claim 1: A_j, B_i, B_{i+1} do not cover the circle, for $1 \leq i \leq k$ and $1 \leq j \leq 3$. Otherwise, A_j intersects all the arcs of $\mathcal{H}$, *i.e.*, $\mathcal{H} \cup \{A_j\}$ induces a wheel.

Claim 2: There is a hole $\mathcal{H}' \subset \mathcal{H} \cup \{A_1, A_2\}$ that contains at least one of A_1 and A_2. Suppose, to obtain a contradiction, that this is not the case. Then, by Proposition 3.2.1 and Claim 1, it follows that $A_1 \subset (B_i \cup B_{i+1}) \setminus (B_{i+2}, \ldots, B_{i-1})$ and that $A_2 \subset (B_j \cup B_{j+1}) \setminus (B_{j+2}, \ldots, B_{j-1})$, for $1 \leq i, j \leq n$. By Claim 1, A_3 does not cover the circle with B_i and B_{i+1}, thus $i \neq j$. So, either $t(A_1) \in B_{i+1} \cap A_2$ and

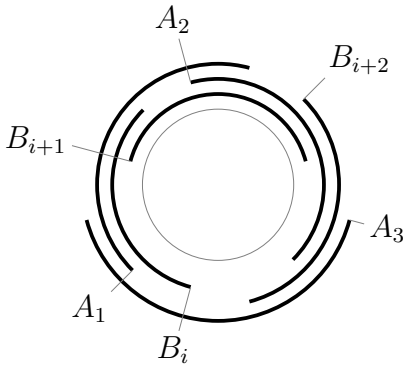


Figure 3.6: Claim 2 of Theorem 3.2.7, $A_1, B_{i+1}, A_2, A_3, B_i$ induce a wheel.

$j = i + 1$ or $t(A_2) \in B_i \cap A_1$ and $i = j + 1$. Assume the former w.l.o.g., thus $A_2 \subset (B_{i+1} \cup B_{i+2}) \setminus (B_{i+2}, \ldots, B_i)$ (see Figure 3.6). Now, consider the position of arc A_3. Since A_1, A_2, A_3 cover the circle, then A_3 crosses both $t(B_{i+2})$ and $s(B_i)$. By Claim 1, A_3 crosses neither $s(B_{i+1})$ nor $t(B_{i+1})$, thus $A_1, B_{i+1}, A_2, A_3, B_i$ induce a wheel where A_1 is the universal arc.

By Claim 2, there is a hole $\mathcal{H}' \subset \mathcal{H} \cup \{A_1, A_2\}$ that contains at least one of A_1, A_2, say A_1. If we exchange $\mathcal{H}$ with $\mathcal{H}'$ and A_1 with A_3 in Claim 2, we obtain that there is a hole $\mathcal{H}'' \subset \mathcal{H}' \cup \{A_2, A_3\}$ that, w.l.o.g., contains A_2 as well. If A_1 is not an arc of $\mathcal{H}''$, it is because A_1 is covered by two arcs of the hole, one of which is A_2. In this case, call B to the arc of $\mathcal{H}''$ that together with A_2 covers A_1. Otherwise, call $B = A_1$. Summing up, $\mathcal{H}'' = B, A_2, B_i, \ldots, B_j$, for some pair $1 \leq i, j \leq k$. Since A_1, A_2, A_3 cover the circle then B, A_2, A_3 cover the circle, thus A_3 must intersect all the arcs of $\mathcal{H}''$, a contradiction. This contradiction appears because we assume that there is a hole in $\mathcal{M}$. Therefore, G contains no holes, 3-suns, umbrellas, rising suns, nor tents as induced subgraphs. This implies that G admits an interval model equivalent to $\mathcal{M}$ by Theorem 3.2.6. □

Corollary 3.2.8. *An NCA graph is NHCA if and only if it contains no wheels, 3-suns, rising suns, umbrellas, nor tents as induced subgraphs.*

The proofs of Theorems 3.2.3 and 3.2.7 can be combined to obtain a result characterizing when a circular-arc graph admits an NHCA model. We are not going to prove the theorem to avoid repetitions, instead we just give a sketch of the proof.

Theorem 3.2.9. *A circular-arc graph is NHCA if and only if it contains no wheels, 3-suns, rising suns, umbrellas, nor tents as induced subgraphs.*

Proof. Wheels, 3-suns, rising suns, umbrellas and tents are not NHCA by Theorem 3.2.7.

For the converse, suppose that there are two arcs A_1, A_2 that cover a circular-arc model $\mathcal{M}$, and that $\mathcal{M}$ has an induced hole $B_1, \ldots, B_k$. Then, by Proposition 3.2.1, we can either find a wheel in $\mathcal{M}$ as in Theorem 3.2.3 or A_1, B_i, B_{i+1} cover the circle for some $1 \leq i \leq k$. In this last case, A_1 is universal to all the arcs of the hole which also implies that $\mathcal{M}$ contains an induced wheel. Then, as before, $\mathcal{M}$ is an interval model or $\mathcal{M}$ is NCA. If $\mathcal{M}$ is NCA then the result follows from Theorem 3.2.7. □

3.2.2 Proper Helly circular-arc graphs

Up to this point we have characterized which circular-arc (resp. HCA, NCA) graphs admit an NHCA model and which NHCA graphs admit a PHCA model. In this section we characterize which PCA graphs are also PHCA. For the proof we may use the same arguments of Theorem 3.2.3 to show that every PCA model of a non-interval PHCA graph is in fact a PHCA model. This would yield an elegant short proof. But instead, we prefer to do a constructive proof that shows how can a PCA model of an interval graph be transformed into a proper interval model. As we will see in Chapter 5, this proof yields an $O(n)$ time algorithm to transform a PCA model into a PHCA model.

Theorem 3.2.10. *Let $\mathcal{M}$ be a PCA model of a graph G. Then the following are equivalent:*

(i) $\mathcal{M}$ is equivalent to a PHCA model.

(ii) $\mathcal{M}$ contains no induced submodel of W_4 and S_3.

(iii) $U_1(\mathcal{M})$ is HCA or $U_0(\mathcal{M})$ is a PIG model.

Proof.

$(i) \implies (ii)$: neither W_4 nor S_3 are NHCA graphs by Theorem 3.2.3, thus $\mathcal{M}$ cannot have induced submodels of them.

$(ii) \implies (iii)$: let $\mathcal{M}$ be a PCA model, containing no induced submodels of W_4 and S_3. By Lemma 2.4.7, $\mathcal{M}_1 = U_1(\mathcal{M})$ is an NPCA model. If $\mathcal{M}_1$ is not an HCA model, Corollary 3.0.4 implies that $\mathcal{M}_1$ contains three arcs A_1, A_2, A_3 covering $C(\mathcal{M}_1)$. No two arcs cover $C(\mathcal{M}_1)$, thus we may assume that $s(A_1), t(A_3), s(A_2), t(A_1), s(A_3), t(A_2)$ appear in this order in a traversal of $C(\mathcal{M}_1)$. First, we prove that one of the above three arcs must be universal. Suppose the contrary. Then, there exist arcs B_1, B_2 and B_3 such that B_i does not intersect A_i, for $i \in \{1, 2, 3\}$. However, since $\mathcal{M}_1$ is a proper model, it follows that B_i intersects A_j, A_k for $\{j, k\} = \{1, 2, 3\} \setminus \{i\}$. The latter leads to a contradiction because the intersection graph of $\{A_i, B_i\}_{i \in \{1,2,3\}}$ is isomorphic to S_3, when B_1, B_2, B_3 are pairwise disjoint, or it contains an induced W_4. Consequently, one of A_1, A_2, A_3, say A_1, is the only universal arc of $\mathcal{M}_1$.

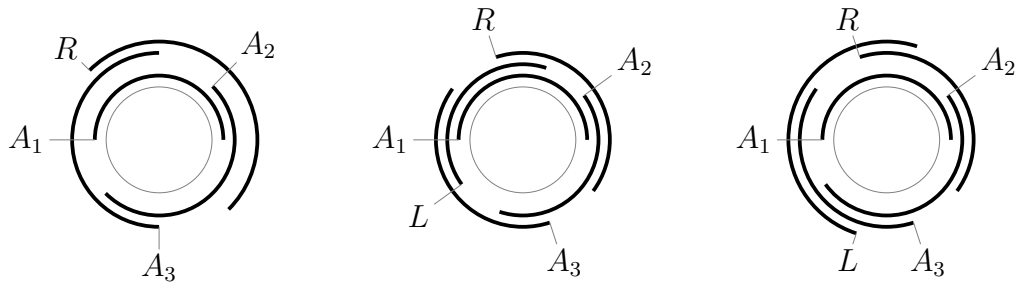


Figure 3.7: Proof of Theorem 3.2.10. If A_1 is completely covered then G is not a PHCA graph.

Next, we examine the arc A_1 in $\mathcal{M}_1$. We will prove that there is no pair of arcs L, R such that $t(L) \in A_1$, $s(R) \in A_1$, and $s(R)$ precedes $t(L)$ in a traversal of $C(\mathcal{M})$ from $s(A_1)$. To obtain a contradiction for this fact, assume the contrary and discuss the following alternatives.

Case 1: $L = A_3$. In this situation, R, A_2, A_3 are three arcs covering C. Because A_1 is the unique universal arc of $\mathcal{M}_1$, we know that R, A_2, A_3 are not universal. Consequently, as above, $\mathcal{M}_1$ contains a submodel of W_4 or S_3, a contradiction (Figure 3.7(a)).

Case 2: $R = A_2$. Similar to Case 1.

Case 3: $L \neq A_3$ and $R \neq A_2$. By Cases 1 and 2, above, it suffices to examine the situation where $s(R), t(L) \in (t(A_3), s(A_2))$. Suppose $L \cap A_2 = R \cap A_3 = \emptyset$. In this case, the arcs A_1, A_2, A_3, L, R form a forbidden W_4, impossible (Figure 3.7(b)). Alternatively, let $L \cap A_2 \neq \emptyset$. Then the arcs A_2, L, R cover the circle and none of them is the universal arc A_1, an impossibility (Figure 3.7(c)). The situation $R \cap A_3 \neq \emptyset$ is similar.

By the above cases, we conclude that all the ending points must precede the beginning points in $(s(A_1), t(A_1))$. Let t and s be the last ending point and the first beginning point inside $(s(A_1), t(A_1))$, respectively. Taking into account that A_1 is the only universal arc, we conclude that no point of the segment $(t, s) \subset A_1$ of $C(\mathcal{M})$ can be contained in any arc of $\mathcal{M}$ except A_1. Hence $\mathcal{M}_0 = \mathcal{M}_1 \setminus \{A_1\}$ is a PIG model.

$(iii) \Longrightarrow (i)$: suppose first that $\mathcal{M}_1 = U_1(\mathcal{M})$ is an HCA model. By Lemma 2.4.4, we can include in the model all the universal arcs that have been possibly removed from it, obtaining a model equivalent to $\mathcal{M}$ that is both PCA and HCA.

Next, suppose that $\mathcal{M}_0 = U_0(\mathcal{M})$ is a PIG model and $\mathcal{M}_1$ is not an HCA model, thus $\mathcal{M}_0 \neq \mathcal{M}_1$. Since $\mathcal{M}_0 \neq \mathcal{M}_1$ then it follows that $\mathcal{M}_1$ has some universal arc A. We

prove that $\mathcal{M}' := \mathcal{M}_0 \cup \{\overline{A}\}$ is a PIG model equivalent to $\mathcal{M}_1$. By Lemma 2.4.7, $\mathcal{M}_1$ is an NPCA model, thus there is exactly one extreme of each non-universal arc inside A. Hence, $\overline{A}$ contains exactly one extreme of each non-universal arc. This means that $\overline{A}$ is a universal arc of $\mathcal{M}'$, and that $\mathcal{M}'$ is an NPCA model equivalent to $\mathcal{M}_1$. On the other hand, since $\mathcal{M}_0$ is an interval model, it follows that there is some point $p \in A$ which is crossed only by A in $\mathcal{M}_1$. Therefore, p is not crossed by any arc of $\mathcal{M}'$, which implies that $\mathcal{M}'$ is an interval model as well. Summing up, $\mathcal{M}'$ is a PIG model equivalent to $\mathcal{M}_1$. Duplicating the universal arc of $\mathcal{M}'$ we can include all the universal arcs that were removed from $\mathcal{M}_1$, to obtain a PIG model of G. □

The characterization in terms of forbidden subgraphs is depicted in the corollary below.

Corollary 3.2.11. *A PCA graph is PHCA if and only if it contains no induced W_4 and no induced S_3.*

In the implication $(ii) \Longrightarrow (iii)$, the non-Helly PCA model $\mathcal{M}_1 = U_1(\mathcal{M})$ has a universal arc A whose removal yields the interval model $U_0(\mathcal{M})$. This arc A can be replaced by $\overline{A}$, and then all the arcs of $\mathcal{M} \setminus \mathcal{M}_1$ can be inserted once again into $\mathcal{M}_0$ by duplicating $\overline{A}$ as in $(iii) \Longrightarrow (i)$. The model so obtained is a PIG model equivalent to $\mathcal{M}$. Thus, if some PHCA graph admits a non-Helly PCA model then the graph is in fact a PIG graph. The following corollary, which is the analogous of Corollary 3.2.5, reflects this fact.

Corollary 3.2.12. *Every PCA model of a non-interval PHCA graph is PHCA.*

Proof. By Corollary 3.2.5, every PCA model of a non-interval PHCA graph is also NHCA. Thus, the model is both PCA and HCA. □

3.2.3 Unit Helly circular-arc graphs

Theorem 3.2.10 describes when can a PCA model be transformed into an equivalent PHCA model. An almost verbatim copy of its proof can be used to characterize those UCA models which are equivalent to some UHCA model. However, this characterization can be done easily once Corollary 3.2.12 is known.

Theorem 3.2.13. *A graph is UHCA if and only if it is PHCA and UCA. Moreover, every UCA model of a non-interval UHCA graph is UHCA.*

Proof. Clearly, every UHCA model is both PHCA and UCA. For the converse, let G be a PHCA and UCA graph and observe that G contains no induced $K_{1,3}$. If G is also an interval graph, then it is a UIG graph by Theorem 3.1.1. If G is not an interval graph then, by Corollary 3.2.12, every UCA model of G is also HCA. □

The partial forbidden induced subgraph characterizations of UHCA graphs are shown below.

Corollary 3.2.14. *A UCA graph is UHCA if and only if it contains no induced* W_4.

Proof. It follows from Theorems 3.2.13 and 2.4.6, Corollary 3.2.11, and the fact that $S_3 = CI(3,1)$. □

Corollary 3.2.15. *A PHCA graph is UHCA if and only if it contains no induced* $CI(n,k)$ *graph with* $n > 3k$.

Proof. By Theorem 2.4.6, UCA graphs contain no induced $CI(n,k)$ for $n > 3k$. For the converse, let G be a PHCA graph with no induced $CI(n,k)$ with $n > 3k$. By definition (see Chapter 2), every $CI(n,k)$ model with $2 < n < 3k$ has three arcs that together the circle. Then, since $CI(n,k)$ graphs are not interval graphs, it follows by Corollary 3.2.12 that $CI(n,k)$ graphs with $2 < n < 3k$ are not PHCA. So, in G there is no induced $CI(n,k)$ for $n > 2k$. Thus, Theorem 2.4.6 and Corollary 3.2.14 imply that G is a UHCA graph. □

The whole picture of the CA class hierarchy is depicted in Figure 3.8. Each box of the picture represents a subclass of circular-arc graphs. An upright edge from the box corresponding to the class $\mathcal{C}_1$ to the box corresponding to the class $\mathcal{C}_2$ means that $\mathcal{C}_1$ is properly contained in $\mathcal{C}_2$. Graphs that belong to $\mathcal{C}_2$ but not to $\mathcal{C}_1$ appear beside the edge corresponding to the inclusion of $\mathcal{C}_1$ in $\mathcal{C}_2$, except for the edge between the CA and the NCA classes since this family is unknown. Finally, the label $\mathcal{O}$ beside the edge between the CA and HCA classes represents the family of obstacles that were defined by Lin and Szwarcfiter in [LS06].

3.3 Some additional properties of the NHCA subclasses

In this section we prove some properties of the normal Helly subclasses that might be useful from an algorithmic point of view. Some of these were presented in Section 3.1 as natural generalizations of properties of interval graphs.

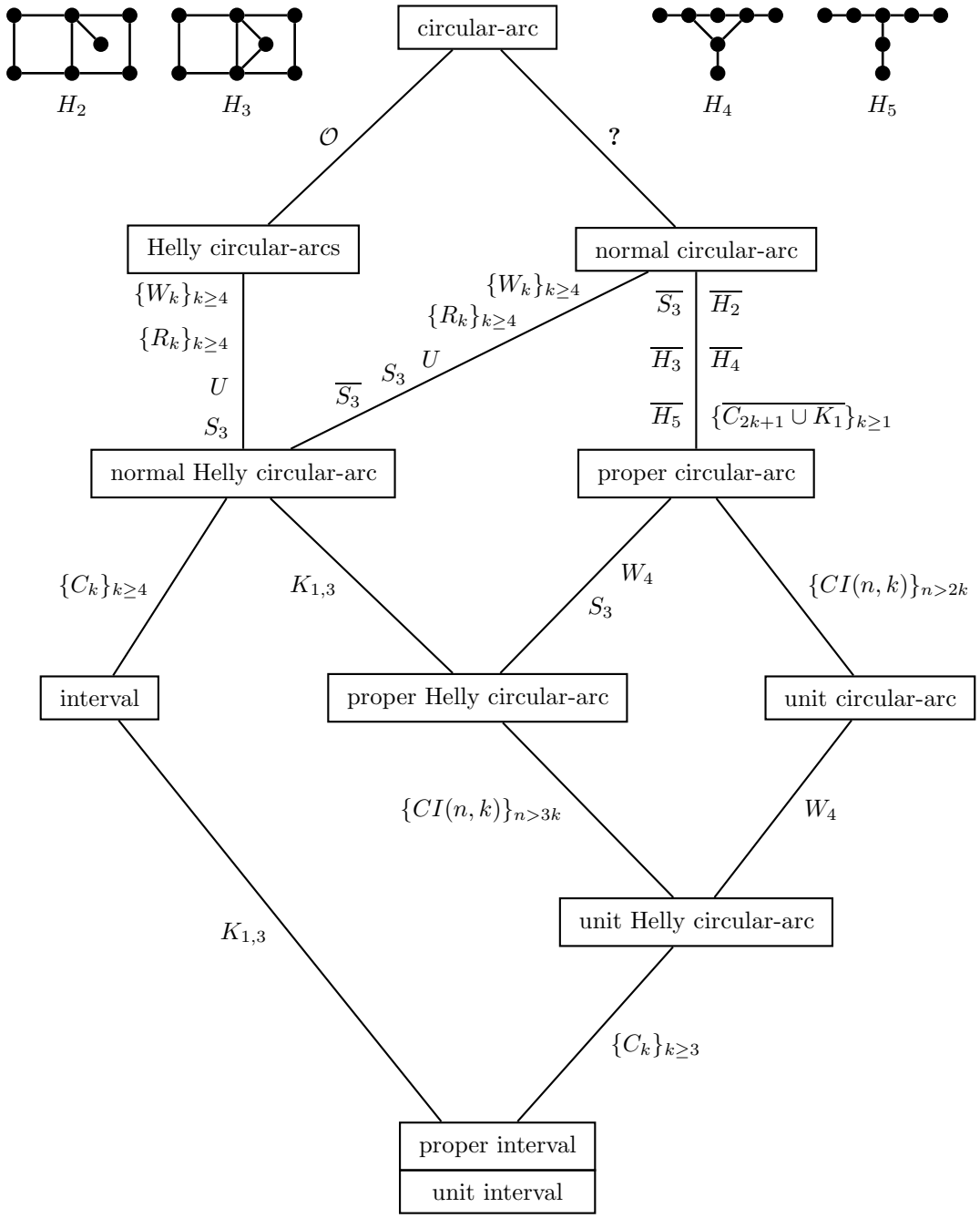


Figure 3.8: The class hierarchy of circular-arc graphs.

3.3.1 Counting NPHCA models of PHCA graphs

The first problem is to count how many NPHCA models a PHCA graph admits. This question of how many representations of a graph are there has been solved for both PIG graphs and co-connected PCA graphs; Roberts [Rob69] proved that the PIG model of a connected PIG graph is unique up to full reversal, while Huang [Hua95] proved that the PCA model of a connected and co-connected PCA graph is unique up to full reversal. Besides the theoretical interest behind these questions, it turns out that the uniqueness of PIG models is strongly used to solve the recognition problem. In fact, interval graphs can be recognized in $O(1)$ time per edge insertion due to this property (see [DHH96, HSS01]). Because of the strong relationship between PIG and PHCA graphs, it should come at no surprise that there is a unique NPHCA model of every PHCA graph, up to full reversal. Even more, this property can be exploited so as to generalize the PIG recognition algorithms to the PHCA case, with not much effort (see Chapter 6).

Before counting the number of NPHCA models, we take a little detour to prove a property of PCA models. Recall that every XCA graph admits a twin-consecutive XCA model. It turns out that PCA graphs admit only twin-consecutive models, except for the universal arcs. This is useful for non-interval PHCA models, because they have no universal arcs.

Theorem 3.3.1. *Every PCA model with at most one universal arc is twin-consecutive.*

Proof. Let A_1 and A_2 be a pair of twin arcs of a PCA model $\mathcal{M}$ and assume, w.l.o.g., that $s(A_1), s(A_2), t(A_1)$, and $t(A_2)$ appear in this order in a traversal of $C(\mathcal{M})$. To obtain a contradiction, suppose that $s(A_1)$ and $s(A_2)$ are not in the same s-sequence. So, there is some ending point t, of some arc $A = (s, t)$, between $s(A_1)$ and $s(A_2)$. Since A_1 is not universal then, by Lemma 2.4.7, it follows that A_1 and A do not cover the circle. Also, since $\mathcal{M}$ is proper then $s, s(A_i), t$ must appear in this order in a traversal of $C(\mathcal{M})$. Finally, since A_1 and A_2 are twins, then $A_2 \cap A \neq \emptyset$. Summing up, $s(A_1), t, s(A_2), t(A_1), s, t(A_2)$ must appear in this order in $C(\mathcal{M})$. Since A_1 is not universal and A_1 and A_2 are twins, then there is some arc whose both extremes lie in $(t(A_2), s(A_1))$. But this is a contradiction to the fact that $\mathcal{M}$ is proper, because this arc is contained in A. Therefore, $s(A_1)$ and $s(A_2)$ lie in the same s-sequence. Analogously, $t(A_1)$ and $t(A_2)$ lie in the same t-sequence.

Now, suppose that there is some beginning point s of an arc $A = (s, t)$ in the segment $(s(A_1), s(A_2))$. Then, t must belong to the segment $(t(A_1), t(A_2))$ since otherwise A is either contained in A_1 or containing A_2. In other words, A is a twin of both A_1 and A_2. Analogously, if the ending point of some arc lies between $t(A_1)$ and $t(A_2)$ then its beginning point lies between $s(A_1)$ and $s(A_2)$. Hence, every maximal set of twin arcs is consecutive in $\mathcal{M}$, *i.e.*, $\mathcal{M}$ is twin-consecutive. □

We record the results by Roberts and Huang for easy of reference.

Theorem 3.3.2 ([Rob69])**.** *Every connected PIG graph admits at most two PIG models, one the reverse of the other.*

Theorem 3.3.3 ([Hua95])**.** *Every connected PCA graph whose complement is either connected or non-bipartite admits at most two PCA models, one the reverse of the other.*

We are now ready to prove the uniqueness of normal PHCA models.

Theorem 3.3.4. *Every connected PHCA graph admits at most two NPHCA models, one the reverse of the other.*

Proof. The proof is by induction on the number of twin vertices. Suppose, for the base case, that G contains no pair of twin vertices. If $\overline{G}$ is connected or it is non-bipartite then the result follows from Theorem 3.3.3. Then, it is enough to deal with the case in which $\overline{G}$ has $k > 1$ components $\overline{H_1}, \ldots, \overline{H_k}$, all of which are bipartite. We denote by H_i the subgraph of G induced by the vertices of $\overline{H_i}$, for $1 \leq i \leq k$. Since G has no twin vertices and it is W_4-free by Corollary 3.2.11, then it follows that $k = 2$. Analyze the following two cases:

Case 1: $|H_1| \geq |H_2| > 1$. If H_1 contains an induced path of four vertices v_1, v_2, v_3, v_4 then v_1, v_2, v_3 together with a pair of non-adjacent vertices of H_2 induce a W_4 in G, a contradiction to Corollary 3.2.11. Then, $\overline{H_1}$ is bipartite and H_1 contains no twins and no paths of four vertices. Thus H_1 if isomorphic to $\overline{P_2}$. Then, by the case hypothesis, H_2 is also isomorphic to $\overline{P_2}$. Consequently, G is isomorphic to C_4, which admits a unique circular-arc model.

Case 2: $|H_2| = 1$. In this case G contains no hole, or otherwise the hole and the vertex of H_2 would induce a wheel. Let $\mathcal{M}$ be an NPHCA model of G. Since G contains no hole, it follows some point of $C(\mathcal{M})$ is not covered by the arcs of $\mathcal{M}$. In other words, every NPHCA model of G is a PIG model. Hence, by Theorem 3.3.2, G admits two NPHCA models, one the reverse of the other.

For the inductive case, observe that either G contains no universal vertices or every NPHCA model of G is PIG as in Case 2 above. In the former case, every NPHCA model of G is twin-consecutive by Theorem 3.3.1, thus there is a unique model by the inductive hypothesis. In the latter case, G admits a unique PIG model by Theorem 3.3.2. □

3.3.2 Circular-ones properties of the clique matrix

For the second part of this section, we study the relationship between PHCA graphs and the circular-ones properties of the clique matrix. A 0-1 matrix M has the *consecutive-ones property for rows* if its columns can be ordered so that, in every row, the ones are consecutive. Matrix M has the *circular-ones property for rows* if the columns can be ordered so that, in every row, either the zeros or the ones are consecutive. The consecutive and circular-ones properties for columns are defined analogously. That is, M has the *consecutive-ones property for columns* if M^T has the consecutive-ones property for rows, while M has the *circular-ones property for columns* if M^T has the circular-ones property for rows. Here M^T is the transpose matrix of M.

Let $C_1, \ldots, C_k$ be the cliques of a graph G and $v_1, \ldots, v_n$ be its vertices. The *clique-vertex incidence matrix* of G, or simply the *clique matrix* of G, is the 0-1 matrix $Q(G)$ with k rows and n columns where $Q(G)_{i,j} = 1$ if and only if vertex v_j belongs to clique C_i, for every $1 \leq i \leq k$, $1 \leq j \leq n$.

We mentioned in Section 3.1 that $Q(G)$ has the consecutive-ones (resp. circular-ones) property for columns if and only if G is an interval graph (resp. HCA graph). The stronger condition of $Q(G)$ having also the consecutive-ones property for rows is equivalent to the condition of G being a proper interval graph. We prove the analogous result for the circular-ones property, *i.e.*, $Q(G)$ has the circular-ones property for both rows and columns if and only if G is a PHCA graph.

Theorem 3.3.5. *A graph G is a PHCA graph if and only if $Q(G)$ has the circular-ones property for both rows and columns.*

Proof. Let $\mathcal{M}$ be a PHCA model of G and let $A_1, \ldots, A_n$ be the arcs of $\mathcal{M}$ in order of appearance of its beginning points. Since $\mathcal{M}$ is HCA then each clique is represented by some clique point. Let $p_1, \ldots, p_k$ be the clique points in circular order. Define Q as the $k \times n$ matrix where $Q_{i,j} = 1$ if A_j crosses p_i, and 0 otherwise. By definition, Q is a clique matrix of G. Since we used the same construction as Gavril did in [Gav74], it follows that Q has the circular-ones property for columns. We now show that Q has also the circular-ones property for rows. Let r be some row of Q and represent by r_i the i-th column of r. Suppose that the ones in r are not all consecutive. Then, there exist a, b, c such that $r_a = r_c = 1$, $r_b = 0$ and $1 \leq a < b < c \leq n$. In other words, the clique point p_i is crossed by A_a and A_c, but not by A_b. Since $s(A_a), s(A_b), s(A_c)$ appear in this order in $\mathcal{M}$ and $\mathcal{M}$ is proper, then $s(A_a), p_i, t(A_c), s(A_b), t(A_b)$ appear in this order in $\mathcal{M}$. Even more, since $\mathcal{M}$ is proper, it follows that A_d crosses p_i for every d such that $1 \leq d \leq a$ or $c \leq d \leq n$. Thus, every zero in r is consecutive, and so Q has the circular-ones property for rows.

For the converse, we show that if G is not a PHCA graph then $Q(G)$ does not have the circular-ones property for either the rows or the columns. If G is not an HCA graph, then

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Figure 3.9: From left to right, the clique matrices of $K_{1,3}$, W_4, W_5 and S_3 are shown.

$Q(G)$ does not have the circular-ones property for columns [Gav74]. Suppose then that G is HCA and it is not PHCA. Then, by Corollary 3.2.4 and Theorem 3.1.2, G contains a $K_{1,3}$, a W_4, a W_5, or an S_3 as an induced subgraph. Clique matrices for these graphs are depicted in Figure 3.9. By inspection, none of these matrices has the circular-ones property for rows. Thus, as $Q(G)$ contains at least one (permutation) of these matrices as a submatrix, it follows that $Q(G)$ does not have the circular-ones property for rows. □

3.3.3 Graph orientations and vertex enumerations

It is easy to see that in any NHCA model $\mathcal{M}$, the submodel induced by the closed neighborhood of some arc is in fact an interval model. Hence, if G is an NHCA graph, then $N[v]$ is an interval graph for every $v \in V(G)$. The converse does not hold; the umbrella is not an NHCA graph, but the closed neighborhood of each of its vertices is an interval graph. Nevertheless, it is possible to give a necessary and sufficient condition that reflects the fact that NHCA graphs are "locally interval". This "locally interval" property is based on interval vertex orders, interval enumerations and out-straight orientations. We begin by introducing all these notions (see also [BJG01]).

An *interval vertex order* of a graph G is a linear ordering $v_1, \ldots, v_n$ of $V(G)$ such that, for every $1 \leq i \leq n$, v_i is adjacent to $v_{i+1}, \ldots, v_j$ and is not adjacent to $v_{j+1}, \ldots, v_n$, for some $i \leq j \leq n$. As it was proved by Olariu in [Ola91], a graph is an interval graph if and only if it admits an interval vertex order. Interval vertex orders can also be defined in terms of vertex enumerations and graph orientations. An oriented graph D is *out-straight* if there is a linear ordering $v_1, \ldots, v_n$ of $V(D)$ such that, for every vertex v_i, $N^+[v_i] = \textsc{Lin}[v_i, v_{i+r}]$, where $r = d^+(v_i)$. The linear ordering $v_1, \ldots, v_n$ is referred to as an *out-straight enumeration* of D. Note that there is a one-to-one mapping between graphs with interval vertex orders and out-straight oriented graphs. Indeed, if G is a graph with an interval vertex order $v_1, \ldots, v_n$, then the digraph D that is obtained by orienting each edge so that $v_i \longrightarrow v_j$ if and only if $v_iv_j \in E(G)$ and $i < j$ is an out-straight orientation of G (see Figure 3.10). Moreover, $v_1, \ldots, v_n$ is an out-straight enumeration of D.

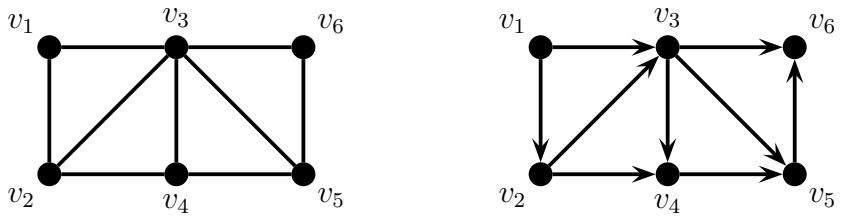


Figure 3.10: An interval graph and its corresponding out-straight orientation. The ordering $v_1, \ldots, v_6$ is an interval vertex order of the graph and an out-straight enumeration its orientation.

The most common way to generalize the definition of out-straight enumerations into circular orderings is to exchange the linear ordering $v_1, \ldots, v_n$ with a circular ordering. That is, an oriented graph D is *out-round* if there is a circular ordering $v_1, \ldots, v_n$ of $V(D)$ such that, for every vertex v_i, $N^+[v_i] = [v_i, v_{i+r}]$ where $r = d^+(v_i)$. The circular ordering $v_1, \ldots, v_n$ is referred to as an *out-round enumeration* of D. Figure 3.11 shows three examples of out-round oriented graphs.

Although out-round oriented graphs are a natural generalization of out-straight oriented graphs, there is a key property about the scopes of the vertices that is completely lost. Let $\phi = v_1, \ldots, v_n$ be an ordering (either linear or circular) of $V(D)$ for an oriented graph D. For $v_i \in V(D)$, define the *leftmost* neighbor of v_i in ϕ as the vertex $v_l \in N^-[v]$ that appears last when traversing ϕ from v_i in reverse order. Similarly, define the *rightmost* neighbor of v_i in ϕ as the vertex $v_r \in N^+[v]$ that appears last when traversing ϕ from v_i in forward order. The *scope* of v_i in ϕ is the range $[v_l, v_r]$ where v_l and v_r are the leftmost and rightmost neighbors of v_i in ϕ (see Figure 3.11 (c)). (Recall that ranges are linear orderings by definition.) In other words, the scope of v_i is the unique range $S = [v_l, v_r]$ such that $N[v_i] \subseteq S$, $v_l \in N^-[v_i]$ and $v_r \in N^+[v_i]$ (if such a range exists).

It is easy to see that if $\phi = v_1, \ldots, v_n$ is an out-straight enumeration of an oriented graph D, then the scope S of the vertex v_i is an out-straight enumeration of $D[S]$, for every $v_i \in V(D)$. This property does not hold for out-round oriented graphs. The oriented 4-wheel graph D in Figure 3.11 (a) is out-round, but the scope S of its universal vertex contains an induced hole and, therefore, $D[S]$ is not out-straight. We define the locally out-straight oriented graphs specifically to restore this property back. That is, an oriented graph D is *locally out-straight* if there is an out-round enumeration $\phi = v_1, \ldots, v_n$ of D such that S is an out-straight enumeration of $D[S]$, for every $v_i \in V(D)$ with scope S in ϕ. As before, the enumeration ϕ is referred to as a *locally out-straight enumeration* of D. The graph depicted in Figure 3.11 (b) is locally out-straight and it is not out-straight.

The next two theorems relate the out-round and the locally out-straight oriented graphs with the NCA and NHCA graphs, respectively.

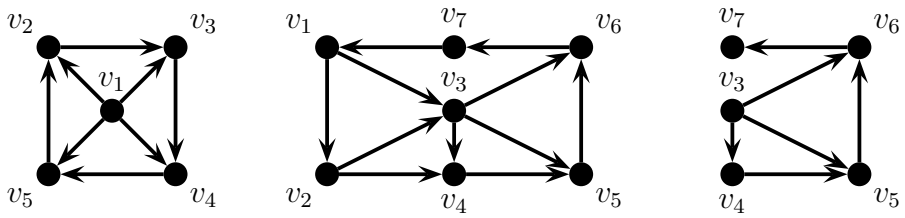


Figure 3.11: Examples of round oriented graphs. An out-round orientation of the 4-wheel graph is depicted in (a). Figure (b) shows an orientation of a non-interval graph together with a locally out-straight enumeration $\phi = v_1, \ldots, v_7$. The scope of v_6 in ϕ appears at (c).

Theorem 3.3.6. *A graph is an NCA graph if and only if it admits an out-round orientation.*

Proof. Let $\mathcal{M}$ be an NCA model of a graph G with arcs $A_1, \ldots, A_n$ where $s(A_1)$, ..., $s(A_n)$ appear in this order in a traversal of $C(\mathcal{M})$. For $1 \leq i \leq n$, call v_i to the vertex of G that corresponds to A_i. Define D as the digraph with vertex set $V(G)$, where $v_i \longrightarrow v_j$ if and only if A_i crosses $s(A_j)$, for every $1 \leq i, j \leq n$. We claim that D is an out-round orientation of G with $\phi = v_1, \ldots, v_n$ as an out-round enumeration. Fix i and j such that $1 \leq i, j \leq n$. First, notice that $s(A_i) \in A_j$ only if $s(A_j) \notin A_i$ because $\mathcal{M}$ is NCA. Thus, $v_i \longrightarrow v_j$ if and only if v_i is adjacent to v_j and $v_j \not\longrightarrow v_i$ which implies that D is an orientation of G. Second, if $v_i \longrightarrow v_j$ then, since $s(A_j) \in A_i$, it follows that $v_i \longrightarrow v_k$ for every $v_k \in (v_i, v_j]$. Hence $N^+[v_i] = [v_i, v_{i+r}]$ where $r = d^+(v_i)$, implying that ϕ is an out-round enumeration of D.

For the converse, let $\phi = v_1, \ldots, v_n$ be an out-round enumeration of some orientation D of G. Pick n points $s(1), \ldots, s(n)$ of a circle C in such a way that $s(1), \ldots, s(n)$ appear in this order in a traversal of C. For each vertex $v_i \in V(D)$, define A_i as the arc of C whose beginning point is $s(i)$ and whose ending point lies in $(s(i + r_i), s(i + r_i + 1))$ where $r_i = d^+(v_i)$. We claim that $\mathcal{M} = (C, \{A_i\}_{1 \leq i \leq n})$ is an NCA model of G. Fix i and j such that $1 \leq i, j \leq n$. By definition, $v_i \longrightarrow v_j$ if and only if $i < j \leq i + r_i$, thus A_i crosses $s(j) = s(A_j)$ if and only if $v_i \longrightarrow v_j$ which implies that $\mathcal{M}$ is a circular-arc model of G. On the other hand, if $v_i \longrightarrow v_j$ then $v_j \not\longrightarrow v_i$, hence if A_i crosses $s(A_j)$ then A_j does not cross $s(A_i)$. That is, $\mathcal{M}$ is an NCA model of G. □

Theorem 3.3.7. *A graph is an NHCA graph if and only if it admits a locally out-straight orientation.*

Proof. Let $\mathcal{M}$ be an NHCA model of a graph G with arcs $A_1, \ldots, A_n$ where $s(A_1)$, ..., $s(A_n)$ appear in this order in a traversal of $C(\mathcal{M})$. Define the set $\{v_i\}_{1 \leq i \leq n}$, the orientation D of G, and the out-straight enumeration ϕ of D as in the proof of Theorem 3.3.6.

We claim that D is actually a locally out-straight orientation of G with ϕ as a locally out-straight enumeration. Fix i such that $1 \leq i \leq n$, and take a small enough ϵ. Let A_l be the arc crossing $s_i + \epsilon$ whose beginning point is farthest from s_i in a counterclockwise traversal of $C(\mathcal{M})$. Similarly, let A_r be the arc crossing $t_i - \epsilon$ whose beginning point is nearest to t_i in a counterclockwise traversal of $C(\mathcal{M})$. By the way that ϕ is defined, it follows that v_l and v_r are the leftmost and rightmost neighbors of v_i in ϕ. Hence, $S = [v_l, v_r]$ is the scope of vertex v_i in ϕ.

Since $s(A_1), \ldots, s(A_n)$ appear in this order in $\mathcal{M}$, it follows that $v_j \in S$ if and only if $s(A_j) \in (s_l - \epsilon, t_i + \epsilon)$. Thus, A_j can not cross s_l, or otherwise A_j together with A_l and A_i would cover the circle. Therefore, the submodel $\mathcal{M}'$ of $\mathcal{M}$ induced by the arcs $A_l, \ldots, A_r$ is an interval model of $G[S]$. This implies that S is an interval vertex order of $G[S]$ and, hence, it is also a locally out-straight enumeration of $D[S]$. Consequently, ϕ is a locally out-straight enumeration of D as claimed.

For the converse, let G be a graph that admits a locally out-straight orientation. By Theorem 3.3.6, G must be an NCA graph. So, it is enough to prove that G contains no wheels, 3-suns, umbrellas, rising suns, nor tents as induced subgraphs, by Corollary 3.2.8. It is not hard to see that the class of graphs that admit a locally out-straight orientation is hereditary, hence we need only to prove that wheels, 3-suns, umbrellas, rising suns, and tents admit no locally out-straight orientations. We shall do this for the proof.

For the first case, let D be an out-round orientation of a wheel and take ϕ as an out-round enumeration of D. Clearly, if v is the universal vertex of the wheel and S is its scope then $N[v] = V(D) \subseteq S$. But then $D[S]$ is not out-straight because it contains a hole, implying that ϕ is not a locally out-straight enumeration. Therefore, D is not locally out-straight.

For the second case, let D be an out-round orientation of the n-rising sun graph with $n \geq 4$. Recall that the n-rising sun is the graph obtained by inserting two universal vertices v_1 and v_n into a path $P = v_2, \ldots, v_{n-1}$, and then inserting three vertices w_1, w_{n-1}, w_n such that w_i is adjacent only to v_i and v_{i+1}, for $i \in \{1, n-1, n\}$ (see Figure 3.12 (a)). Suppose, to obtain a contradiction, that D admits a locally out-straight enumeration ϕ. It is not hard to see that the induced path w_1, P, w_{n-1} admits only the following out-round enumerations, all of which are out-straight:

- $\rho = w_1, v_2, \ldots, v_{n-1}, w_{n-1}$,
- $\gamma = v_2, w_1, v_3, \ldots, v_{n-1}, w_{n-1}$,
- $\rho' = w_{n-1}, v_{n-1}, \ldots, v_2, w_1$, and
- $\gamma' = v_{n-1}, w_{n-1}, v_{n-2}, \ldots, v_2, w_1$.

Each of these enumerations corresponds to one of the four possible out-straight orientations of a path (see Figures 3.12 (b) and (c)). So, one of these enumerations must appear in this order inside ϕ. Furthermore, a vertex z_1 in ϕ' has a directed edge to

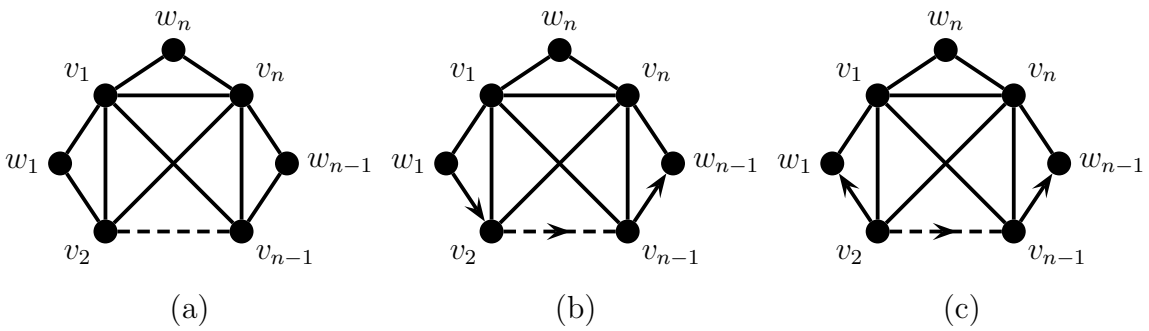


Figure 3.12: The n-rising sun graph and the orientations corresponding to ρ and γ.

another vertex z_2 in ϕ' only if z_1 appears before z_2 in ϕ'. Enumerations ρ and ρ' are symmetric, in the sense that the former is obtained from the latter by exchanging the labels of the vertices. This is also true for enumerations γ and γ', so we need to consider only two cases, either $\phi' = \rho$ or $\phi' = \gamma$. For the sake of simplicity, define $u_1 = w_1$ and $u_2 = v_2$ in ρ, and $u_1 = v_2$ and $u_2 = w_1$ in γ. Thus, $\phi' = u_1, u_2, v_3, \ldots, v_{n-1}, w_{n-1}$. Denote by $l(v)$ and $r(v)$ the leftmost and rightmost neighbors of v in ϕ for every $v \in V(D)$. The following claims analyze the positions of w_n and v_n in ϕ.

Claim 1: $w_n \notin [u_1, w_{n-1}]$ in ϕ. Otherwise, there would be two adjacent vertices z_1 and z_2 such that z_1, z_2 appear in this order in ϕ' and z_1, w_n, z_2 appear in this order in ϕ. But this is impossible, because $z_1 \longrightarrow z_2$ and $z_1 \not\longrightarrow w_n$.

Claim 2: $v_n \in [w_1, w_{n-1}]$ in ϕ. Again, suppose otherwise, so in particular $v_n \notin [w_1, v_3]$ in ϕ. Vertices v_3 and v_n are adjacent, so either $v_n \longrightarrow v_3$ or $v_3 \longrightarrow v_n$. The former is impossible because it implies that $w_1 \in [v_n, r(v_n)]$, contradicting the fact that v_n is not adjacent to w_1. In the latter case, $v_2 \not\longrightarrow v_n$ because $w_{n-1} \notin [v_2, r(v_2)]$. But this is also impossible since $v_3 \in [l(v_n), v_n]$, $v_2 \in [v_n, r(v_n)]$ and $v_2 \longrightarrow v_3$, contradicting the fact that $[l(v_n), r(v_n)]$ is out-straight.

By Claims 1 and 2, w_1, v_n, w_{n-1}, w_n must appear in this order in a traversal of ϕ. So, $v_n \longrightarrow w_n$ in D because $w_1 \notin [w_n, r(w_n)]$ or otherwise ϕ would not be out-round. Hence, v_1, w_{n-1}, w_n can not appear in this order in the range $[l(v_n), r(v_n)]$, because v_1 is adjacent to w_n and not to w_{n-1}. Consequently, v_n, w_{n-1}, v_1 appear in this order in $[l(v_n), r(v_n)]$. Analogously, v_2, w_{n-1}, v_1 can not appear in this order in $[l(v_n), r(v_n)]$, so v_n, w_{n-1}, v_2 must appear in this order in $[l(v_n), r(v_n)]$. Recall that $v_2 \longrightarrow v_3$ in D, thus v_n, w_{n-1}, v_2, v_3 must appear in this order in $[v_n, r(v_n)]$ and, therefore, $v_n \longrightarrow v_3$. But since ϕ' is either ρ or γ, it follows that $v_n, w_{n-1}, u_1, u_2, v_3$ appear in this order in ϕ. This is a contradiction to the fact that ϕ is out-round, because $v_n \longrightarrow v_3$ but $v_n \not\longrightarrow w_1 \in \{u_1, u_2\}$.

Finally, by using backtracking arguments, it can be proved that there are no locally out-straight orientations of the 3-sun, the tent, and the umbrella. □

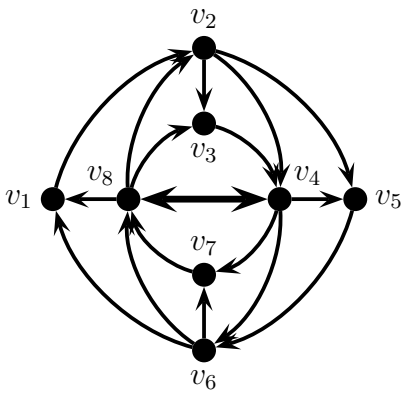


Figure 3.13: Two out-round orientations of a universal-free non-interval NCA graph (the edge between v_4 and v_8 can be oriented both ways). We can see that $\phi = v_1, \ldots, v_8$ is an out-round enumeration of both orientations.

As we already argued, every interval graph G admits an out-straight orientation. We can encode this information with the orientation D and its out-straight enumeration ϕ. But, we can also encode this information only with an enumeration ϕ. This is because if $\phi = v_1, \ldots, v_n$, then $v_i \longrightarrow v_j$ in D if and only if v_i is adjacent to v_j in G and $i < j$. That is, we do not really need to say who is D once ϕ and G are given, *i.e.*, ϕ and G are an implicit representation of D. This is what Olariu did in [Ola91], he encoded D with G and an interval vertex order of G without using any kind of orientations.

This nice encoding of out-straight orientations is lost for out-round digraphs. That is, there are NCA graphs that admit two different out-round orientations that share the same out-round enumeration, as it is depicted in Figure 3.13. So, this is another example of a property that holds for interval graphs and it does not hold for NCA graphs. The property can be restored, by restricting the attention to the non-interval NHCA graphs. We show how to do this in the following proposition, but before we need one definition. For a circular ordering $\phi = v_1, \ldots, v_n$ of the vertices of G, define $\longrightarrow_\phi$ as the relation on $V(G)$ such that $v_i \longrightarrow_\phi v_j$ if and only if v_i is adjacent to all the vertices in $[v_{i+1}, v_j]$ $(1 \leq i, j \leq n)$.

Proposition 3.3.8. *Let D be a locally out-straight orientation of a non-interval graph G, and let $\phi = v_1, \ldots, v_n$ be one of its locally out-straight enumerations. Then, for $1 \leq i, j \leq n$, $v_i \longrightarrow v_j$ in D if and only if $v_i \longrightarrow_\phi v_j$.*

Proof. If $v_i \longrightarrow v_j$ in D and $v_i \not\longrightarrow_\phi v_j$, then there is a vertex $v_h \in [v_i, v_j]$ such that $v_i \not\longrightarrow v_h$ in D. Then, ϕ is not out-round.

For the converse, suppose that $v_i \longrightarrow_\phi v_j$ and yet $v_i \not\longrightarrow v_j$ in D. Let $r = i + d^+(v_i)$. Since ϕ is locally out-straight, then $[v_i, v_r] \subset [v_i, v_j]$. By definition of $\longrightarrow_\phi$, v_i is adjacent

to all the vertices in $[v_{i+1}, v_j]$, thus v_i is adjacent to v_{r+1}. Then, $v_{r+1} \longrightarrow v_i$ in D, because D is an orientation of G and $v_i \not\longrightarrow v_{r+1}$. Let $\mathcal{M}$ be the NCA model that is obtained from D and ϕ as in the proof of Theorem 3.3.6. By Theorem 3.3.7 and Corollary 3.2.5, G is an NHCA graph and $\mathcal{M}$ is an NHCA model of G. Call A_a to the arc of $\mathcal{M}$ that corresponds to the vertex v_a, for every $1 \leq a \leq n$. As in Theorem 3.3.6, the arc A_a crosses $s(A_b)$ if and only if $v_a \longrightarrow v_b$ in D. Then, A_i crosses $s(A_r)$ and A_{r+1} crosses $s(A_i)$ in $\mathcal{M}$. Since $\mathcal{M}$ is NHCA, then it follows that no arc crosses both $t(A_i)$ and $s(A_{r+1})$. Thus, no arc of $\mathcal{M}$ crosses $s(A_{r+1})$ and, therefore, $\mathcal{M}$ is an interval model of G. □

By the above proposition, there is no need to say who is D once a non-interval NHCA graph G and a circular ordering ϕ of $V(G)$ are given. We can think of ϕ as a some sort of locally out-straight enumeration of G, instead of thinking ϕ as a locally out-straight enumeration of an orientation of G. To make this more formal, let ϕ be a circular ordering of $V(G)$. Define the *ϕ-orientation* of G as the digraph D that has the same vertices as G and such that $v_i \longrightarrow v_j$ in D if and only if $v_i \longrightarrow_\phi v_j$. Then, we have the following definition, which is similar to interval vertex orders.

Definition 3.1. *Let ϕ be a circular ordering of $V(G)$ and D be the ϕ-orientation of G. Say that ϕ is an* NHCA order *of G if and only if ϕ is a locally out-straight enumeration of D and D is an orientation of G.*

To see the analogy between interval vertex orders and NHCA orders, it is better to rephrase the definition of interval vertex orders. An interval vertex order is a linear ordering $\phi = v_1, \ldots, v_n$ of $V(G)$ such that ϕ is an out-straight enumeration of the ϕ-orientation of G. The condition saying that the ϕ-orientation is also an orientation of G is not required, because if $v_i \longrightarrow_\phi v_j$ then $v_j \not\longrightarrow_\phi v_i$. Why is this condition required for NHCA orders?

Proposition 3.3.8 guaranties that if G is a non-interval NHCA graph and ϕ is a locally out-straight enumeration of an orientation of G then ϕ is an NHCA order of G. However, this is not true when G is an interval graph. That is, ϕ can be a locally out-straight enumeration of an orientation of G but not an NHCA order of G. The problem is that, in this case, the ϕ-orientation of G could not be an orientation of G at all. This is true even when G is universal-free, as is depicted in Figure 3.14. For this reason is that we ask the ϕ-orientation to be an orientation of G in Definition 3.1.

The problem to relate interval vertex orders with NHCA orders is that the former are linear orderings whereas the latter are circular orderings. So, in an interval vertex order ϕ, we may have a vertex w (as v_5 in Figure 3.14) which is adjacent to all the vertices to the right of it, and it is also adjacent to the leftmost vertex v (v_1 in Figure 3.14). This implies that the interval vertex order is not an NHCA order because $v \longrightarrow_\phi w \longrightarrow_\phi v$. When G is universal-free, there is always a non-neighbor z of w, between v and w, that

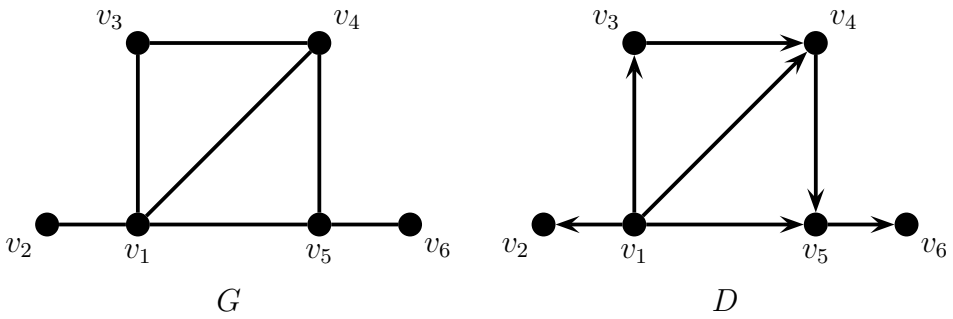


Figure 3.14: The ϕ-orientation of G, for $\phi = v_1, \ldots, v_6$, is not an orientation of G because $v_1 \longrightarrow_\phi v_5$ and $v_5 \longrightarrow_\phi v_1$. However, ϕ is an out-straight enumeration of the orientation D of G.

can be put as the leftmost vertex (as v_2 in Figure 3.14). This vertex is what we call a messy vertex. If we move the messy vertices to the leftmost position, then we will obtain an NHCA order. In the example of Figure 3.14 such enumeration is $v_2, v_1, v_3, v_4, v_5, v_6$. We now formalize this idea.

Let $\phi = v_1, \ldots, v_n$ be an interval vertex order of some graph G. Say that v_i is a *messy* vertex of ϕ if there is a vertex v_j, $j > i$, such that v_1 is adjacent v_j and v_j is not adjacent to v_i. A *messy-free* interval vertex order is an interval vertex order that has no messy vertices.

Proposition 3.3.9. *Every universal-free interval-graph admits a messy-free interval vertex order.*

Proof. Suppose that G is a universal-free interval graph that admits no messy-free interval vertex orders. Take the interval vertex order $\phi = v_1, \ldots, v_n$ of G that has the fewer messy vertices. Let v_i be the messy vertex with the lowest index and define $\phi' = v_i, [v_1, v_{i-1}], [v_{i+1}, v_n]$. By the minimality of i and the fact that v_1 is adjacent to all the vertices in $[v_2, v_i]$, it follows that $[v_1, v_i]$ is a complete set. Hence, for every $1 \leq k \leq i$, $N[v_k] = [v_1, v_{r(k)}]$ where $r(k) = d(v_k) + 1$ which implies that ϕ' is an interval vertex order of G. Again, by minimality of i, it follows that $r(k) > r(i)$ for every $1 \leq k < i$. So, $v_1, \ldots, v_{i-1}$ are not messy vertices of ϕ'. Finally, if v_k is messy in ϕ' then v_k is messy in ϕ because $r(1) > r(i)$. Hence, ϕ' contains at least one messy vertex less than G, a contradiction. □

After moving all the messy vertices to the left as in the proposition above, we have a messy-free interval vertex order. Such an interval vertex order is an NHCA order as it is shown in the next theorem.

Theorem 3.3.10. *If $\phi' = v_1, \ldots, v_n$ is a messy-free interval vertex order of a universal-free graph G, then the circular ordering $\phi = v_1, \ldots, v_n$ is an NHCA order of G.*

Proof. Suppose, to obtain a contradiction, that $v_i \longrightarrow_\phi v_j$ for some $1 \leq j < i \leq n$. Then, by the definition of $\longrightarrow_\phi$, it follows that v_i is adjacent to all the vertices in $[v_{i+1}, v_j]$. In particular, v_i is adjacent to v_1 in G and, since ϕ' is an interval vertex order, it follows that v_1 is adjacent to all the vertices in $[v_2, v_i]$. Therefore, since there are no messy vertices in ϕ', we obtain that v_k is adjacent to v_i in G, for every $1 \leq k < i$. But this implies that v_i is a universal vertex, a contradiction.

Suppose now that $v_i \not\longrightarrow_\phi v_j$ for $1 \leq j < i \leq n$. Then, the ϕ-orientation D is in fact an out-straight digraph that has ϕ' as an out-straight enumeration. That is, D is an orientation of G and ϕ is a locally out-straight enumeration of D. □

The main theorem about NHCA orders is the following.

Theorem 3.3.11. *A universal-free graph is NHCA if and only if it admits an NHCA order.*

Proof. If G is a non-interval graph then it admits an NHCA order by Proposition 3.3.8 (cf. above). Otherwise, G admits a messy-free interval vertex order by Proposition 3.3.9, and the circular version of this order is an NHCA order of G by Theorem 3.3.10.

The converse of the theorem follows from the fact that graphs admitting an NHCA order have an orientation as a locally out-round digraph, hence, by Theorem 3.3.7, these graphs are NHCA. □

Now that we are done with the NHCA orders, consider once again the universal-free non-interval graph G in Figure 3.13. It is not hard to see that G admits exactly two isomorphic out-round orientations, both depicted in Figure 3.13. The unique out-round enumeration ϕ these orientations is $v_1, \ldots, v_8$. It turns out that the ϕ-orientation of G is not an orientation of G. Thus, as we already argued informally, ϕ and G are not enough to encode the orientation D; even if ϕ and G are given, we still have to say who is the rightmost neighbor of every vertex v of G.

Up to this point we have described interval, NHCA, and NCA graphs in terms of orientations and enumerations. We showed that for interval graphs and universal-free NHCA graphs one can get rid of the orientation by using interval vertex orders and NHCA orders, respectively. For interval graphs this is a natural thing because there is no need of the orientation to define interval vertex orders. For universal-free NHCA graphs, this is somehow artificial because the ϕ-orientation is not always an orientation. Even worst, for some universal-free NCA graphs this is impossible. It is also interesting to note what happens when interval vertex orders are generalized into circular orderings without orienting the graph. Say that a circular ordering $\phi = v_1, \ldots, v_n$ of $V(G)$ is a

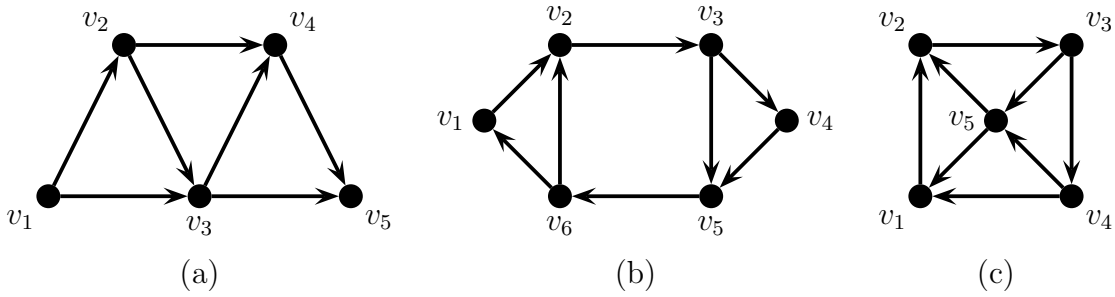


Figure 3.15: Examples of straight (a), locally straight (b) and round (c) oriented graphs.

circular-arc ordering whenever, for $1 \leq i, j \leq n$, if v_i is adjacent to v_j then v_i is adjacent to $v_{i+1}, \ldots, v_j$ or v_j is adjacent to $v_{j+1}, \ldots, v_i$. In this case, a graph G admits a circular-arc ordering if and only if G is a circular-arc graph [Tuc74].

PIG, PCA and PHCA graphs can also be described in terms of orders, orientations and enumerations. The analogous of an interval vertex order is called a PIG order. A *PIG order* of a graph G is a linear ordering $\phi = v_1, \ldots, v_n$ of $V(G)$ such that, for every $1 \leq i \leq n$, there exist two non-negative values l, r such that $N[v_i] = \textsc{Lin}[v_{i-l}, v_{i+r}]$ and both $\textsc{Lin}[v_{i-l}, v_i]$ and $\textsc{Lin}[v_i, v_{i+r}]$ are complete sets. Looges and Olariu [LO93], and Huang [Hua92] independently proved that G is a PIG graph if and only if it admits a PIG order. As before, this condition can be described in terms of orientations and enumerations. An oriented graph D is *straight* if there is a linear ordering $v_1, \ldots, v_n$ of $V(D)$ such that, for every vertex v_i, $N^-[v_i] = \textsc{Lin}[v_{i-l}, v_i]$ and $N^+[v_i] = \textsc{Lin}[v_i, v_{i+r}]$, where $l = d^-(v_i)$ and $r = d^+(v_i)$ (see Figure 3.15 (a)). As usual, the ordering $v_1, \ldots, v_n$ is called a *straight enumeration* of D. Deng et al. [DHH96] (see also [Hua92]) proved that G is a PIG graph if and only if it admits an straight orientation. Furthermore, ϕ is a PIG order of G if and only if ϕ is a straight enumeration of the ϕ-orientation of G.

As before, straight oriented graphs can be generalized either as locally straight oriented graphs or as round oriented graphs. An oriented graph D is *round* if there is a circular ordering $v_1, \ldots, v_n$ of $V(D)$ such that $N^-[v_i] = [v_{i-l}, v_i]$ and $N^+[v_i] = [v_i, v_{i+r}]$, for every vertex v_i with $l = d^-(v_i)$ and $r = d^+(v_i)$. When the scope S of each vertex is a straight enumeration of $D[S]$, then D is also a *locally straight* oriented graph. As before, the circular orderings corresponding to the round and locally straight oriented graphs are called *round enumerations* and *locally straight enumerations*, respectively. Examples of straight, locally straight and round oriented graphs are depicted in Figure 3.15. Hell and Huang [HH95] (see also [Skr82]) proved that the class of PCA graphs is exactly the class of graphs that admit a round orientation. As with NCA graphs, we can restore the Helly condition of PIG graphs by restricting the attention to locally straight graphs.

Theorem 3.3.12. *A graph is a PHCA graph if and only if it admits a locally straight orientation.*

Proof. Let $\mathcal{M}$ be an NPHCA model of a graph G and define D and $\phi = v_1, \ldots, v_n$ as in the proof of Theorem 3.3.7. By Theorem 3.3.7, the scope S of each vertex v_i in ϕ is an out-straight enumeration of $D[S]$, for every $1 \leq i \leq n$. Also, as in the proof of Theorem 3.3.7, the submodel $\mathcal{M}'$ of $\mathcal{M}$ induced by the vertices of S is an interval model of $G[S]$. Since $\mathcal{M}'$ is also PCA then $\mathcal{M}'$ is a PIG model and so S is in fact a PIG order of $G[S]$. Therefore, S is a straight enumeration of $D[S]$ as well.

For the converse, let G be a locally straight orientable graph. Since locally straight oriented graphs are locally out-straight, then G is an NHCA graph by Theorem 3.3.7. Thus, it is enough to see that G contains no induced $K_{1,3}$ by Theorem 3.1.2. But, since the class of locally straight orientable graphs is hereditary, it suffices to prove that no orientation of $K_{1,3}$ is locally straight. This is clearly true because the $K_{1,3}$ graph admits no round orientation. □

A *directed-triangle* of a digraph D is a triple $v_1, v_2, v_3 \in D$ such that $v_1 \longrightarrow v_2$, $v_2 \longrightarrow v_3$, and $v_3 \longrightarrow v_1$. Locally straight orientations can be characterized in terms of directed triangles. The following theorem is a corollary of Theorem 3.3.12, and it will be taken as an alternative definition of locally straight orientations, without further references.

Theorem 3.3.13. *A round oriented graph is locally straight if and only if it contains no directed-triangle.*

To end this chapter, we discuss the implicit encodings of round and locally straight oriented graphs. As for NHCA graphs, the orientation D of G can be implicitly encoded with a round enumeration of D, when G is a universal-free PCA graph. In fact, D is isomorphic to the ϕ-orientation of G in this case, as in Proposition 3.3.8. The proof is a simplification of the one in Proposition 3.3.8.

Proposition 3.3.14. *Let D be a round orientation of a universal-free graph G, and let $\phi = v_1, \ldots, v_n$ be one of its round enumerations. Then, for $1 \leq i, j \leq n$, $v_i \longrightarrow v_j$ in D if and only if $v_i \longrightarrow_\phi v_j$.*

Proof. If $v_i \longrightarrow v_j$ in D and $v_i \not\longrightarrow_\phi v_j$ then there is a vertex $v_h \in [v_i, v_j]$ such that $v_i \not\longrightarrow v_h$ in D. But then ϕ is not out-round.

For the converse, suppose that $v_i \longrightarrow_\phi v_j$ and yet $v_i \not\longrightarrow v_j$ in D. Let $r = i + d^+(v_i)$. Since ϕ is round, then $[v_i, v_r] \subset [v_i, v_j]$. By the definition of $\longrightarrow_\phi$, v_i is adjacent to all the vertices in $[v_{i+1}, v_j]$, thus v_i is adjacent to v_{r+1}. Then, $v_{r+1} \longrightarrow v_i$ in D, because D is an orientation of G and $v_i \not\longrightarrow v_{r+1}$. But then, since ϕ is round, it follows that $v_k \longrightarrow v_i$ for every $v_k \in [v_{r+1}, v_{i-1}]$, *i.e.*, v_i is universal. □

This proposition can be used to define PCA orders in a similar way as NHCA orders. However, observe that in this case if $v_i \longrightarrow_\phi v_j$ then $v_j \not\longrightarrow_\phi v_i$. Thus, we need not to ask for the ϕ-orientation to be an orientation of G. In fact, we do not have to use an orientation at all. Let $\phi = v_1, \ldots, v_n$ be a circular order of G. For every $1 \leq i \leq n$, denote by $r(i)$ the maximum value such that v_i is adjacent or equal to all the vertices of $[v_i, v_{i+r(i)}]$. Similarly, denote by $l(i)$ the maximum value such that v_i is adjacent or equal to all the vertices of $[v_{i-l(i)}, v_i]$. In other words, $v_{i-l(i)}$ and $v_{i+r(i)}$ are the leftmost and rightmost neighbors of v_i in the ϕ-orientation. PCA and PHCA orders are defined as follows.

Definition 3.2. *Let $\phi = v_1, \ldots, v_n$ be a circular ordering of $V(G)$. Say that ϕ is a* PCA order *of G if and only if $N[v_i] = [v_{i-l(i)}, v_{i+r(i)}]$ and both $[v_{i-l(i)}, v_i]$ and $[v_i, v_{i+r(i)}]$ are complete sets, for every $v_i \in \phi$.*

Definition 3.3. *Let $\phi = v_1, \ldots, v_n$ be a circular ordering of $V(G)$. Say that ϕ is a* PHCA order *of G if and only if $N[v_i] = [v_{i-l(i)}, v_{i+r(i)}]$ and $[v_{i-l(i)}, v_{i+r(i)}]$ is a PIG order of the subgraph of G induced by $N[v_i]$, for every $v_i \in \phi$.*

The analogy between PCA orders and NHCA orders is explicit in the following proposition.

Proposition 3.3.15. *Let G be a universal-free PCA graph, ϕ be a circular ordering of $V(G)$ and D be the ϕ-orientation of G. Then ϕ is a PCA order of G if and only if ϕ is a round enumeration of D and D is an orientation of G. Furthermore, ϕ is a PHCA order if and only if ϕ is a locally straight enumeration of D.*

Proof. Let $\phi = v_1, \ldots, v_n$ be a PCA order of G and suppose that D is not an orientation of G. This means that either G is not isomorphic to underlying graph H of D, or that there are two vertices v_i and v_j such that both $v_i \longrightarrow v_j$ and $v_j \longrightarrow v_i$ are edges of D.

First we show that G is isomorphic to H. For this, fix two values i and j such that $1 \leq i, j \leq n$. By definition, if $v_i \longrightarrow_\phi v_j$ then v_i and v_j must be adjacent in G, thus H is a subgraph of G. On the other hand, for ϕ to be a PCA order, if v_i and v_j are adjacent, then either $v_j \in (v_i, v_{i+r(i)}]$ or $v_j \in [v_{i-l(j)}, v_i)$. In the former case $v_i \longrightarrow_\phi v_j$, while in the latter case $v_j \longrightarrow_\phi v_i$ because $[v_{i-l(i)}, v_i]$ is a complete set and thus $v_i \in (v_j, v_{j+r(j)}]$. Hence G is isomorphic to H.

Now, suppose that there are two vertices v_i and v_j such that $v_i \longrightarrow_\phi v_j$ and $v_j \longrightarrow_\phi v_i$. By definition of $\longrightarrow_\phi$, v_i is adjacent to all the vertices of $[v_{i+1}, v_j]$ and v_j is adjacent to all the vertices of $[v_{j+1}, v_i]$. Now, since ϕ is a PCA order, the latter implies that $[v_j, v_i]$ is a complete set, thus v_i is universal, a contradiction.

For the converse, let ϕ be a round enumeration of D, and D be an orientation of G. Fix $i \in \{1, n\}$ and let $l = d^-(v_i)$ and $r = d^+(v_i)$. It is easy to see that $N(v_i) = [v_{i-l}, v_{i+r}]$, and that $[v_{i-l}, v_i]$ and $[v_i, v_{i+r}]$ are complete sets.

The furthermore part is trivial. □

The characterizations of PCA and PHCA graphs by PCA and PHCA orders are obtained as corollaries.

Theorem 3.3.16. *A universal-free graph is a PCA graph if and only if it admits a PCA order.*

Proof. A similar proof to the one of Theorem 3.3.11 shows that a universal-free graph is a PCA graph if and only if there is a circular enumeration ϕ which is a round enumeration of its ϕ-orientation D, while D is an orientation of G. Then, apply Proposition 3.3.15. □

Theorem 3.3.17. *A universal-free graph is PHCA if and only if it admits a PHCA order.*

4 Powers of paths and cycles

In 1988, Golumbic and Hammer [GH88] proposed a linear-time algorithm for the maximum independent set problem restricted to circular-arc graphs. Their algorithm is based on a simple rule called the neighborhood reduction. Basically, they propose to remove each dominator vertex of the graph, until no more dominator vertices remain. The main observation is that the graph so obtained is isomorphic to a power of a cycle.

There are other nice connections between powers of cycles and circular-arc graphs. For instance, Bonomo [Bon06] proved that an HCA graph is self-clique if and only if it is isomorphic to C_n^k for some pair of values n, k such that $n > 3k$. In fact, a general circular-arc graph is K-convergent if and only if it converges to C_n^k for $n > 3k$ (cf. Chapter 8). Powers of cycles are also considered in some different contexts, for example in coloring problems ([CdM07, EK03, KN04, LL07, Tho05]).

Induced subgraphs of powers of cycles have also been examined. Bondy and Locke [BL92] have described bounds for the number of edges in a triangle-free induced subgraph of a power of a cycle. Bermond and Peyrat [BP89] have determined a lower bound for the number of vertices of an induced subgraph G of C_n^k, with the restriction that $\delta(G) \geq k+l$, for $k, l > 0$.

The aim of this chapter is to give more connections between powers of cycles and circular-arc graphs. In Section 4.1 we make explicit all characterizations of powers of cycles implicit in [GH88], and we add some more equivalences. A similar series of equivalent characterizations of powers of paths are given later in the same section. Section 4.2 is devoted to induced subgraphs of powers of cycles and paths. We prove that the former form precisely the class of UCA graphs while the latter form the class of UIG graphs. In Section 4.3 we present a new constructive proof of Roberts' "proper = unit" Theorem. Our proof shows how to obtain a power of a path supergraph of the input PIG graph G, which is then used to obtain a UIG model of G. Furthermore, we show that the extremes of the output model are natural numbers of polynomial value with respect to the size of G, improving over some of the previously known characterization. Finally, we make some further remarks in Section 9.

Sections 4.1 and 4.2 are part of a joint work with Dieter Rautenbach.

4.1 Powers of cycles and paths

Our first result collects several equivalent descriptions of powers of cycles. Theorem 1 in [GH88] actually only states that a circular arc graph without dominators is a power of a cycle. Nevertheless, the given arguments imply the following equivalences from our Theorem 4.1.1 below:

$$(i) \iff (v) \iff (vii) \iff (ix)$$

Theorem 4.1.1. *For a graph G of order n which is not complete, the following statements are equivalent.*

(i) G *is isomorphic to* C_n^k *for some integer* k.

(ii) G *is a regular UCA graph with no twins.*

(iii) G *is a regular PCA graph with no twins.*

(iv) G *is a UCA graph without dominators.*

(v) G *is a PCA graph without dominators.*

(vi) G *is a UCA graph and in every UCA model of* G *the beginning and ending points alternate.*

(vii) G *is a PCA graph and in every PCA model of* G *the beginning and ending points alternate.*

(viii) G *is a UCA graph and in some UCA model of* G *the beginning and ending points alternate.*

(ix) G *is a PCA graph and in some PCA model of* G *the beginning and ending points alternate.*

Proof. The implications $(ii) \Longrightarrow (iii)$, $(vi) \Longrightarrow (viii)$, $(vii) \Longrightarrow (ix)$ and $(viii) \Longrightarrow (ix)$ are trivial.

$(i) \Longrightarrow (ii)$. Clearly, C_n^k is regular and has no pair of twins. If $s_1, s_2, \ldots, s_n$ are n equally spaced points on a circle C and $\mathcal{A}$ is a set which contains the n open arcs of equal length with beginning point s_i and ending point between s_{i+k} and s_{i+k+1} for $1 \leq i \leq n$, then $(C, \mathcal{A})$ is a UCA model of C_n^k.

$(ii) \Longrightarrow (iv)$ (and resp. $(iii) \Longrightarrow (v)$). If $N[u] \subseteq N[v]$, then the regularity of G implies $N[u] = N[v]$ and the twin-freeness of G implies $u = v$. Hence G is a UCA (resp. PCA) graph without dominators.

$(iv) \Rightarrow (vi)$ (and resp. $(v) \Rightarrow (vii)$). If $s(A_1)$ and $s(A_2)$ are two consecutive extreme points of a UCA (resp. PCA) model of G, where A_1 and A_2 are the arcs corresponding to the vertices v_1 and v_2 of G, then every arc of the model which intersects A_1 also intersects A_2 and, so, $N[v_1] \subseteq N[v_2]$.

$(ix) \Rightarrow (i)$. Let $s_0, t_0, s_1, t_1, \ldots, s_{n-1}, t_{n-1}$ be the cyclically consecutive extreme points of a PCA model $\mathcal{M}$ of G as in (ix), where s_i is a beginning point and t_i is an ending point for $1 \leq i \leq n$. It suffices to prove the existence of some $k \in \mathbb{N}$ such that $\mathcal{A}(\mathcal{M})$ consists of the arcs with beginning point s_i and ending point t_{i+k} for $0 \leq i \leq n-1$.

For contradiction, we may assume that the arc A_0 beginning with s_0 ends with t_k and that the arc A_1 beginning with s_1 ends with t_{k+i} for some k with $i \neq 1$. Since $\mathcal{M}$ is proper, $A_1 \not\subset A_0$, thus $i \geq 2$. Let the arc A_j ending with t_{k+1} begin with s_j for some j. Again, since the model is proper, $A_j \not\subset A_1$, thus $j < 1$. Similarly, $A_j \not\supset A_0$, hence $j > 0$. We obtain the contradiction that the integer j satisfies $0 < j < 1$.

In view of the following diagram of the implications, the proof is complete.

$$\begin{array}{ccccccccccc} (i) & \Longrightarrow & (ii) & \Longrightarrow & (iv) & \Longrightarrow & (vi) & \Longrightarrow & (viii) & & \\ & & \Downarrow & & & & & & \Downarrow & & \\ & & (iii) & \Longrightarrow & (v) & \Longrightarrow & (vii) & \Longrightarrow & (ix) & \Longrightarrow & (i) \end{array}$$

□

Our next result collects several equivalent descriptions of powers of paths. Before we can state it, we need some further definitions.

Let $\mathcal{I} = \{(s_i, t_i) \mid 1 \leq i \leq n\}$ be a PIG model for a connected graph G with vertex set $\{v_1, v_2, \ldots, v_n\}$ such that

$$(s_i, t_i) \text{ corresponds to } v_i \text{ for } 1 \leq i \leq n \tag{4.1}$$

and

$$s_1 < s_2 < \ldots < s_n. \tag{4.2}$$

The ordering $\phi = v_1, v_2, \ldots, v_n$ is a PIG order of G (cf. Chapter 3). That is, for every $1 \leq i \leq n$, there exist two non-negative values l, r such that $N[v_i] = \textsc{Lin}[v_{i-l}, v_{i+r}]$, and both $\textsc{Lin}[v_{i-l}, v_i]$ and $\textsc{Lin}[v_i, v_{i+r}]$ are complete sets. As noted by Roberts [Rob69], the ordering ϕ is unique up to permutation of twins and up to full reversal (see Theorem 3.3.2).

If

$$\begin{aligned} p &= \max\{i \mid 1 \leq i \leq n,\ v_i \in N_G[v_1]\} = d(v_1) + 1 \text{ and} \\ q &= \min\{i \mid 1 \leq i \leq n,\ v_i \in N_G[v_n]\} = n - (d(v_n) + 1), \end{aligned} \tag{4.3}$$

then

$$\{v_i \mid \min\{p, q\} \leq i \leq \max\{p, q\}\}$$

is the set of *middle* vertices. By Roberts' result [Rob69], this set does not depend on the PIG order. Note that the vertices v_i with $q \leq i \leq p$ are exactly the universal vertices of G.

Theorem 4.1.2. *For a connected graph G of order n which is not complete, the following statements are equivalent.*

(i) *G is isomorphic to P_n^k for some integer k.*

(ii) *G is a UIG graph in which all twins are universal and whose middle vertices have the same degree.*

(iii) *G is a UIG graph in which all twins are universal. Furthermore, if $v_1, v_2, \ldots, v_n$ is a PIG order and p and q are as in (4.3) then the only dominator sequences of G are*

$$v_r, v_{\min\{p,q\}-1}, v_{\min\{p,q\}-2}, \ldots, v_1$$

with $\min\{p,q\} \leq r \leq p$ and

$$v_s, v_{\max\{p,q\}+1}, v_{\max\{p,q\}+2}, \ldots, v_n$$

with $q \leq s \leq \max\{p,q\}$.

(iv) *G is a UIG graph and for all PIG models $\mathcal{I} = \{(s_i, t_i) \mid 1 \leq i \leq n\}$ with (4.1) and (4.2), and p and q as in (4.3), the beginning and ending points between s_p and t_q alternate.*

(v) *G is a UIG graph and for some UIG model $\mathcal{I} = \{(s_i, t_i) \mid 1 \leq i \leq n\}$ with (4.1) and (4.2), and p and q as in (4.3), the beginning and ending points between s_p and t_q alternate.*

Proof.

$(i) \Longrightarrow (ii)$. Clearly, in P_n^k all twins are universal and $\{(2i, 2(i+k)+1) \mid 1 \leq i \leq n\}$ is a UIG model for P_n^k which yields the desired degree property.

$(ii) \Longrightarrow (iii)$. Since G is connected and not complete, we obtain $1 < \min\{p,q\} \leq \max\{p,q\} < n$. By (4.1) to (4.3), this implies

$$N[v_1] \subseteq N[v_2] \subseteq \ldots \subseteq N[v_{\min\{p,q\}-1}] \subseteq N[v_r]$$

for $\min\{p,q\} \leq r \leq p$, and

$$N[v_n] \subseteq N[v_{n-1}] \subseteq \ldots \subseteq N[v_{\max\{p,q\}+1}] \subseteq N[v_s]$$

for $q \leq s \leq \max\{p,q\}$. Since v_n is not adjacent to $v_{\min\{p,q\}-1}$ and v_1 is not adjacent to $v_{\max\{p,q\}+1}$, we obtain that no vertex in $[v_1, v_r) \cup (v_s, v_n]$ is universal and, so, (ii) implies that all the above inclusions are proper. This yields the dominator sequences described in (iii). It remains to prove that there are no further dominator sequences.

Since, by (ii), all middle vertices have the same degree, every dominator sequence contains at most one middle vertex. Consider two values i and j such that v_i is not a middle

vertex. If $i < \min\{p, q\}$ and $p < j$ then $v_1 \in N[v_i] \setminus N[v_j]$. Furthermore, either $q \leq j$ which implies that $v_n \in N[v_j] \setminus N[v_i]$, or $q > j$ which implies that v_j is a middle vertex and, by (ii), $d(v_j) = d(v_p) > d(v_i)$. Hence, in both cases, v_i and v_j do not both appear in one dominator sequence. Similarly, if $i > \max\{p, q\}$ and $j < q$, then v_i and v_j do not both appear in one dominator sequence. Altogether, this implies that there are no further dominator sequences as those described in (iii).

$(iii) \Longrightarrow (iv)$. Let $\mathcal{I}$ be a PIG model for G as in (iv). For contradiction, we may assume, by symmetry, that there are two consecutive beginning points s_i and s_{i+1} between s_p and t_q. By (4.3), the extreme point following s_p is t_1 which implies $i > p$. Since $\mathcal{I}$ is a PIG model, we obtain $N[v_i] \subseteq N[v_{i+1}]$. Since $v_1 \notin N[v_i] \cup N[v_{i+1}]$, the vertices v_i and v_{i+1} are not universal. Hence, by (iii), v_{i+1} and v_i are no twins and they appear in this order in some dominator sequence of G. By (iii), this implies the contradiction $i + 1 \leq p$.

$(iv) \Longrightarrow (v)$. trivial.

$(v) \Longrightarrow (i)$. Let $\mathcal{I}$ be a UIG model as described in (v). By (4.3), (v), and the fact that $s_i < s_j$ implies $t_i < t_j$, the order of the extreme points is as follows

$$s_1 < s_2 < \ldots < s_p < t_1 < s_{p+1} < t_2 < s_{p+2} < t_3 < \ldots < s_n < t_q < t_{q+1} < \ldots < t_n$$

which implies that $p = n - q - 1$ and that G is isomorphic to P_n^{p-1}. □

In view of the corresponding recognition algorithms for PCA graphs and PIG graphs (see *e.g.* [DHH96]), Theorems 4.1.1 and 4.1.2 imply that powers of cycles and paths can be recognized in linear time.

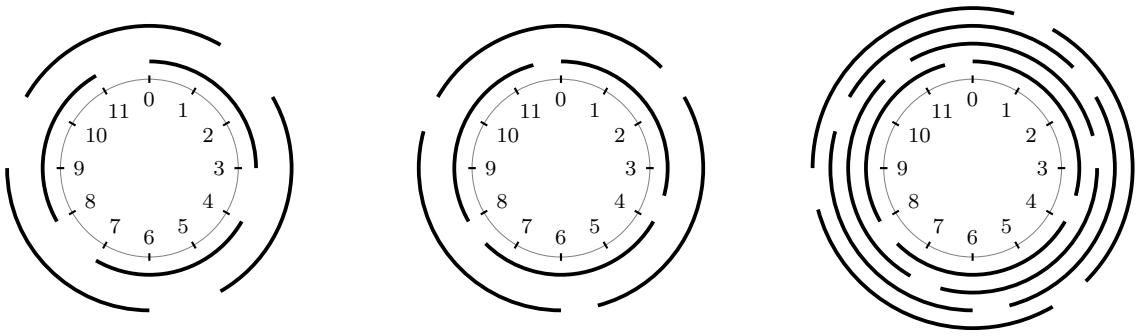
4.2 Induced subgraphs

In the previous section, we have shown that powers of cycles and powers of paths are special UCA graphs and special UIG graphs, respectively. In this section, we prove that the induced subgraphs of powers of cycles are exactly the UCA graphs. Similarly, we prove that the induced subgraphs of powers of paths are exactly the unit interval graphs.

Theorem 4.2.1.

(i) A graph is an induced subgraph of a power of a cycle if and only it is a UCA graph.

(ii) A graph is an induced subgraph of a power of a path if and only it is a UIG graph.



(a) (b) (c)

Figure 4.1: Proof of Theorem 4.2.1 (i): (a) is a UCA model, (b) is the model obtained by extending each arc of (a), and (c) is a UCA model of C_{12}^4.

Proof. Since the proofs of (i) and (ii) are very similar, we will only give details for (i).

By Theorem 4.1.1, powers of cycles are UCA graphs and, hence, so are their induced subgraphs. For the converse, let G be a UCA graph and $\mathcal{M} = (C, \{(s_v, t_v) \mid v \in V(G)\})$ be UCA model of G. As noted in Chapter 2, we may assume that all the extremes points are distinct. Even more, since $|V(G)|$ is finite, we can assume that all the extreme points are integer values of C (see *e.g.* Figure 4.1 (a)). By replacing t_v with $t_v + \frac{1}{2}$ for every $v \in V(G)$, we obtain a UCA model $\mathcal{M}'$ of G whose arcs have length $k+\frac{1}{2}$ for some integer value k (see *e.g.* Figure 4.1 (b)). Inserting one arc $(s, s+k+\frac{1}{2})$ for every integer $s \in C$ such that s is not an extreme of $\mathcal{M}'$, we obtain a PCA model in which every beginning point is followed by an ending point (see *e.g.* Figure 4.1 (c)). Hence, by Theorem 4.1.1, G is an induced graph of a power of a cycle. □

4.3 Yet another "proper = unit" proof

In [Rob69], Roberts proved that every PIG graph admits a UIG model (see Theorem 3.1.1). Roberts used the Scott-Suppes Theorem [SS58] to prove that every claw-free interval graph is a UIG graph. Since then, at least three constructive proofs have been proposed to show how to transform a PIG model into a UIG model. We will review these characterizations in Chapter 5, where we consider the problem of transforming a PIG model into a UIG model. In this section we give yet another proof of Roberts' "proper = unit" Theorem. Our proof comes with the guarantee that the length of every interval is at most $2n^2$, and it serves to exemplify how Theorems 4.1.2 and 4.2.1 can serve as a framework for thinking about PIG graphs.

The idea of our proof is to include new intervals into a PIG model $\mathcal{I}$, to build an extended model $\mathcal{I}' \supseteq \mathcal{I}$ such that the "middle" extreme points of $\mathcal{I}'$ alternate between beginning and ending points. By Theorem 4.1.2, the intersection graph of $\mathcal{I}'$ is isomorphic to the power of a path, hence, by Theorem 4.2.1, $\mathcal{I}'$ is equivalent to some UIG model and so is $\mathcal{I}$. To take into account the iterative process that it is used to obtain $\mathcal{I}'$, it is better to adapt our terminology.

Let $\mathcal{I} = \{(s_i, t_i) \mid 1 \leq i \leq n\}$ be a PIG model satisfying (4.2). The extremes of $\mathcal{I}$ in $[t_1, s_n]$ is the set of *middle extremes* of $\mathcal{I}$. (There are no middle extremes when $t_1 > s_n$.) The s-sequences and t-sequences that are contained in $[t_1, s_n]$ are called the *middle s-sequences* and *middle t-sequences*, respectively. In general, a *middle extreme sequence* is either a middle s-sequence or a middle t-sequence. In a PIG model of a power of a path, the middle points alternate between beginning points and ending points. This is equivalent to say that every middle extreme sequence is a singleton sequence. Say that an extreme point is *separated* if it belongs to a singleton extreme sequence.

To build the extended model $\mathcal{I}'$, we employ a *separation procedure* which is divided into two steps: the *t-separation* and the *s-separation*. The former separates the middle ending points, while the latter separates the middle beginning points. First, we apply the t-separation procedure to the given model $\mathcal{I}$, obtaining an intermediate model. Then we apply the s-separation procedure to the intermediate model and obtain the final model $\mathcal{I}'$. The separation procedure, the t-separation procedure and the s-separation procedure are shown in Algorithm 4.1, Algorithm 4.2 and Algorithm 4.3, respectively.

The effects of Algorithm 4.1 on a PIG model are depicted in Figure 4.2. The following theorem employs the separation procedure and shows its correctness.

Theorem 4.3.1. *The following statements are equivalent for a graph G:*

(i) G is a PIG graph.

(ii) G is an induced subgraph of a power of a path.

(iii) G is a UIG graph.

Moreover, if $\mathcal{I}$ is a PIG model, then Algorithm 4.1 applied on $\mathcal{I}$ yields a PIG model of a power of a path.

Proof.

$(i) \implies (ii)$. Let $\mathcal{I}$ be a PIG model of G and suppose, w.l.o.g., that G is connected. Perform Algorithm 4.1 on $\mathcal{I}$ to obtain the model $\mathcal{I}' \supseteq \mathcal{I}$. By induction, we show that all middle extremes of $\mathcal{I}'$ are separated. We start by discussing the effects of the t-separation step described in Algorithm 4.2. We prove that this procedure in fact terminates.

Examine the iteration corresponding to the leftmost non-separated middle ending point t_i, and let (s_j, t_j) and (s, t) be as in Algorithm 4.2. Call T_i and T_j to the t-sequences that

Algorithm 4.1 Separation procedure
Input: A connected PIG model $\mathcal{I}$.
Output: A connected PIG model $\mathcal{I}' \supseteq \mathcal{I}$ whose middle extremes are separated.

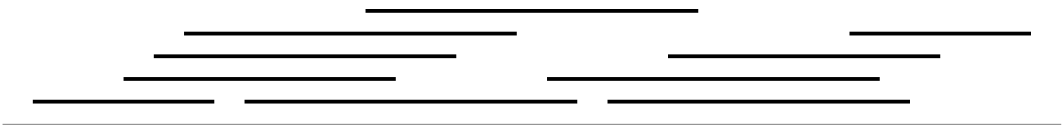
1. Apply Algorithm 4.2 to $\mathcal{I}$ to obtain $\mathcal{I}''$.
2. Apply Algorithm 4.3 to $\mathcal{I}''$ to obtain $\mathcal{I}'$.

Algorithm 4.2 t-separation procedure
Input: A connected PIG model $\mathcal{I}$.
Output: A connected PIG model $\mathcal{I}' \supseteq \mathcal{I}$ whose middle ending points are separated.

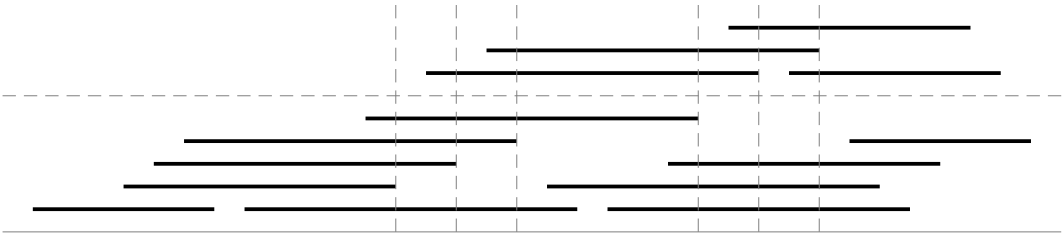
1. Set $\mathcal{I}' := \mathcal{I}$.
2. While there are non-separated middle ending points in $\mathcal{I}'$:
3. Let t_i be the leftmost non-separated middle ending point and (s_j, t_j) be the interval such that s_j precedes t_i in $\mathcal{I}'$.
4. Insert in $\mathcal{I}'$ a new interval (s, t), placing s immediately after t_i and t immediately after t_j.

Algorithm 4.3 s-separation procedure
Input: A connected PIG model $\mathcal{I}$.
Output: A connected PIG model $\mathcal{I}' \supseteq \mathcal{I}$ whose middle beginning points are separated.

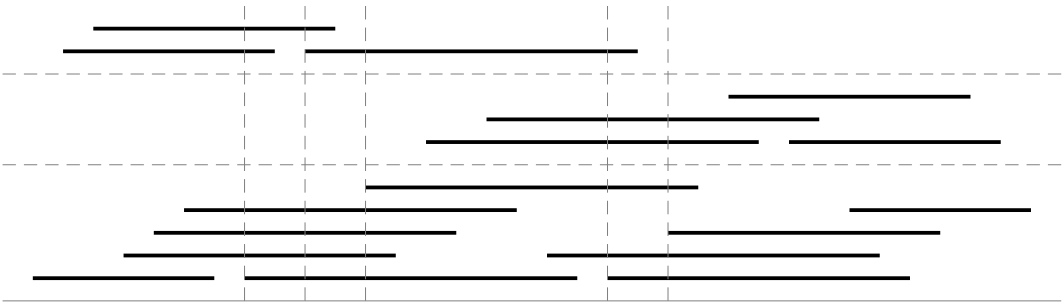
1. Set $\mathcal{I}' := \mathcal{I}$.
2. While there are non-separated middle beginning points in $\mathcal{I}'$:
3. Let s_i be the rightmost non-separated middle beginning point and (s_j, t_j) be the interval such that t succeeds s_i in $\mathcal{I}'$.
4. Insert in $\mathcal{I}'$ a new interval (s, t), placing t immediately before s_i and s immediately before s_j.



(a) A PIG model prior to separation.



(b) The PIG model after the t-separation step.



(c) The PIG model after the s-separation step.

Figure 4.2: The effects of the separations procedures. All middle ending points are followed by a beginning point after the t-separation step, while all middle beginning points are preceded by an ending point after the s-separation step.

contain t_i and t_j prior to the insertion of (s, t), respectively. After the insertion of (s, t), T_i is divided into two t-sequences, namely $\{t_i\}$ and $T_i \setminus \{t_i\}$. On the other hand, t is inserted into T_j and $T_j \neq T_i$. We have then two possibilities after the insertion of (s, t), according to the size of $T_i \setminus \{t_i\}$. If $|T_i \setminus \{t_i\}| = 1$ then the number of t-sequences to the right of the leftmost non-singleton t-sequence is decreased. Otherwise, the number of extremes in the leftmost non-singleton t-sequence decreases while the number of t-sequences to its right remains the same. Therefore the t-separation procedure terminates.

Next, examine the actual effect of inserting (s, t) in the t-separation procedure. Clearly, t_i becomes separated because s immediately follows it. Similarly, s is also separated. Furthermore, because the inclusion of (s, t) does not affect separated beginning points,

we conclude that no newly non-separated beginning points have been created during the examination of t_i. Consequently, at the end of the t-separation step, all the ending points are separated, all the beginning points of the intervals inserted during the t-separation step are also separated, while the separated beginning points of $\mathcal{I}$ have been preserved. Moreover, (s,t) cannot contain nor be contained in any other interval (s_k,t_k) of $\mathcal{I}'$, or otherwise (s_j,t_j) would also contain or be contained in (s_k,t_k), contradicting the fact that $\mathcal{I}'$ is proper before the inclusion of (s,t).

The s-separation step described in Algorithm 4.3 takes as input the model generated by the t-separation step, and transforms it into the final model $\mathcal{I}'$. The proof of its correctness is similar. We can conclude that after termination of Algorithm 4.1, $\mathcal{I}$ is included in a PIG model $\mathcal{I}'$, which has all its middle extremes separated. Therefore, by Theorem 4.1.2, the intersection graph of $\mathcal{I}'$ is isomorphic to P_j^k for some pair of values j and k. Then G is an induced subgraph of a power of a path.

$(ii) \implies (iii)$. See Theorem 4.2.1.

$(iii) \implies (i)$. Trivial. □

Theorem 4.3.1 can be used to effectively transform a PIG model $\mathcal{I}$ into an equivalent UIG model. First, run the separation algorithm on $\mathcal{I}$ to obtain a PIG model $\mathcal{I}_1$ of a power of a path. Then, use $(i) \implies (ii)$ of Theorem 4.1.2 to obtain a UIG model $\mathcal{I}_2$ equivalent to $\mathcal{I}_1$. Finally, remove every interval of $\mathcal{I}_2$ that was inserted by Algorithm 4.1. The model $\mathcal{I}_3$ so obtained is a UIG model equivalent to $\mathcal{I}$. The next theorem guaranties that the separation algorithm finishes in $O(n^2)$ steps and that all the intervals of model $\mathcal{I}_3$ have length at most $2n^2$.

Theorem 4.3.2. *Let $\mathcal{I}$ be a PIG model and $\mathcal{I}'$ be the model obtained from $\mathcal{I}$ after applying Algorithm 4.1. Then in $\mathcal{I}'$, there are less than $2n$ extremes of $\mathcal{I}' \setminus \mathcal{I}$ between any two consecutive extremes of $\mathcal{I}$.*

Proof. Let $\mathcal{I} = \{(s_i,t_i) \mid 1 \leq i \leq n\}$ be such that $s_1 < s_2 \ldots < s_n$. First, we consider the t-separation procedure described in Algorithm 4.2. Let $\mathcal{I}_t$ be the set of intervals included in $\mathcal{I}$ by the t-separation procedure. We attach a label to each ending point of $\mathcal{I} \cup \mathcal{I}_t$, as follows. The ending point t_i is assigned the label i, for every $1 \leq i \leq n$. Further, following the progress of the t-separation procedure, whenever a new interval (s,t) is included for the purpose of separating an ending point having label i from its successor in the model, the new ending point t also gets the label i (see Figure 4.3). Clearly, there can be several ending points in $\mathcal{I} \cup \mathcal{I}_t$ sharing the same label. However, in any t-sequence all the ending points have distinct labels, at any time of the execution of the t-separation procedure. Consequently, the size of any t-sequence at any stage of the algorithm is less than n. On the other hand, every non-singleton middle t-sequence T is separated by including one new beginning point of $\mathcal{I}_t$ between every consecutive pair of

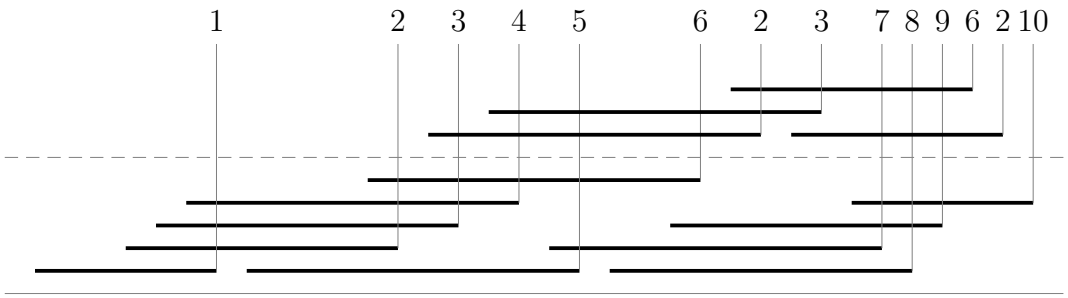


Figure 4.3: Labeling of the ending points in the proof of Theorem 4.3.2. Throughout the execution of the t-separation step, no t-sequence has two ending points with the same labels.

ending points of T. Hence, T is transformed into a sequence of size $2|T| - 1$, in which the ending points and beginning points alternate. By invariant, the first ending point of T belongs to $\mathcal{I}$. All the other $2|T| - 2$ extremes may perhaps belong to $\mathcal{I}_t$, but they are followed by a beginning point of $\mathcal{I}$. Therefore, in $\mathcal{I} \cup \mathcal{I}_t$ there are less than $2n$ extremes of $\mathcal{I}_t$ between any two extremes of $\mathcal{I}$. More precisely, less than i ending points of $\mathcal{I}_t$ have been included in the segment of $\mathcal{I}$ whose left extreme is t_i. Furthermore, all extremes of $\mathcal{I}_t$ have been included in segments of $\mathcal{I}$ of types t-t or t-s.

Next, we apply the s-separation procedure described in Algorithm 4.3 to the model $\mathcal{I} \cup \mathcal{I}_t$. Denote by $\mathcal{I}_s$ the set of intervals introduced in the model by the s-separation. That is, the final model is $\mathcal{I}' = \mathcal{I} \cup \mathcal{I}_t \cup \mathcal{I}_s$. We know that the t-separation procedure separates all ending points, while preserving the already separated beginning points, and neither increasing the size nor creating new non-singleton s-sequences. Consequently, the t-separation and s-separation procedures are independent. Furthermore, in $\mathcal{I}'$ there can be no three consecutive extremes, such that the first and the third belong to $\mathcal{I}_t$ and the second belongs to $\mathcal{I}_s$, or vice-versa. So, similarly as above, we can conclude that in $\mathcal{I}'$ there are less than $2n$ extremes of $\mathcal{I}_s$ between any two consecutive extremes of $\mathcal{I}$. More precisely, less than $n - i$ beginning points of $\mathcal{I}_s$ have been introduced in the segment of $\mathcal{I}$ whose right extreme is s_i. Furthermore, all extremes of $\mathcal{I}_s$ have been included in segments of $\mathcal{I}$ of types s-s or t-s.

Finally, we examine the total number of extremes of $\mathcal{I}_t \cup \mathcal{I}_s$ that have been introduced in a segment z of $\mathcal{I}$, according to the type of z. If z is an s-s type segment then less than $2n$ extremes of $\mathcal{I}_s$ and zero extremes of $\mathcal{I}_t$ have been introduced. Similarly, if z is of type t-t then zero extremes of $\mathcal{I}_s$ and less than $2n$ extremes of $\mathcal{I}_t$ have been introduced. In an s-t type segment of $\mathcal{I}$ no extremes of $\mathcal{I}_t \cup \mathcal{I}_s$ can be included at all. It remains to examine the case when z is a t-s segment. Let t_i and s_j be the left and right extremes of z, respectively. We know that less than i ending points of $\mathcal{I}_t$ and less than $n - j$ beginning points of $\mathcal{I}_s$ have been included in z. That is, less than $2(i + n - j)$ extremes

of $\mathcal{I}_s \cup \mathcal{I}_t$. However, we also know that $i < j$, because s_i must precede t_i. Consequently, there are less than $2n$ extremes of $\mathcal{I}' \setminus \mathcal{I} = \mathcal{I}_t \cup \mathcal{I}_s$ between any two consecutive extremes of $\mathcal{I}$. □

As a corollary, we can obtain a UIG model of a PIG graph in $O(n^2)$ time, when a PIG model of the graph is given as input. In Chapter 5 we show that this algorithm can be improved so as to run in $O(n)$ time.

5 Transformations between circular-arc models

Recall that there are ten different subclasses of circular-arc graphs: CA, NCA, HCA, NHCA, PCA, PHCA, UCA, UHCA, IG and PIG. For each of these classes, except for the NCA class, there are several $O(n+m)$ time algorithms that recognize if a given graph is an XCA graph, and these algorithms output an XCA model in the affirmative case (see *e.g.* Table 5.1). So, the recognition problem for XCA graphs is well solved. As for the model construction, first we refer to the more general class of circular-arc graphs. Considering that the corresponding models of this class can be represented by just $O(n)$ elements, there is a motivation for trying to find $O(n)$ time algorithms that solve the recognition and model construction problems for each of the subclasses. In this case, the input is a general circular-arc model of a graph G and the question is deciding whether G belongs to a restricted class of circular-arc graphs and, whenever affirmative, constructing the corresponding restricted circular-arc model. For example, given an arbitrary circular-arc model of a graph G, algorithms running in $O(n)$ time have been recently described to construct, whenever possible, a proper circular-arc model of G [Nus08], a unit circular-arc model of G [LS08], or a Helly circular-arc model of G [JLM+09].

Recall also that there are some circular-arc subclasses, such as the PIG class, which are defined by more than one kind of model, one strong and one weak. For these graphs there is also a motivation for constructing a strong model from a weak model. For instance, an

CA subclass	Algorithms
CA	[McC03, KN06]
HCA	[JLM+09, LS06]
NHCA	Chapter 3
PCA	[DHH96, KN09, Nus08]
PHCA	Chapter 3
UCA	[KN09, LS08]
UHCA	Chapter 3
IG	[BL76, COS98, HMPV00, KM89]
PIG	[Cor04, DHH96, HH05, HdFMPdM95]

Table 5.1: Linear-time recognition algorithms for the circular-arc subclasses.

$O(n)$ time algorithm for transforming any PCA model into an NPCA model was given in [LS08].

In the present chapter we consider the representation problems for several circular-arc subclasses. We propose new algorithms along the above lines and we discuss about the related certification and authentication problems. In Section 5.1 we propose a new $O(n)$ time algorithm that transforms any PCA model into an NPCA model. Our algorithm is simpler than the previous known algorithms, both in its implementation and its time complexity analysis. Next, in Section 5.2, we describe the $O(n+m)$ time algorithm to recognize NHCA graphs when circular-arc models are given as input. This is the only problem for which we have not found an $O(n)$ time algorithm. After, in Section 5.3, we show how to obtain in $O(n)$ time a PHCA model from either a PCA model or an NHCA model. As a corollary, we obtain an $O(n)$ time algorithm to transform a PCA model into a PIG model. Similarly, in Section 5.4 we present an $O(n)$ time algorithm to transform UCA models and PHCA models into UHCA models. Finally, in Section 5.5 we revisit the problem of transforming a PIG model into a UIG model, and we present a new algorithm for this problem.

5.1 Normalization of PCA models

In Chapter 2 we gave a new proof of Tucker's theorem which states that every PCA graph admits an NPCA model. But besides its theoretical interest, why is it important to find an NPCA model of a PCA graph? One answer is the following. Let $\mathcal{M}$ be any normal PCA model of a UCA graph G. Tucker [Tuc74] proved that G admits a UCA model $\mathcal{M}'$ in which all the extremes of $\mathcal{M}'$ appear in the same order as in $\mathcal{M}$. In all the polynomial-time algorithms to recognize UCA graphs (see [DGM+06, KN09, LS08]), an NPCA model is required as the input model of the recognition algorithm.

In this chapter we use our elementary proof of Tucker's theorem to develop a simple $O(n)$ time algorithm that transforms a general PCA model into an NPCA model. We begin with a brief review of the work done so far with respect to this problem.

In [DGM+06], Durán et al. show how to normalize a PCA model in $O(n^2)$ time. Their idea is to traverse each arc A_i so as to find if there is some arc A_j that together with A_i cover the circle, *i.e.*, $s(A_i), t(A_j), s(A_j), t(A_i)$ appear in this order in a traversal of the circle. If such A_j is found then $t(A_j)$ and all the ending points inside $(s(A_i), t(A_j))$ are moved so that $t(A_j)$ and $s(A_i)$ appear in this order. The main point is that every A_j can be found and shrunk in $O(n)$ time, for each arc A_i. Lin and Szwarcfiter [LS08] improve this algorithm so that all the pairs A_i, A_j are found and shrunk in $O(n)$ time, obtaining an $O(n)$ time normalization algorithm. The inconvenience of this new algorithm is that a somehow difficult analysis is used to show that every arc is shrunk at most twice.

Our normalization algorithm is depicted in Algorithm 5.1. The main loop computes the model $U_1(\mathcal{M})$ and Step 5 inserts back all the possibly removed arc. The algorithm is correct by Theorem 2.4.8 and Lemma 2.4.9. This algorithm has three advantages over the one in [LS08]. First, it is easier to implement and to describe. Second, every extreme of the model is traversed only once, while for the algorithm in [LS08] there is only the guaranty that the main cycle is repeated at most $5n$ times. Third, the model generated by Algorithm 5.1 is twin-consecutive, as Theorem 5.1.1 shows.

Algorithm 5.1 Normalization procedure.
Input: A PCA model $\mathcal{M}$ of a graph G.
Output: A twin-consecutive NPCA model of G.

1. Set $u := 0$ and $U := NULL$.
2. Traverse each arc A of $\mathcal{M}$ and apply the following operations when A contains at least $n-1$ extremes:
3. If $u = 0$ then set $u := 1$ and $U := A$.
4. Otherwise, remove A from $\mathcal{M}$ and set $u := u + 1$.
5. If $u > 0$ then duplicate $u - 1$ times the arc U as in Lemma 2.4.4.

With respect to the time complexity of the algorithm, Step 2 takes $O(1)$ time if each extreme has an index to its position in the traversal, and a pointer to the other extreme of the arc. This information can be easily preprocessed in $O(n)$ time with one traversal of the model. Therefore, the total time complexity of the algorithm is $O(n)$.

Theorem 5.1.1. *The model generated by Algorithm 5.1 is twin-consecutive.*

Proof. Immediately before the execution of Step 5, $\mathcal{M}$ is a PCA model with at most one universal arc. Then, by Theorem 3.3.1, $\mathcal{M}$ is twin-consecutive at this point of the execution. After the insertion of the $u-1$ copies of the universal arc as in Lemma 2.4.4, the beginning points as well as the ending points of the universal arcs are consecutive. Therefore, the generated model is twin-consecutive. □

5.2 Recognition of NHCA graphs

In this section we develop an algorithm to test whether a circular-arc graph is NHCA, when the input is a circular-arc model. The algorithm follows directly from Corollary 3.2.5 and its time complexity is $O(n + m)$.

Let $\mathcal{M}$ be a circular-arc model of a graph G. If there are neither two nor three arcs covering $C(\mathcal{M})$ then G is an NHCA graph and $\mathcal{M}$ is an NHCA model of G, so there is nothing to be done for this case. Otherwise, it is enough to check if $\mathcal{M}$ is equivalent to an interval model since, by Corollary 3.2.5, those NHCA graphs that admit a non NHCA model are interval graphs. If $\mathcal{M}$ is equivalent to an interval model, then G is an NHCA graph and any interval model of G is an NHCA model of G.

We use Algorithm 5.2 to test whether there are two or three arcs covering the circle. There, $NEXT(A)$ represents the arc crossing $t(A)$ whose ending point reaches farthest and $NEXT^2(A) = NEXT(NEXT(A))$. In the second traversal of the first loop, N is the arc reaching farthest of those crossing e, when e is reached. Thus, $NEXT$ is correctly computed in the first loop. Now, observe that if A_1 and A_2 cover $C(\mathcal{M})$, then A_1 and $NEXT(A_1)$ also cover $C(\mathcal{M})$. Similarly, if A_1, A_2, A_3 cover $C(\mathcal{M})$, then $A_1, NEXT(A_1)$ and $NEXT^2(A_1)$ also cover $C(\mathcal{M})$. Thus, $\mathcal{M}$ is an NHCA model if and only if neither $NEXT(A)$ nor $NEXT^2(A)$ crosses $s(A)$, for every $A \in \mathcal{M}$. Therefore, Algorithm 5.2 is correct.

Algorithm 5.2 Authentication of an NHCA model.

Input: A circular-arc model $\mathcal{M}$.
Output: If $\mathcal{M}$ is not NHCA, then two or three arcs that cover $\mathcal{M}$. Otherwise, there is no output.

1. Let A_1 be any arc of $\mathcal{M}$ and set $A := A_1$.

2. Traverse $C(\mathcal{M})$ twice from $s(A_1)$ and apply the following operation when an extreme e of an arc A is reached.

3. If $e = s(A)$ and $t(A)$ reaches farther than $t(N)$ then set $N := A$.

4. If $e = t(A)$ then set $NEXT(A) := N$.

5. Traverse $C(\mathcal{M})$ once again from $s(A_1)$ and apply the following evaluation when $s(A)$ is reached.

6. If $NEXT(A)$ crosses $s(A)$ then output $A, NEXT(A)$.

7. If $NEXT^2(A)$ crosses $s(A)$ then output $A, NEXT(A)$ and $NEXT^2(A)$.

With respect to the time complexity, all the operations of both loops take $O(1)$ time, thus the total time complexity of Algorithm 5.2 is $O(n)$.

To test if $\mathcal{M}$ is equivalent to an interval model when $\mathcal{M}$ is not NHCA, we compute the intersection graph G of $\mathcal{M}$ and we invoke the $O(n + m)$ time certified algorithm by Kratsch et al. [KMMS06]. Unfortunately, we were unable to find an $O(n)$ time algorithm to obtain an interval model from $\mathcal{M}$.

We now discuss the certification and the authentication procedures. When $\mathcal{M}$ is an NHCA model, the positive certificate is just $\mathcal{M}$. If $\mathcal{M}$ is not an NHCA model, but it is equivalent to an interval model, the certificate is provided by the certified interval graph recognition algorithm in $O(n+m)$ time [KMMS06]. If $\mathcal{M}$ is neither NHCA nor equivalent to an interval model then the negative certificate is obtained by combining the certificate of the interval graph recognition algorithm with the two or three arcs that cover the circle. This certificate is enough by Corollary 3.2.5. The negative certificate can be authenticated in $O(n)$ time as in [KMMS06]. To authenticate the positive certificates it is enough to test that the output model $\mathcal{M}'$ is NHCA and equivalent to $\mathcal{M}$. For the NHCA authentication use Algorithm 5.2, and for the isomorphism authentication use the $O(n)$ time algorithm by Curtis [Cur07]. The complete certified procedure is summarized in Algorithm 5.3.

Algorithm 5.3 Recognition of NHCA graphs.
Input: A circular-arc model $\mathcal{M}$.
Output: Either an NHCA model equivalent to $\mathcal{M}$ or a subset of arcs whose intersection graph is not NHCA.

1. Execute Algorithm 5.2 to authenticate if $\mathcal{M}$ is an NHCA model. If so, output $\mathcal{M}$.
2. Otherwise, execute the algorithm of [KMMS06] to the intersection graph G of $\mathcal{M}$. If G is an interval graph then output the interval model obtained by the algorithm. Otherwise, output the two or three arcs covering $C(\mathcal{M})$ together with the negative certificate obtained by the interval graph recognition algorithm.

5.3 Recognition of PHCA graphs

In this section we show two algorithms that can be used to recognize PHCA graphs. The first one transforms an NHCA model into a PHCA model in $O(n)$ time, if possible. The second one transforms a PCA model into a PHCA model in $O(n)$ time, if possible.

To transform an NHCA model into a PHCA model we need only to sort the extreme sequences as in Theorem 3.1.2. If the model so obtained is not PHCA then we can search for an induced $K_{1,3}$. We begin by describing how to sort all the extreme sequences in $O(n)$ time.

Let $\mathcal{M}$ be an NHCA model. For arcs $A_i, A_j \in \mathcal{A}(\mathcal{M})$ with nonempty intersection, say that $s(A_i)$ *appears before* $s(A_j)$ if $s(A_i)$ appears before $s(A_j)$ in a traversal of $C(\mathcal{M})$ from some point $p \in C(\mathcal{M}) \setminus (A_i \cup A_j)$ (see Figure 5.1). Observe that the point p must always exist because A_i and A_j do not cover $C(\mathcal{M})$. Similarly, say that $t(A_i)$ *appears before*

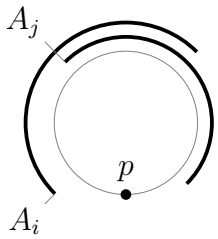


Figure 5.1: In this model $s(A_i)$ *appears before* $s(A_j)$ and $t(A_i)$ *appears before* $t(A_j)$.

$t(A_j)$ if $t(A_i)$ appears before $t(A_j)$ in a traversal of $C(\mathcal{M})$ from the same point p (see Figure 5.1). The sorting of the extremes sequences in Theorem 3.1.2 can be rephrased as follows. First, sort each t-sequence T so that, for $t(A_i), t(A_j) \in T$, if $s(A_i)$ appears before $s(A_j)$, then $t(A_i)$ appears before $t(A_j)$. Next, sort each s-sequence S so that, for $s(A_i), s(A_j) \in S$, if $t(A_i)$ appears before $t(A_j)$, then $s(A_i)$ appears before $s(A_j)$. The algorithms to sort the t-sequences and s-sequences are symmetric, so we only describe how to sort the t-sequences.

Let $T_1, \ldots, T_k$ be the t-sequences of $\mathcal{M}$ and T be the set of all the ending points corresponding to arcs that cross some fixed beginning point s. Consider the t-sequence T_i' that results from sorting the t-sequence T_i, for some $1 \leq i \leq k$. In T_i', all the ending points of $T_i \cap T$ appear before all the ending points of $T_i \setminus T$. Thus, we can sort all the t-sequences with four traversals of $C(\mathcal{M})$ as in Algorithm 5.4. In the first traversal of $C(\mathcal{M})$, Algorithm 5.4 marks all those arcs that cross the fixed beginning point $s(A_1)$. Thus, the ending points of T are precisely those ending points corresponding to the marked arcs. The second traversal is used to find all the t-sequences $T_1, \ldots, T_k$ of $\mathcal{M}$. The third traversal computes and sorts the sequences $T_{i,1} = T_i \cap T$ and $T_{i,2} = T_i \setminus T$, for every $1 \leq i \leq k$. Note that, in Step 9, $t(A)$ is stored at the end of either $T_{i,1}$ or $T_{i,2}$, and all the ending points corresponding to arcs whose beginning point appears before $s(A)$ were already stored. Finally, the last traversal of $C(\mathcal{M})$ replaces each t-sequence with the sorted t-sequence. Thus, the algorithm is correct.

With respect to the time complexity, all the operations of both loops take $O(1)$ time, while the computation of the t-sequences can be easily done in $O(n)$ time. Therefore, the total time complexity of the sorting algorithm is $O(n)$.

After sorting the extremes, we must check whether its output model $\mathcal{M}'$ is PHCA or not. Algorithm 5.4 does not modify the elements that compose each t-sequence, thus $\mathcal{M}'$ is NHCA. So, it is enough to check whether $\mathcal{M}'$ is PCA. That is, we ought to check if the beginning points of the arcs appear in the same order as the ending points. If affirmative, then $\mathcal{M}'$ is a PHCA model equivalent to the input model $\mathcal{M}$. Otherwise, there are two arcs A_i and A_j such that $s(A_i), s(A_j), t(A_j)$ and $t(A_i)$ appear in this order in a traversal $C(\mathcal{M}')$. Let L be the arc whose ending point appears immediately after $s(A_i)$ and R be

Algorithm 5.4 Sorting of the t-sequences of an NHCA model.
Input: An NHCA model $\mathcal{M}$.
Output: An NHCA model $\mathcal{M}'$ equivalent to $\mathcal{M}$ in which every t-sequences is sorted. That is, if $t(A_i), t(A_j)$ are extremes of the t-sequence T of $\mathcal{M}'$ and $s(A_i)$ appears before $s(A_j)$ then $t(A_i)$ appears before $t(A_j)$.

1. Let A_1 be any arc of $\mathcal{M}$.

//Find the arcs that cross $s(A_1)$.

2. Traverse $\mathcal{M}$ from $s(A_1)$ and apply the following operation when an extreme e of an arc A is reached:

3. If $e = s(A)$, then mark A.

4. If $e = t(A)$, then clear the mark of A.

//Sort the extremes

5. Traverse $\mathcal{M}$ to compute the family $T_1, \ldots, T_k$ of t-sequences of $\mathcal{M}$.

6. For each $i := 1, \ldots, k$, define $T_{i,1}$ and $T_{i,2}$ as empty sequences.

7. Traverse $\mathcal{M}$ from $s(A_1)$ and apply the following each time a beginning point $s(A)$ is reached:

8. Find the t-sequence T_i that contains $t(A)$.

9. If A is marked, insert $t(A)$ at the end of $T_{i,1}$; otherwise, insert $t(A)$ at the end of $T_{i,2}$.

10. Replace T_i with $T_{i,1}, T_{i,2}$ in $\mathcal{M}$ for every $1 \leq i \leq k$.

11. Output $\mathcal{M}$.

the arc whose beginning point appears immediately before $t(A_i)$. Arcs A_i, A_j, L, and R are taken as the negative certificate since, as in the proof of Theorem 3.1.2, they induce a circular-arc model of $K_{1,3}$ (see Figure 3.2). As for the authentication, the negative certificate can be tested to be an induced submodel of $K_{1,3}$ in $\mathcal{M}$ in $O(1)$ time, if it is implemented as a set of four pointers. To authenticate the positive certificate we ought to verify that the output model $\mathcal{M}'$ is normal, proper, Helly and equivalent to $\mathcal{M}$. The NHCA properties can be checked with Algorithm 5.2, while the test of whether $\mathcal{M}'$ is PCA or not is done as in the PHCA recognition algorithm (Steps 3–5 of Algorithm 5.5). Finally, if $\mathcal{M}'$ is PCA, then the equivalence of $\mathcal{M}$ and $\mathcal{M}'$ can be tested by running the PCA isomorphism algorithm of Chapter 7. Algorithm 5.5 summarizes the complete procedure.

We now proceed to describe the algorithm that transforms a PCA model $\mathcal{M}$ into a PHCA model $\mathcal{M}'$, when possible. The algorithm is a direct consequence of Theorem 3.2.10.

Algorithm 5.5 Recognition of PHCA graphs from NHCA models.
Input: An NHCA model $\mathcal{M}$.
Output: Either a PHCA model equivalent to $\mathcal{M}$ or an induced submodel of $K_{1,3}$.

1. Apply Algorithm 5.4 to sort the t-sequences, and the symmetric algorithm to sort the s-sequences.
2. Let $A_1 := (s_1, t_1)$ be any arc of $\mathcal{M}$.
3. For $i := 1, \ldots, n$ do:
4. Find the arc $A_{i+1} = (s_{i+1}, t_{i+1})$ whose beginning point is the first after s_i.
5. If $A_i \supset A_{i+1}$, then output A_i, A_{i+1}, the arc whose ending point appears first from s_i, and the arc whose beginning point appears first from t_i in a counterclockwise traversal of $C(\mathcal{M})$.
6. Output $\mathcal{M}$.

That is, it verifies if either $U_1(\mathcal{M})$ is an HCA model or $U_0(\mathcal{M})$ is an interval model. If affirmative, then $\mathcal{M}$ is equivalent to a PHCA model, and one such PHCA model can be obtained as in Theorem 3.2.10. Otherwise, the algorithm finds an induced submodel of W_4 or S_3.

The PHCA recognition algorithm is obtained by gluing together several parts of the algorithms developed so far. The first step is to compute $U_1(\mathcal{M})$ as in Steps 1–4 of Algorithm 5.1. The second step is to verify whether $U_1(\mathcal{M})$ is HCA. For this, it is enough to invoke Algorithm 5.2 so as to verify if $U_1(\mathcal{M})$ is NHCA, because $U_1(\mathcal{M})$ is NCA by Lemma 2.4.7. However, we can simplify the computation of $NEXT$ so that it takes only one traversal of $C(\mathcal{M})$. Let $t(A_1), \ldots, t(A_k)$ be a t-sequence of $\mathcal{M}$. Since $\mathcal{M}$ is PCA, then $NEXT(A_i)$ is the arc of $\mathcal{M}$ whose beginning point is closest to $t(A_i)$ in the counterclockwise direction. Hence $NEXT(A_i) = NEXT(A_1)$, for every $1 \leq i \leq k$. Therefore, with only one traversal we can find $NEXT(A)$ for every $A \in \mathcal{A}(\mathcal{M})$. The last step is to test whether $U_0(\mathcal{M})$ is a PIG model or not, whenever $U_1(\mathcal{M})$ has a universal arc A. Instead of doing this, we can check that no beginning point appears before an ending point inside A, as it is done in Theorem 3.2.10. All these steps can be implemented so as to run in $O(n)$ time with techniques similar to those discussed so far.

The algorithm described above can be modified so as to produce certificates in $O(n)$ time. When $U_1(\mathcal{M})$ is a PHCA model, we can obtain a PHCA model equivalent to $\mathcal{M}$ by including the universal arcs that where possible removed by Steps 1–4 of Algorithm 5.1. This can be achieved as in Step 5 of Algorithm 5.1. To obtain the positive certificate when $U_0(\mathcal{M})$ is a PIG model, we refer to the proof of Theorem 3.2.10, in particular, the implication $(iii) \Longrightarrow (i)$. In this situation, $U_1(\mathcal{M})$ contains a universal arc A. To obtain the required model, we include in $\mathcal{M}_0$ the arc $\overline{A}$, and then we can include the

possible remaining universal arcs as in Step 5 of Algorithm 5.1. The authentication of these certificates can be done in $O(n)$ time, as discussed for Algorithm 5.5.

The algorithm fails to transform the input PCA model into a PHCA model when $U_1(\mathcal{M})$ is not HCA and $U_0(\mathcal{M})$ is not PIG. According to Theorem 3.2.10, an induced submodel of $\mathcal{M}$ whose intersection graph is either isomorphic to W_4 or to S_3 can be obtained as follows. Let A_1, A_2, and A_3 be the three arcs that together cover the circle of $U_1(\mathcal{M})$. If none of these arcs is universal, then, as in Theorem 3.2.10, we know that there are three arcs B_1, B_2, B_3, such that B_i intersects A_j and not A_i, for all $1 \leq i, j \leq 3$, $i \neq j$. In this case, the arcs A_1, A_2, A_3, B_1, B_2, and B_3 either induce a model of S_3 or contain a model of W_4. On the other hand, if one among A_1, A_2, A_3, say A_1, is a universal arc then there are arcs L, R, such that $s(R)$ precedes $t(L)$ in A_1. In this situation, a negative certificate can be obtained as above by replacing A_1 with either R or L when $R = A_2$ or $L = A_3$. Finally, when $R \neq A_2$ and $L \neq A_3$, the arcs A_1, A_2, A_3, L and R induce the model of a forbidden W_4. The authentication takes $O(1)$ time if the forbidden submodel is stored as a set of five or six pointers to the corresponding arcs of the model. We summarize the above discussion in Algorithm 5.6.

5.4 Recognition of UHCA graphs

In this section we briefly discuss how to transform either a UCA or a PHCA model $\mathcal{M}$ into a UHCA model, when possible. We begin with the case in which $\mathcal{M}$ is PHCA. In this case, apply the algorithm in [LS08] to transform $\mathcal{M}$ into a UCA model $\mathcal{M}'$, if possible. Since this algorithm preserves the order of the extremes of $\mathcal{M}$ then $\mathcal{M}'$ is both UCA and PHCA, *i.e.*, $\mathcal{M}'$ is UHCA. This algorithm takes $O(n)$ time and the model $\mathcal{M}'$ so generated can be authenticated to be UCA in $O(n)$ time. If $\mathcal{M}$ is not equivalent to a UHCA model, then apply the algorithm of [KN09] to generate a negative certificate in $O(n)$ time. This negative certificate can also be authenticated in $O(n)$ time.

Finally, consider the case in which $\mathcal{M}$ is a UCA model. If $\mathcal{M}$ has two arcs that cover the circle, then the intersection graph of $\mathcal{M}$ is a complete graph and an equivalent UIG model is easily obtained in $O(n)$ time. Suppose, then, that $\mathcal{M}$ is an NCA model, and consider the model $U_1(\mathcal{M})$. If $U_1(\mathcal{M})$ is HCA then it is also UHCA and a UHCA model equivalent to $\mathcal{M}$ can be easily obtained in $O(n)$ time by duplicating the universal arc of $U_1(\mathcal{M})$, if existing, as in Lemma 2.4.4. If $U_1(\mathcal{M})$ is not HCA but $U_0(\mathcal{M})$ is an interval model then a PIG model equivalent to $\mathcal{M}$ can be obtained by applying Algorithm 5.6. A UIG model equivalent to $\mathcal{M}$ can be constructed in $O(n)$ time as in Section 5.5. In the last case, if $U_1(\mathcal{M})$ is not an HCA model and $U_0(\mathcal{M})$ is not an interval model then, by Theorem 3.2.10, $\mathcal{M}$ is not equivalent to a UHCA model and a negative certificate is obtained as in Algorithm 5.6. All the certificate authentications take $O(n)$ time as before.

Algorithm 5.6 Recognition of PHCA graphs from PCA models.
Input: A PCA model $\mathcal{M}$.
Output: Either a PHCA model equivalent to $\mathcal{M}$ or an induced submodel of W_4 or S_3.

1. Apply Steps 1–4 of Algorithm 5.1 to obtain $U_1(\mathcal{M})$.
2. Execute Algorithm 5.2 (simplified for PCA graphs), to verify whether $U_1(\mathcal{M})$ contains three arcs that cover the circle. If negative, output the model obtained by the execution of Step 5 of Algorithm 5.1 on $U_1(\mathcal{M})$.
3. Let A_1, A_2 and A_3 be the three arcs of $U_1(\mathcal{M})$ that were obtained in the previous step.
4. If A_1, A_2 and A_3 are not universal, then:
5. Let B_i be the arc whose beginning point is the first from $t(A_i)$, for $i \in \{1, 2, 3\}$.
6. If B_1, B_2 and B_3 are pairwise disjoint, then output $\{A_i, B_i\}_{1 \leq i \leq 3}$; otherwise, output A_1, A_2, A_3 and two intersecting arcs of B_1, B_2 and B_3.
7. Sort A_1, A_2 and A_3 so that A_1 is the universal arc.
8. Traverse $\mathcal{M}$ from $s(A_1)$ to $t(A_1)$ to find if there are two arcs L and R such that $s(R)$ appears before $t(L)$. If L and R are found then:
9. If $R = A_2$, then set $A_1 := R$ and goto Step 5. If $L = A_3$, then set $A_1 := L$ and goto Step 5. Output A_1, A_2, A_3, L and R.
10. Let $\mathcal{M}' := (\mathcal{M} \setminus \{A_1\}) \cup \{\overline{A_1}\}$.
11. Duplicate $\overline{A_1}$ in $\mathcal{M}'$ as many times as arcs where removed in Step 1, and return the model so obtained.

5.5 UIG models of PIG graphs

Recall that, by Theorem 3.1.1, the classes of UIG and PIG graphs are equal. For the recognition problem of UIG graphs we may apply any of the many PIG recognition algorithms (see eg. [Cor04, DHH96, HH05, HdFMPdM95, KN09]). However, not all these algorithms are able to produce a UIG model of the graph, *i.e.*, a model in which all the intervals have the same length. We begin this section with a brief description of the known representation algorithms for this class.

In [CKN+95], Corneil et al. developed an $O(n + m)$ time algorithm for the recognition and representation problems, when the input is the adjacency lists of the graph. Their algorithm is divided into two main phases. In the first phase they are able to produce a PIG order of the vertices of the graph. In the second phase they actually transform this PIG order into a UIG model of the graph. For the second phase, a special tree is

built, and then a postorder traversal of this tree is used to obtain the desired model. One of the nice properties of the output model is that every extreme is represented by an integer in the range $[0, n^2]$, which is asymptotically the best possible.

A few years later, Bogart and West [BW99] gave a new simple and constructive proof of Roberts' "proper = unit" Theorem (see Theorem 3.1.1). This proof yields—with a naive implementation—an $O(n^2)$ operations algorithm to transform a PIG model into a UIG model. However, the extremes of the intervals could be of exponential value with respect to n. In the book by Spinrad [Spi03] there is a short description of the algorithm by Corneil et al., in the case that the input is a PIG model. This algorithm is equivalent to the second phase of the original algorithm, but the input PIG order is replaced with a PIG model of the graph. The algorithm obtained by Spinrad runs in $O(n+m)$ time, but this is not linear with respect to the size of the input. More recently, Gardi [Gar07] gave one more proof of the "proper = unit" Theorem, which yields an $O(n)$ operations algorithm. Unfortunately, the extremes of the output model may have exponential value.

In [Gar07], Gardi states that it would be interesting to find a linear-time (and linear-space) algorithm for constructing a UIG model with extremes of polynomial value, when a PIG model is given as input, without traversing a tree. This problem involves two interesting questions by its own. The first question is how to find an $O(n)$ time algorithm for constructing a UIG model with extremes of polynomial value. The second question is how to do it without traversing any tree. Lin and Szwarcfiter gave an answer to the former question in [LS08], where they develop an $O(n)$ time and space algorithm to transform a PCA model into a UCA model, whenever possible. When the input of the algorithm is a PIG model, the output is a UIG model. However, this algorithm is to general for UIG graphs, and it requires several traversals of the input model and the use of network flow techniques.

In this section we revisit both of the questions posed by Gardi. To answer the first question we show how to implement the algorithm by Corneil et al., so that it runs in $O(n)$ time when the input is a PIG model. This is not an answer to the second question because we still need to traverse a tree. To answer the second question, we develop a new $O(n)$ time algorithm based on the separation algorithm of Chapter 4. This algorithm requires only two traversals of the input model and very few computations with numbers in the range 0 to $O(n^2)$.

5.5.1 Corneil et al. algorithm

Let $\mathcal{I}$ be a PIG model with intervals $I_1, I_2, \ldots, I_n$, where $I_i = (s_i, t_i)$ and $s_1 < s_2 < \ldots < s_n$. The idea of Corneil et al. is to build a special breath-first search (BFS) tree, where the root corresponds to I_1. This tree is used, in part, to check if the input graph

is actually a PIG graph. However, in this case we already know that the graph is PIG, thus the construction of the BFS tree can be simplified as follows. For $i = 2, \ldots, n$, let $PREV(I_i)$ be the interval whose ending point appears first from $s(I_i)$. Let T be an ordered tree with vertices $v_1, \ldots, v_n$ such that

- v_i is the parent of v_j if and only if $I_i = PREV(I_j)$ and
- if v_i and v_j are siblings nodes then v_i appears before v_j if and only if $i < j$.

Tree T is a BFS tree of G from the vertex v_1 that corresponds to I_1. The algorithm by Corneil et al. is really simple. Let ℓ_i be the level of v_i in T, k_i be the position of v_i in a postorder traversal of T, and define $a_i = n\ell_i + k_i$ and $b_i = (n+1)\ell_i + k_i$. Then, the model $\{(a_i, b_i) \mid 1 \leq i \leq n\}$ is a UIG model equivalent to $\mathcal{I}$ [CKN+95].

The function $PREV$ and the tree T can be easily computed in $O(n)$ time, as we did with $NEXT$ before in this chapter. As it is well known, the computation of ℓ_i and k_i takes $O(n)$ time, and so the whole algorithm needs only $O(n)$ operations. Observe that all the operations are done with numbers in the range $[0, n^2]$, thus the first question of Gardi is actually answered by the algorithm of Corneil et al. It is not our objective to show how to implement this algorithm in the most space efficient way. However, it is good to note that there is no need to actually compute all the edges of T. Observe that the children of a node v_i form a range $\textsc{Lin}[v_j, v_k]$, thus the children of v_i can be stored with only two pointers.

5.5.2 An algorithm with no tree traversals

In Chapter 4 we developed an algorithm to transform a PIG model $\mathcal{I}$ into an equivalent UIG model. Recall that this algorithm has three main steps. First, the Separation Algorithm on page 64 is run with input $\mathcal{I}$, to obtain a PIG model $\mathcal{I}_1$ of a power of a path. Then, implication $(i) \Longrightarrow (ii)$ of Theorem 4.1.2 is used to obtain a UIG model $\mathcal{I}_2$ equivalent to $\mathcal{I}_1$. Finally, every interval of $\mathcal{I}_2$ that was inserted by the Separation Algorithm is removed. The model so obtained is a UIG model equivalent to $\mathcal{I}$.

By Theorem 4.3.2, model $\mathcal{I}_2$ in the above algorithm contains $O(n^2)$ extremes. Therefore, any algorithm for explicitly constructing $\mathcal{I}_2$ would require $\Omega(n^2)$ steps, which precludes an $O(n)$ time algorithm—our initial goal. However, we do not have to construct $\mathcal{I}_2$ explicitly. We just need to find the submodel of $\mathcal{I}_2$ induced by the extremes of $\mathcal{I}$, which of course has $2n$ extremes. Our idea is to determine the number of extremes of $\mathcal{I}_2 \setminus \mathcal{I}$ which would be included inside each segment of $\mathcal{I}$ by the separation procedures, without actually including them. In this way, we determine the segment lengths of the desired unit interval model of G, obtaining the required unit model. The final algorithm follows this idea.

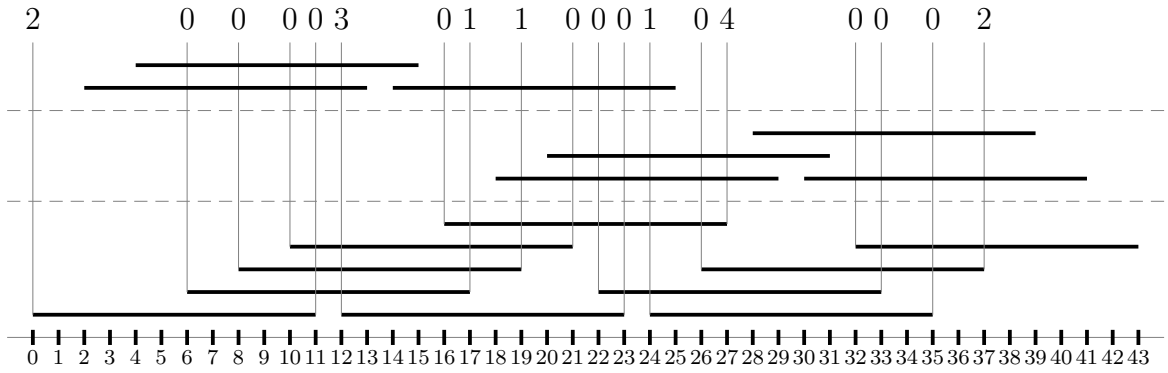


Figure 5.2: The UIG model corresponding to $\mathcal{I}_2$ for the model $\mathcal{I}$ in Figure 4.2. The value $r(e)$ is shown at the top of the picture for every extreme $e \neq t_n$. If the position of each extreme e is modified according to Equation 5.1 then the size of each interval is 11.

Let $\{(s_i, t_i)\}_{1 \leq i \leq n}$ be the intervals of $\mathcal{I}$, where $s_1 < s_2 < \ldots < s_n$. For an extreme e, we denote by e^- and e^+ the extremes of $\mathcal{I}$ lying immediately before and after e, respectively. Denote by $r(e)$ the number of extremes of $\mathcal{I}_2$ belonging to (e, e^+) in $\mathcal{I}$, for each extreme $e \neq t_k$ of $\mathcal{I}$. In the output UIG model, the position $p(e)$ of each extreme e of $\mathcal{I}$ is equal to the position $p(e)$ of e in $\mathcal{I}_2$. To compute $p(e)$, we first compute the $r(e)$ values since, at is shown in Figure 5.2, we know that the position of e in $\mathcal{I}_2$ is:

$$p(e) = \begin{cases} 0, & \text{if } e = s_1; \\ p(e^-) + 2(r(e) + 1), & \text{if } e \in (s_1, t_1^-) \cup (s_n^+, t_n]; \\ p(e^-) + r(e) + 1, & \text{otherwise.} \end{cases} \tag{5.1}$$

Our algorithm is based on the above remarks. It consists of two stages. In Stage 1, we increase the values of r to take into account all the extremes that are inserted for the purpose of separating the beginning points. In Stage 2, we further increase the values of r, taking into account the separation of the ending points. In fact, Stages 1 and 2 are variations of the s-separation and t-separation procedures that were described in Page 64. However, we change the order of the operations of the Separation Algorithm; first we separate the beginning points and second we separate the ending points. The reason for this change is that, while the ending points are separated, we output the extremes of every interval so as to minimize the number of model traversals.

Algorithm 5.7 is used to transform a PIG model into an equivalent UIG model. We briefly discuss the implementation of Stage 1. First observe that when a middle beginning point s_i is traversed in this stage, t_j is the first ending point to the right of s_i. On the other hand, $r(e)$ represents the number of extreme points that where inserted into (e, e^+) for

Algorithm 5.7 PIG to UIG model transformation

Input: A connected PIG model $\mathcal{I} = \{(s_i, t_i)\}_{1 \leq i \leq n}$ where $s_1 < s_2 < \ldots < s_n$.
Output: A UIG model equivalent to $\mathcal{I}$.

1. Set $r(e) := 0$ for every extreme e of $\mathcal{I}$.

//Stage 1

2. Set $I := (s_n, t_n)$.

3. For each extreme e in a traversal of $\mathcal{I}$ from right to left, do:

4. If e is an ending point t_j then set $I := (s_j, t_j)$.

5. Otherwise, e is a beginning point s_i and $I = (s_j, t_j)$. If e is a middle extreme then:

6. Set $r(s_j^-) := r(s_j^-) + r(s_i^-)$.

7. Set $r(s_i^-) := r(s_i^-) + r(s_i^-)$.

8. If s_i^- is a beginning point then add one to both $r(s_i^-)$ and $r(s_j^-)$.

//Stage 2

9. Set $I := (s_1, t_1)$ and $p(s_1) := 0$.

10. For each extreme e in a traversal of $\mathcal{I}$ from left to right, do:

11. If e is a beginning point s_j then set $I := (s_j, t_j)$.

12. Otherwise, e is an ending point t_i and $I = (s_j, t_j)$. If e is a middle extreme then:

13. Set $r(t_j) := r(t_j) + r(t_i)$.

14. Set $r(t_i) := r(t_i) + r(t_i)$.

15. If t_i^+ is an ending point then add one to both $r(t_i)$ and $r(t_j)$.

//Output of the model

16. If $e \in (s_1, t_1^-) \cup (s_n^+, t_n]$ then set $p(e) := p(e^-) + 2(r(e^-) + 1)$. Otherwise set $p(e) := p(e^-) + r(e^-) + 1$.

the purpose of separating all the beginning points to the right of s_i, for every extreme e. In particular, $r(s_i^-)$ accounts only for beginning points. The virtual insertion of intervals is done in Steps 6–8. To separate all the beginning points between s_i^- and s_i we need to insert $r(s_i^-)$ ending points. All these new ending points correspond to intervals whose beginning points appear inside (s_j^-, s_j). We simulate these insertions in Steps 6 and 7, where we increase the values of $r(s_j^-)$ and $r(s_i^-)$, respectively. Similarly,

if s_i^- is a beginning point, we should add one more ending point to separate the last beginning point inserted from s_i^-. This new interval also has its beginning point inside (s_j^-, s_j) and its ending point inside (s_i^-, s_i), thus we increase by one the numbers $r(s_j^-)$ and $r(s_i^-)$ in Step 8. The implementation of Stage 2 is analogous. However, note that $p(e)$ is computed in this second traversal according to Equation 5.1.

The time and space required by the algorithm is clearly $O(n)$, and the length of the largest segment of the unit model is less than $2n$ by Theorem 4.3.2. This algorithm is, therefore, a solution to Gardi's problem.

5.6 Summary

In Table 5.2 we summarize the time complexities of the transformation algorithms between the CA subclasses.

From	To	Time complexity	References
CA	NCA	open	
co-bipartite CA	NCA	$O(n^5m^6 \log m)$	[Mül97, HH04]
CA	HCA	$O(n)$	[JLM+09]
CA ∪ NCA	PCA	$O(n)$	[Nus08]
PCA	UCA	$O(n)$	[LS08]
CA ∪ HCA ∪ NCA	NHCA	$O(n+m)$	[KMMS06] & § 5.2
NHCA	PHCA	$O(n)$	§5.3
PCA	PHCA	$O(n)$	§5.3
UCA	UHCA	$O(n)$	§5.3 & §5.4
PHCA	UHCA	$O(n)$	[LS08] & §5.3
NHCA	IG	$O(n)$	Corollary 3.2.5
PHCA ∪ UHCA	PIG	$O(n)$	§5.3
IG	PIG	$O(n)$	§5.2
PIG	UIG	$O(n)$	[CKN+95] & §5.5

Table 5.2: Time complexities of the transformation algorithms. All algorithms proposed in this chapter are certified.

6 Dynamic recognition of PCA and PHCA graphs

In a previous chapter we dealt with the recognition and representation problems for several subclasses of circular-arc graphs. In that chapter we developed algorithms that transform a circular-arc model $\mathcal{M}$ into a specific circular-arc model that satisfies some property P, whenever possible. A more classic approach to the recognition and representation problems is to develop an algorithm that, given a graph G, first recognizes if G admits a circular-arc model with property P and then outputs such a circular-arc model. In this chapter we consider this classical view of the proper circular-arc graph recognition and representation problems, but we do it for dynamically changing graphs.

For a simple motivation, suppose that we have a PCA model $\mathcal{M}$ of a graph G and we are asked to find a PCA model of $G \cup \{v\}$ for a vertex $v \notin V(G)$. Of course, vertex v is given together with its set of neighbors $N(v) \subseteq V(G)$. One way to solve our problem is to run the fastest PCA recognition and representation algorithm on $G \cup \{v\}$, to find the desired model in $O(n + m)$ time. That is, we can throw away the PCA model $\mathcal{M}$, and compute a new PCA model $\mathcal{M}'$ from scratch. However, considering that the size of the input v is only $d(v)$, our naive solution is not as efficient as it could be. Instead of throwing away the model $\mathcal{M}$, we can use it to find a PCA model of $G \cup \{v\}$ more efficiently.

The *dynamic graph recognition and representation problem* for a class of graphs $\mathcal{C}$, or simply the *dynamic recognition problem* for a class $\mathcal{C}$, is the problem of maintaining a representation of a dynamically changing graph while the graph belongs to the class $\mathcal{C}$. The input of the problem is a graph G together with a sequence of update operations that change the size of the graph. A *dynamic recognition algorithm* is composed by the algorithm that builds the initial representation of G and the algorithms that apply each update operation. The time efficiency of a dynamic algorithm is measured by the time efficiency of all the algorithms that compose the dynamic algorithm. There are other kinds of dynamic graph problems besides the graph recognition and representation problems. Eppstein et al. [EGI99] consider such dynamic graph problems and give a nice introduction to the subject.

There are four special dynamic recognition problems, according to the update operations allowed. The basic recognition problem in which no update is allowed is called *static*;

the input of a static recognition problem is a graph and the output is whether the graph belongs to some class. For instance, all the recognition algorithms in Chapter 5 are static. The dynamic recognition problem in which only updates that increment the size of the graph are allowed is called *incremental*. Similarly, the dynamic recognition problem in which only updates that decrement the size of the graph are allowed is called *decremental*. Finally, the recognition problem in which both kinds of updates are allowed is called *fully dynamic*.

In traditional fully dynamic graph algorithms, only insertions and removals of edges are supported [EGI99]. There is no loss of generality, because any graph on n vertices can be obtained from another graph on n vertices with edge insertions and removals. However, these operations are not general enough for the dynamic recognition problems, since we cannot guarantee to obtain all the graphs in the class without going through intermediate graphs that do not belong to the class. For instance, in the dynamic P_3-free recognition problem, we cannot obtain a triangle without going through an intermediate P_3. Similarly, a tree on n vertices cannot be obtained from another tree on n vertices if only edge insertions and removals are allowed. For a graph G, the updates allowed in a general dynamic graph recognition problem are:

- Vertex insertion: given a vertex $v \notin V(G)$ and a set of neighbors of $N(v) \subseteq V(G)$, update the representation of G into a representation of $H = G \cup \{v\}$, if possible, where $N_H(v) = N(v)$.
- Vertex removal: given a vertex $v \in G$, update the representation of G into a representation of $G \setminus \{v\}$, if possible.
- Edge insertion: given an edge $vw \notin E(G)$, update the representation of G into a representation of $G \cup \{vw\}$, if possible.
- Edge removal: given an edge $vw \in E(G)$, update the representation of G into a representation of $G \setminus \{vw\}$, if possible.

A dynamic recognition problem in which only insertions and removals of vertices are allowed is called a *vertex-only* problem. Similarly, a dynamic recognition problem in which only insertions and removals of edges are allowed is called an *edge-only* problem. There are problems in which other structures, such as complete sets, are included or removed (*e.g.* [kY06]), but we shall not deal with such problems.

Several incremental and fully dynamic recognition algorithms have been developed in the last years. Just to name a few examples from the last six years,

- In 2004, Shamir and Sharan [SS04] developed a fully dynamic cograph recognition algorithm that runs in $O(1)$ time per edge modification and in $O(d(v))$ time per vertex modification.

- Between 2005 and 2006, Crespelle and Paul devised a certified fully dynamic recognition algorithm for directed cographs that runs in $O(1)$ time per edge modification and in $O(d(v))$ time per vertex modification [CP06], and a fully dynamic permutation graph recognition algorithm that runs in $O(n)$ time per operation [CP05].
- Also in 2006, Nikolopoulos et al. [NPP06] proposed a dynamic algorithm for the recognition of P_4-sparse graphs that requires $O(1)$ time per edge update and $O(d(v))$ time per vertex insertion. This algorithm can be converted into a fully dynamic algorithm that handles each edge removal in $O(\log n)$ time, at the extra cost of $O(\log n)$ time per edge updated and $O(d(v) \log n)$ per vertex insertion.
- In 2007, Tedder and Corneil [TC07] gave an optimal edge-only fully dynamic algorithm for the recognition of distance hereditary graphs.
- In 2008, Ibarra [Iba08] presented different algorithms for the edge-only dynamic chordal graph recognition problem and an optimal edge-only dynamic recognition algorithm for split graphs.
- Even more recently, Ibarra developed edge-only dynamic algorithms for the recognition of interval and PIG graphs. The former runs in $O(n \log n)$ per edge modification [Iba09a] while the latter runs in $O(\log n)$ time per edge modification [Iba09b].
- Finally, Crespelle [Cre09] designed a fully dynamic algorithm for the recognition of interval graphs that handles each operation in $O(n)$ time.

With respect to circular-arc graphs and its subclasses—in addition to the algorithms by Ibarra and Crespelle—, Hsu [Hsu96] developed a vertex-only incremental algorithm for the recognition of interval graphs that runs in $O(d(v) + \log n)$ time per vertex insertion; Deng et al. [DHH96] developed a vertex-only incremental algorithm for the recognition of connected PIG graphs that runs in $O(d(v))$ time per vertex insertion; and Hell et al. [HSS01] developed a fully dynamic algorithm for the recognition of PIG graphs that runs in $O(d(v) + \log n)$ time per vertex update and in $O(\log n)$ time per edge update. Hell et al. [HSS01] also developed an incremental PIG graph recognition algorithm and a decremental PIG graph recognition algorithm. Both of them run $O(d(v))$ time per vertex update and in $O(1)$ time per edge update.

In this chapter we develop new dynamic algorithms for the PCA and PHCA recognition problems. For the PHCA recognition problem we generalize all the results by Hell et al. That is, we develop:

- a fully dynamic algorithm that runs in $O(d(v) + \log n)$ per vertex update and in $O(\log n)$ time per edge update,
- an incremental algorithm that runs in $O(d(v))$ time per vertex insertion and $O(1)$ time per edge insertion, and
- a decremental algorithm that runs in $O(d(v))$ time per vertex removal and $O(1)$ time per edge removal.

For the PCA recognition problem, in turn, we develop vertex-only dynamic algorithms that run in the same time as the algorithms by Hell et al. That is, we develop[1]:

- a vertex-only fully dynamic algorithm that runs in $O(d(v) + \log n)$ per vertex update, and
- a vertex-only incremental algorithm that runs in $O(d(v))$ time per vertex insertion.

There is a strong relationship between the dynamic representations of the PIG, PHCA and PCA algorithms. In fact, the dynamic representation that is maintained in the algorithms by Deng et al. and by Hell et al. is essentially a straight enumeration, whereas the representation that we use for the PCA algorithm is essentially a round enumeration (see Chapter 3). Furthermore, the representation that the PCA recognition algorithm generates is a locally straight enumeration when G is PHCA, while it is a straight enumeration when G is a PIG graph. Therefore, the dynamic PCA recognition algorithm is also able to answer in constant time if the dynamic graph is a PIG or PHCA graph.

This chapter is organized as follows. In Section 6.1 we give a brief overview of the algorithms by Deng et al. and by Hell et al. These algorithms play a central role in the present chapter for two reasons, one practical and one theoretical. The practical reason is that we use them when an update operation is applied to a PIG graph. The theoretical reason is that our algorithm follows the same ideas as these algorithms, and hence we prove results similar to those in [DHH96, HSS01]. In Section 6.2 we describe the dynamic data structures for our algorithms and their relationships with the data structures of the algorithms by Hell et al. In Section 6.3 we develop an algorithm to find all the co-components of a PCA graph when a vertex is inserted or removed. The incremental algorithm for inserting a vertex into a PCA graph appears in Section 6.4. Next, in Section 6.5, we describe a decremental algorithm that removes vertices from a PCA graph, and a fully dynamic algorithm that combines vertex insertions with vertex removals. The algorithms to insert and remove edges from PHCA graphs are described in Sections 6.6 and 6.7, respectively. Finally, in Section 6.8 we discuss the connectivity data structure that is required by the fully dynamic algorithm.

6.1 The DHH and HSS algorithms: an overview

In this section we briefly review the algorithm by Deng et al. [DHH96] and the algorithms by Hell et al. [HSS01]. By Corollary 2.4.5, twin vertices can be ignored by any static recognition algorithm. This is not the case when the graph is dynamic, because the insertion and removal of vertices generate new twin vertices. Nevertheless, it is convenient to treat every maximal set of twin vertices as if they were just one vertex of the

[1] The vertex-only decremental algorithm for PCA graphs is trivial, it can be done by removing one arc of a PCA model in $O(1)$ time.

graph. This is one of the key ideas of the DHH algorithm that is replicated in the HSS algorithms and in ours. We present here the definitions that are needed to treat the sets of vertices as if they were vertices.

Let G be a graph, $v \in V(G)$ and $B \subseteq V(G) \setminus \{v\}$. Say that v is *adjacent* to B when $B \cap N(v) \neq \emptyset$, while v is *co-adjacent* to B when $B \setminus N(v) \neq \emptyset$. In other words, v is adjacent to B if v has some neighbor in B, while it is co-adjacent if it has some non-neighbor in B. When v is adjacent to all the vertices in B, then v is *fully adjacent* to B. Similarly, when v is not adjacent all the vertices in B, then v is *not adjacent* to B. Observe that v is fully adjacent to B if and only if v is not co-adjacent to B.

Define a *semiblock* to be a set of twin vertices, and a *block* to be a maximal semiblock. A partition of $V(G)$ into semiblocks is a *semiblock partition* of G. The special semiblock partition defined by the blocks of G is called the *block partition* of G. For a semiblock partition $\mathcal{B}$, we define the *$\mathcal{B}$-reduction* of G as the graph obtained by identifying each semiblock $B \in \mathcal{B}$ into a *representative vertex* v. Observe that G can be decoded from its $\mathcal{B}$-reduction by inserting $|B| - 1$ copies of v for each semiblock $B \in \mathcal{B}$. When $\mathcal{B}$ is the block partition of G, the $\mathcal{B}$-reduction of G is called the *block-reduction* of G. We extend all the graph theoretical definitions about vertices of the $\mathcal{B}$-reduction of G to the corresponding semiblocks of $\mathcal{B}$. For instance, we say that two semiblocks B and B' are adjacent, call $N(B)$ to the family of adjacent semiblocks of B, say that a semiblock is universal, etc. We also extend all the graph theoretical definitions about sets of vertices to families of semiblocks. For instance, we may also say that a family of semiblocks $\mathcal{B}$ is a complete set or a co-bipartition of a graph, that some semiblocks induce a path, etc.

Recall that a PIG order of a graph G is a linear ordering $\phi = v_1, \ldots, v_n$ of $V(G)$ such that, for every $1 \leq i \leq n$, there exist two non-negative values l, r such that $N[v_i] = \textsc{Lin}[v_{i-l}, v_{i+r}]$, and both $\textsc{Lin}[v_{i-l}, v_i]$ and $\textsc{Lin}[v_i, v_{i+r}]$ are complete sets. By Theorem 3.3.2 and Lemma 2.4.4, it follows that the vertices of every block of G are consecutive in the PIG order. Thus, we can uniquely encode all the PIG orders of the vertices with one PIG order of the blocks of G. Let $\mathcal{B}$ be a semiblock partition of G and H be the $\mathcal{B}$-reduction of G. Suppose that $v_1, \ldots, v_k$ is a PIG order of H, where v_i is the representative vertex of $B_i \in \mathcal{B}$. Then, a PIG order of G can be obtained by inserting first the vertices of B_1 in any order, then the vertices of B_2 in any order, and so on. We call the ordering $\Phi = B_1, \ldots, B_k$ of $\mathcal{B}$ a *semiblock PIG order* of G, while if $\mathcal{B}$ is a block partition then Φ is also a *block PIG order* of G. To distinguish between semiblock PIG orders and PIG orders, we always denote the former by Capital Greek letters and the latter with lower case Greek letters.

In [DHH96], Deng et al. developed an incremental algorithm, which we call the *DHH algorithm*, for the connected PIG graph recognition problem. The dynamic representation that is maintained by the algorithm is a block PIG order $\Phi = B_1, \ldots, B_k$ of the input graph G. When a new vertex v is inserted into G, there are two possibilities. If v has some twin in some block, say B_p, then v is inserted into B_p and the algorithm

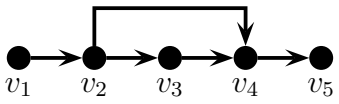
finishes. Otherwise, a new block B_p has to be created for v and a new block PIG order Ψ of $G \cup \{v\}$ has to be generated. Since the class of PIG graphs is hereditary, it follows that $\Phi' = \Psi \setminus \{B_p\}$ is a semiblock PIG order of G. By Theorem 3.3.2 and Lemma 2.4.4, Φ' is rather similar to Φ in the sense that Φ' is obtained from Φ just by splitting some blocks into consecutive semiblocks. Thus, Φ is modified just a little to obtain Ψ. In fact, v is adjacent and co-adjacent to at most two blocks of G, say B_l and B_r. Even more, v has to be fully adjacent to all the blocks in $\textsc{Lin}[B_{l+1}, B_{r-1}]$ and Ψ is equal to Ψ_l, Ψ_v, Ψ_r where

$$\begin{aligned}
\Psi_l &= \textsc{Lin}[B_1, B_{l-1}], B_l \setminus N(v) \\
\Psi_v &= B_l \cap N(v), \textsc{Lin}[B_{l+1}, B_h], B_p, \textsc{Lin}[B_{h+1}, B_{r-1}], B_r \cap N(v) \\
\Psi_r &= B_r \setminus N(v), \textsc{Lin}[B_{r+1}, B_k]
\end{aligned}$$

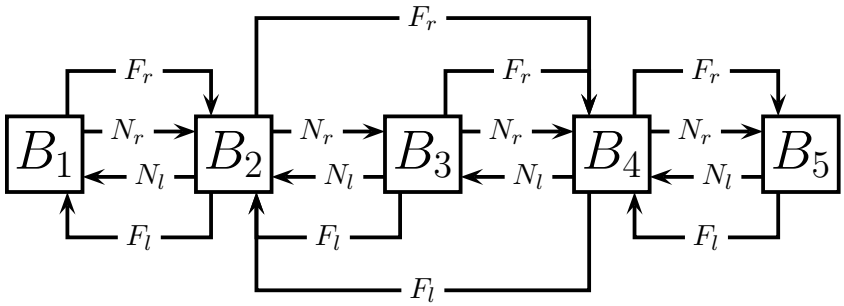
for some value h. Of course, there are other cases in which v is not adjacent to B_l and B_r, but this is the more general case. The DHH algorithm finds the position h and inserts B_p into Φ, whenever possible.

The implementation of the block PIG order $\Phi = B_1, \ldots, B_k$ in this algorithm is really simple (see Figure 6.1). There is doubly-linked list of blocks, where B_i has two near pointers $N_l^\Phi(B_i)$ and $N_r^\Phi(B_i)$ that respectively point to B_{i-1} and B_{i+1}, for every $1 < i < k$. In turn, every block B_i has two *far pointers* pointers $F_l^\Phi(B_i)$ and $F_r^\Phi(B_i)$ that point to the leftmost and rightmost neighbors of B_i in Φ. Finally, every vertex has a pointer to its block. When B_p is inserted as a new block into the PIG order, we have to partition the blocks B_l and B_r, and possibly change the far pointers of all the blocks in $B_l, \ldots, B_r$. All these operations are done in $O(d(v))$ time, *i.e.*, $O(1)$ time per edge insertion, which is optimal.

The DHH algorithm was extended by Hell et al. [HSS01] to handle the case in which the graph is not connected. In this case, G admits an exponential number of block PIG orders which can be constructed by permuting the block PIG orders of its components, and by reversing each connected block PIG order. To handle this situation, the *vertex-incremental HSS* algorithm keeps two block PIG orders for each component, one the reversal of the other. When a new vertex v is inserted, there are two possibilities. Either $N(v)$ is included in one component G_1 of G, or $N(v)$ intersects exactly two components G_1 and G_2 of G. In the former case, v is inserted into the block PIG orders of G_1 as in the DHH algorithm. In the latter case, G_1 and G_2 have to be combined into a new component and the four block PIG orders of G_1 and G_2 have to be replaced with two block PIG orders of $G_1 \cup G_2 \cup \{v\}$. Let $\Psi = B_1, \ldots, B_h, \{v\}, B_{h+1}, \ldots, B_k$ be a block PIG order of $G_1 \cup G_2 \cup \{v\}$. Since the class of PIG graphs is hereditary, we know that $B_1, \ldots, B_h, B_{h+1}, \ldots, B_k$ is a semiblock PIG order of $G_1 \cup G_2$. Even more, $B_1, \ldots, B_h$ is a semiblock PIG order of one of the components, $B_{h+1}, \ldots, B_k$ is a semiblock PIG order of the other component, and v is adjacent and co-adjacent to at most one block of G_1, and to at most one block of G_2.



A straight oriented graph.



The data structure implementing the PIG order v_1, v_2, v_3, v_4, v_5.

Figure 6.1: The PIG data structure for a small graph. The pointers from each vertex to its corresponding blocks are not shown.

Once G_1 and G_2 are determined and that the adjacent and co-adjacent blocks are known, it is not so hard to insert the new block for v. However, it is not easy to find G_1 and G_2 if the connected block PIG orders are represented as in the DHH algorithm. To find G_1 and G_2, the simplest way is to first locate the ranges of blocks adjacent to v. For this purpose, $N(v)$ is first traversed and the blocks adjacent to v are marked. Then, a connected block PIG order is traversed to the right and to the left, starting from a block B adjacent to v. The traversal stops either when a block not adjacent to v is found or when all the blocks in the PIG order have been traversed. The set of blocks traversed is a range of blocks, all of which are adjacent to v. In the case that v is adjacent to two ranges, then the graph is PIG only if these two ranges fall in block PIG orders of different components, and each of these blocks contains at least one of the extreme—leftmost or rightmost—blocks of the PIG order. To test if two ranges, both containing at least one extreme block, belong to the same component or not, an *end pointer* $E^\Phi(B)$ is stored for each block B. If B is not an extreme block, then E^Φ points to NULL, otherwise it points to the other extreme of the block PIG order (see Figure 6.2). With this new data structure, the HSS algorithm handles the insertion of a vertex in $O(d(v))$ time.

The vertex-incremental HSS algorithm can be adapted to allow the insertion of edges as well. Suppose that some edge vw is to be inserted into G. We consider here only the case in which G is connected. Let $\Phi = B_1, \ldots, B_k$ be the block PIG order of G and suppose that $v \in B_i$ and $w \in B_j$, for some $1 \leq i < j \leq k$. In G, the block B_i is adjacent to all the blocks in $\textsc{Lin}(B_i, F_r^\Phi(B_i))]$, and the block B_j is adjacent to all the blocks in $\textsc{Lin}[F_l^\Phi(B_j), B_j)$. In this case, for $G \cup \{vw\}$ to be a PIG graph, $F_r^\Phi(B_{i-1})$ has to point

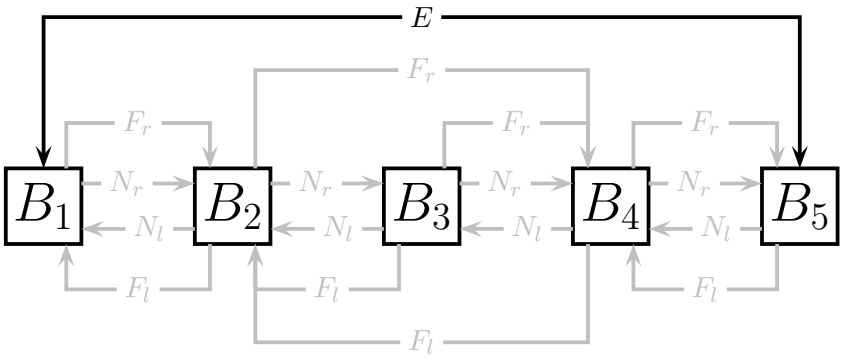


Figure 6.2: The data structure with end pointers for the graph in Figure 6.1.

to B_{j-1} and $F_l^\Phi(B_j)$ has to point to B_{i+1}. We have at least two possibilities for the insertion of the edge. Either v becomes a member of B_{i+1} or v gets separated from B_i to form a new block B_p that lies between B_i and B_{i+1}. In the latter case, the far pointers $F_r^\Phi(B)$ of all those blocks B pointing to B_i have to be updated so as to point to B_p.

To update all these far pointers to point to the new block B_p in $O(1)$ time, the HSS algorithm uses the technique of *nested pointers*. For each block B_i, two *self pointers* $S_l^\Phi(B_i)$ and $S_r^\Phi(B_i)$ that point to B_i are stored. Every far pointer $F_l^\Phi(B)$ that was pointing to B_i now points to $S_l^\Phi(B_i)$. Similarly, every far pointer $F_r^\Phi(B)$ that was pointing to B_i now points to $S_r^\Phi(B_i)$ (see Figure 6.3). To indicate that all the far pointers $F_r^\Phi(B)$ that were pointing to B_i now point to B_p, we only need to exchange the value of $S_r^\Phi(B_i)$ so as to point to B_p.

Up to this point we have discussed the incremental algorithms for the PIG graph recognition problem. The decremental algorithms for the removal of vertices and edges are similar to the incremental ones. However, the end pointers have to be removed from the data structure that is used to represent a block PIG order. This is because when two components result from the removal of a vertex or an edge, the new end pointers cannot be computed efficiently. On the other hand, without the end pointers, a vertex v can be removed in $O(d(v))$ time, while an edge vw can be removed in $O(1)$ time.

Finally, Hell et al. developed a fully dynamic recognition algorithm in where insertions and removals of vertices and edges are unrestricted. The algorithm is simply the combination of the incremental and decremental algorithms that we described above. However, there is an incompatibility with respect to the use of the end pointers. They are needed by the incremental algorithm to test whether two vertices belong to the same component, while they are harmful for the decremental algorithm. To solve this problem, Hell et al. propose the use of a *dynamic connectivity structure* that supports an operation to test if two vertices belong to the same component and to update the components in $O(\log n)$

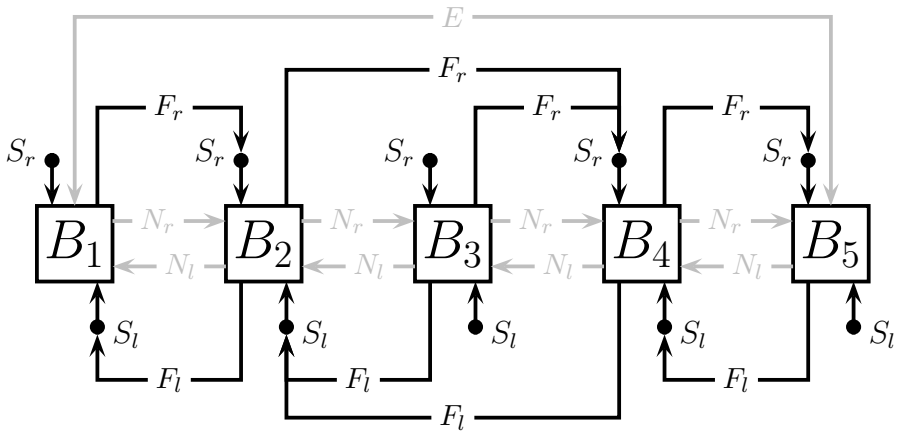


Figure 6.3: The data structure with self pointers for the graph in Figure 6.1. Now, every far pointer points to a self pointer.

Operation	No connectivity	End pointers	Connectivity structure
Vertex insertion	not allowed	$O(d(v))$	$O(d(v) + \log n)$
Edge insertion	not allowed	$O(1)$	$O(\log n)$
Vertex removal	$O(d(v))$	not allowed	$O(d(v) + \log n)$
Edge removal	$O(1)$	not allowed	$O(\log n)$

Table 6.1: Time complexities of the HSS algorithms.

time per update of the graph.

Table 6.1 summarizes the time complexities of the HSS algorithms. Each column of the table indicates the data structure that is implemented by the dynamic algorithm. No connectivity means that there is no way to test if two vertices belong to the same component. End pointers indicates that there is one end pointer for each block of the block PIG orders. Finally, the connectivity structure means that there is a dynamic structure to test if any two vertices belong to the same component or not.

6.2 The data structure

In the previous section we saw that three different data structures are used by the HSS algorithms. There is one with end pointers for the incremental algorithm, one with no support for connectivity queries for the decremental algorithm, and one with a connectivity structure for the fully dynamic algorithm. We will extend these data

structures for our algorithms, so as to represent round enumerations instead of PIG orders.

As we have discussed in Chapter 3, PIG orders are the same as straight enumerations of the straight orientations of the graph. Thus, we can think that the data structures of the HSS algorithms represent straight "block" enumerations as well. In the PCA recognition algorithm, the data structure represents a round enumeration of blocks, not a PCA order. This is because round enumerations are not the same as PCA orders when the graph has universal vertices (see Theorem 3.3.15). Nevertheless, we will use the same data structure as for PIG orders to specify the orientation; vertex v has a directed edge to w if w appears between v and the right far pointer of v. We formalize this idea in the following paragraph.

Recall that a round enumeration of a round digraph D is a circular ordering $\phi = v_1, \ldots, v_n$ of $V(D)$ such that, for every vertex v_i, $N^-[v_i] = [v_{i-l}, v_i]$ and $N^+[v_i] = [v_i, v_{i+r}]$, where $l = d^-(v_i)$ and $r = d^+(v_i)$. In Chapter 3, we defined the PCA orders so that there is no need to specify the round orientation of a non-universal PCA graph. In this chapter we cannot get rid of the orientation, because of the universal vertices. Say that $\phi = v_1, \ldots, v_n$ is a *round enumeration* of a graph G if ϕ is a round enumeration of some orientation D of G. For the sake of notation, when ϕ is a locally straight enumeration of D, then we say that ϕ is a *locally straight enumeration* of G. Similarly, if ϕ is a straight enumeration of D, then we say that ϕ is also a *straight enumeration* of G. That is, in this chapter we call locally straight enumerations to PHCA orders and straight enumerations to PIG orders. Although we need the round orientation D, it can be encoded with a round enumeration ϕ and two values $f_l^\phi(v)$ and $f_r^\phi(v)$ for every vertex v, so that $v \longrightarrow w$ in D if and only if $w \in (v, f_r^\phi(v)]$ and $v \in [f_l^\phi(w), w)$. Whenever we take a round enumeration ϕ, we assume that the values of f_l^ϕ and f_r^ϕ are already defined. We will write $v \longrightarrow_\phi w$ when $w \in (v, f_r^\phi(v)]$ and $v \not\longrightarrow_\phi w$ when $w \notin [v, f_r^\phi(v)]$. When there is no confusion about ϕ we will omit the subscripts and superscripts.

A word of caution is required before we continue. The notation $\longrightarrow_\phi$ was already used in Chapter 3 to denote the implicit orientation in PHCA, PCA, and NHCA orders. In this chapter we use $\longrightarrow_\phi$ for round enumerations of graphs, which are not precisely PCA orders. Nevertheless, by Theorem 3.3.15 there is a one-to-one mapping between PCA orders and round enumerations of universal-free graphs. In fact, $\longrightarrow_\phi$ can be thought of as an orientation for those graphs with no universal vertices, thus the meaning was not changed. Furthermore, we have never used $\longrightarrow_\phi$ for graphs with universal vertices before.

As Deng et al. and Hell et al., we have to deal with twin vertices as if they were a unit. Let $\mathcal{B} = B_1, \ldots, B_k$ be a semiblock partition of G and suppose that $v_1, \ldots, v_k$ is a round enumeration of the $\mathcal{B}$-reduction of G, where v_i is the representative of B_i. We say that $\Phi = B_1, \ldots, B_k$ is a *round semiblock enumeration* of G, while Φ is a *round block*

enumeration of G if $\mathcal{B}$ is a block partition of G. These definitions are extended to locally straight and straight enumerations as usual. As before, we will use Capital Greek letters for round semiblock enumerations. We also use Capital letters to denote f_l and f_r for semiblocks. That is, if v is the representative vertex of B, we denote by $F_l(B)$ and $F_r(B)$ the semiblocks that are represented by $f_l(v)$ and $f_r(v)$ in the $\mathcal{B}$-reduction, respectively.

Hell et al. introduce the concept of a contig[2] to deal with connected straight block enumerations. For a connected graph G, a *semiblock contig* is a straight semiblock enumeration of G, while a *contig* is a semiblock contig whose semiblocks are also blocks. A semiblock contig contains two special semiblocks: the *leftmost end semiblock* is the semiblock B such that $F_l(B) = B$, and the *rightmost end semiblock* is the semiblock B such that $F_r(B) = B$. We always assume that B_1 and B_k are respectively the leftmost and rightmost end semiblocks of the semiblock contig $\Phi = B_1, \ldots, B_k$. The generalizations of contigs to round enumerations are called rings. A *semiblock ring* is a round semiblock enumeration of the connected graph G, while a *ring* is a semiblock ring whose semiblocks are also blocks.

The data structure that we use to represent a semiblock ring $\Phi = B_1, \ldots, B_k$ is almost the same as the one used by the HSS algorithm for contigs. The main difference is that *near pointers* now represent a doubly-linked circular list instead of a doubly-linked list. We also add a *PHCA flag* that is true if and only if Φ is locally straight. Summing up, the incremental algorithm will maintain the following data to represent Φ for each semiblock B_i:

1. The vertices that compose B_i.
2. Left and right *near pointers*, $N_l^\Phi(B_i)$ and $N_r^\Phi(B_i)$. If B_i is not the leftmost end semiblock, then $N_l^\Phi(B_i)$ points to B_{i-1}; otherwise, $N_l^\Phi(B_i)$ points to B_i. Similarly, if B_i is not the rightmost end semiblock, then $N_r^\Phi(B_i)$ points to B_{i+1}; otherwise, $N_r^\Phi(B_i)$ points to B_i.
3. Left and right *self pointers*, $S_l^\Phi(B_i)$ and $S_r^\Phi(B_i)$, pointing to B_i.
4. Left and right *far pointers*, $F_l^\Phi(B_i)$ and $F_r^\Phi(B_i)$, pointing to the self pointers of $F_r(B_i)$ and $F_l(B_i)$, respectively.
5. An *end pointer* $E^\Phi(B_i)$, pointing to $NULL$ if B_i is not an extreme semiblock of a connected PIG order, or pointing to the other extreme semiblock otherwise.

As usual, we omit the superscript Φ when no confusions arise. The overloaded notation for F_l and F_r as pointers and blocks is intentional. We also overload the notation N_l and N_r to represent the blocks to the left and the right of some block. Observe that $N_l(B) = F_l(B) = B$ whenever B is the leftmost end semiblock and $N_r(B) = F_r(B) = B$ whenever B is the rightmost end semiblock.

[2]Proper interval graphs are related to DNA sequences

Operation	No connectivity	End pointers	Connectivity structure
Vertex insertion	not allowed	$O(d(v))$	$O(d(v) + \log n)$
Edge insertion PCA	not allowed	$O(\Delta(G))$	$O(\Delta(G) + \log n)$
Edge insertion PHCA	not allowed	$O(1)$	$O(\log n)$
Vertex removal	$O(d(v))$	not allowed	$O(d(v) + \log n)$
Edge removal PCA	$O(\Delta(G))$	not allowed	$O(\Delta(G) + \log n)$
Edge removal PHCA	$O(1)$	not allowed	$O(\log n)$

Table 6.2: Time complexities of the dynamic recognition algorithms for PCA and PHCA graphs.

We will represent each component of a dynamic PCA graph G with two rings, one the reverse of the other. The algorithms will assure that, after each update, the end blocks are always the first and last blocks of the ring's implementation. Of course, we will assume that this condition holds for the semiblock rings as well. So, we can test whether a ring of G is a contig by querying whether its first block is an end block or not. All the rings that represent a component of G are stored in one set that represents a round block enumeration of G. The algorithms will also assure that these rings are contigs when G is a PIG graph, while the rings are locally straight when G is a PHCA graph. So, we can apply the HSS algorithm without modifications when G is a PIG graph and the dynamic PHCA recognition problem is solved with the same algorithm as the PCA recognition problem.

Following the ideas by Hell et al., the data structure for the decremental algorithms is the same as the one for the incremental algorithm, with the exception that end pointers are removed. Similarly, the data structure for the fully dynamic algorithms is obtained by replacing the end pointers with a connectivity data structure. The connectivity data structure is described in Section 6.8.

Table 6.2 is a preview of the time complexities of the algorithms. The columns of this table have the same meaning as the corresponding columns in Table 6.1.

6.3 Co-components of PCA graphs

The incremental connected PIG recognition algorithm by Deng et al. takes advantage of the fact that every connected PIG graph admits a unique PIG model, up to full reversal. For the dynamic recognition of general PIG graphs, Hell et al. have to deal with each component in a separate way, since the components can be permuted to form several PIG orders. For PCA graphs, the situation is similar. Huang [Hua95] proved that every connected and co-connected PCA graph admits a unique round enumeration, up to full reversal (see Theorem 3.3.3). However, if G is not co-connected, then its co-components

can be permuted so as to form several round enumerations. Instead of dealing with these co-components in the data structure, we take a more lazy approach: we compute all the co-components only when they are needed. The advantage of this approach is that it allows us to use the same data structure as Hell et al. The disadvantage is that we have to find all the co-components fast.

To begin this section, we show how to compute the co-component $\mathcal{C}$ that contains a semiblock B in $O(d_{\mathcal{C}}(B))$ time, when a semiblock ring of a graph G is given as input. Here $d_{\mathcal{C}}(B) = |N(B) \cap \mathcal{C}|$ is the number of neighbors of B that belong to $\mathcal{C}$. The solution to this problem yields an $O(\Delta(G))$ time algorithm that computes all the co-components of the graph. This is not enough to assure that only $O(d(v))$ time is spent to compute the co-components, when v is to be inserted into G. We will show how to do this in Section 6.3.1. The following proposition, that follows from Theorem 2.4.10, is essential for our purposes.

Proposition 6.3.1. *If a PCA graph is not co-connected, then it is co-bipartite.*

Algorithm 6.1 outputs the co-component of a co-bipartite graph G that contains B. The correctness of the algorithm is given in the following proposition.

Proposition 6.3.2. *Let G be a co-bipartite graph and let $\mathcal{X}, \mathcal{Y}$ be the families in Algorithm 6.1 at some step of the execution. Then, $\mathcal{X} \cup \mathcal{Y}$ is co-connected, $\mathcal{X} \cap \mathcal{Y} = \emptyset$ and $B \in \mathcal{X}$. Moreover, when Algorithm 6.1 finishes, $\langle \mathcal{X}, \overline{N}(\mathcal{X}) \rangle$ is a co-bipartition of a co-component of G and $B \in \mathcal{X}$.*

Algorithm 6.1 Co-bipartition of the co-component containing B.

Input: A co-bipartite graph G with a semiblock partition $\mathcal{B}$, and $B \in \mathcal{B}$.
Output: A co-bipartition $\langle \mathcal{X}, \mathcal{Y} \rangle$ of the co-component that contains B, where $B \in \mathcal{X}$.

1. Set $\mathcal{X} := \{B\}$ and $\mathcal{Y} := \emptyset$.

2. Perform the following operations while $\mathcal{Y} \neq \overline{N}(\mathcal{X})$.

3. Set $\mathcal{Y} := \overline{N}(\mathcal{X})$.

4. Set $\mathcal{X} := \overline{N}(\mathcal{Y})$.

5. Output $\langle \mathcal{X}, \mathcal{Y} \rangle$.

Let $\Phi = B_1, \ldots, B_k$ be a round semiblock enumeration of a graph G. A range $[B_l, B_r]$ of Φ is a *co-bipartite chunk* if $[B_l, B_r] \subseteq \mathcal{X}$ for some co-bipartition $\langle \mathcal{X}, \overline{N}(\mathcal{X}) \rangle$ of a co-component of G. A maximal co-bipartite chunk is called a *co-bipartite range*. The next proposition shows how do $\mathcal{X}$ and $\mathcal{Y}$ look like at each step of the algorithm.

Proposition 6.3.3. *Let Φ be a round semiblock enumeration of a co-bipartite graph G, and $\mathcal{X} = [B_l, B_r]$ be a co-bipartite chunk of Φ. Then $\overline{N}(\mathcal{X}) = (F_r(B_l), F_l(B_r))$.*

Proof. Let $\Phi = B_1, \ldots, B_k$ and call $B_a = F_r(B_l)$ and $B_b = F_l(B_r)$. Suppose, to obtain a contradiction, that $B_l \neq B_r$ and $B_l \not\longrightarrow B_r$. In this case $B_r \longrightarrow B_l$ because $\mathcal{X}$ is a complete set. Hence, $B_r \longrightarrow B_j$ for every $B_j \in (B_r, B_l]$ which implies that B_r is universal. This is impossible because B_l and B_r belong to the same co-component by definition. Therefore, either $B_l = B_r$ or $B_l \longrightarrow B_r$. Consequently, $B_r \in [B_l, B_a]$ which implies that $B_i \longrightarrow B_j$ for every $B_i \in [B_l, B_r]$ and every $B_j \in [B_r, B_a]$. A similar argument can be used to prove that $B_j \longrightarrow B_i$ for every $B_i \in [B_l, B_r]$ and every $B_j \in [B_b, B_l]$. Then, all the semiblocks in $[B_b, B_a]$ belong to $N[B_i]$ for every $B_i \in \mathcal{X}$, thus $\overline{N}(X) \subseteq (B_a, B_b)$.

For the other inclusion, suppose that there is some semiblock $B \in (B_a, B_b)$ that is adjacent to all the semiblocks in $\mathcal{X}$. This implies that B_l and B_r are not universal since otherwise $(B_a, B_b) = (F_r(B_l), F_l(B_l)) = \emptyset$. Hence, B_{a+1} is not adjacent to B_l and B_{b-1} is not adjacent to B_r, so $B \notin \{B_{a+1}, B_{b-1}\}$. Consequently, $[B_{a+1}, B)$ and $(B, B_{b-1}]$ are nonempty ranges. In particular, both B_{a+1} and B_{b-1} belong to $\overline{N}(\mathcal{X})$, thus there is a path between B_{a+1} and B_{b-1} in $\overline{G}$. Such path must contain three blocks B_x, B_y, B_z such that B_y is not adjacent to B_x, B_x is not adjacent to B_z, $B_x \in \mathcal{X}$, $B_y \in [B_{a+1}, B)$, and $B_z \in (B, B_{b-1}]$. By hypothesis, either $B \longrightarrow B_x$ or $B_x \longrightarrow B$. The former is impossible because $B_z \not\longrightarrow B_x$, while the latter is impossible because $B_x \not\longrightarrow B_y$. □

Corollary 6.3.4. *Let Φ be a round semiblock enumeration of a co-bipartite graph. Then, at each step of Algorithm 6.1, the variable $\mathcal{X}$ is a co-bipartite chunk and the variable $\mathcal{Y}$ is either empty or a co-bipartite chunk.*

Proof. Observe that if $\mathcal{X}$ is a co-bipartite chunk, then either $\overline{N}(\mathcal{X}) = \emptyset$ or $\overline{N}(\mathcal{X})$ is a nonempty range by Proposition 6.3.3. In the former case $\mathcal{X} = \{B\}$ for some universal block B while in the latter case $\overline{N}(\mathcal{X})$ is a co-bipartite chunk. If $\overline{N}(\mathcal{X})$ is a co-bipartite chunk, then $\overline{N}(\overline{N}(\mathcal{X})) \neq \emptyset$ is also a co-bipartite chunk by Proposition 6.3.3. Therefore, since $\mathcal{X}$ is a co-bipartite chunk before the main loop of Algorithm 6.1, we obtain that $\mathcal{X}$ and $\mathcal{Y}$ are both co-bipartite chunks after every step of the main loop of Algorithm 6.1. □

The above propositions show how can we compute Algorithm 6.3.1 for co-bipartite PCA graphs when a round semiblock enumeration $\Phi = B_1, \ldots, B_k$ is given. We only need to maintain the co-bipartite chunk $\mathcal{X} = [B_l, B_r]$, compute the co-bipartite chunk $\mathcal{Y} = (F_r(B_l), F_l(B_r))$, and then compute $\overline{N}(\mathcal{Y})$. The computation of $\mathcal{Y}$ is easy with the information stored in our dynamic data structure, once B_l and B_r are given. On the other hand, to compute $\overline{N}(\mathcal{Y})$ we need to have access to the leftmost and rightmost semiblocks in $\mathcal{Y}$. That is, we need to find the semiblock immediately to the right of $F_r(B_l)$ and the semiblock immediately to the left of $F_l(B_r)$. When $F_r(B_l)$ is an end

semiblock of Φ, the semiblock to the right of $F_r(B_l)$ is the leftmost end semiblock of some contig of Φ. The dynamic data structure provides no efficient operation to obtain the rightmost end semiblock from a leftmost end semiblock. Of course, there could be some clever way to use the end pointer of a leftmost end semiblock to obtain its rightmost end semiblock in $O(1)$ time, but that would preclude the decremental problem from invoking this algorithm. To avoid these complications, we restrict ourselves to semiblock rings. For the semiblock ring $\Phi = B_1, \ldots, B_k$, define $Next(B_i) = B_{i+1}$ and $Prev(B_i) = B_{i-1}$. (Note that $Prev(B) \neq N_l(B)$ (resp. $Next(B) \neq N_r(B)$) when B is the leftmost (resp. rightmost) end semiblock of the ring.) The following proposition shows that $Prev$ and $Next$ can be efficiently computed without using any end pointers.

Proposition 6.3.5. *Let Φ be a semiblock ring of a co-bipartite graph G and $B \in \Phi$. If $N_r(B) \neq B$, then $Next(B) = N_r(B)$; otherwise, $Next(B) = F_l(F_l(F_l((B))))$. Similarly, if $N_l(B) = B$, then $Prev(B) = N_l(B)$; otherwise, $Prev(B) = F_r(F_r(F_r(B)))$.*

Proof. If $N_r(B) = B$ then B is the rightmost end semiblock of Φ, thus Φ is a contig and $Next(B)$ is the leftmost end semiblock B' of Φ. Let $B_1 = F_l(B)$, $B_2 = F_l(B_1)$, and $B_3 = F_l(B_2)$. If $B' \in \{B_1, B_2, B_3\}$, then $B_3 = B'$ because $F_l(B') = B'$. Otherwise, neither B nor B' are adjacent to B_2 which implies that B, B' and B_2 are pairwise non-adjacent, *i.e.*, G is not co-bipartite. The proof is analogous when $N_l(B) = B$. □

Algorithm 6.1 can be implemented so as to run in $O(d_{\mathcal{C}}(B))$ time, when Φ is a semiblock ring, as in Algorithm 6.2. Note that Algorithm 6.2 outputs just the co-bipartite range $\mathcal{X}$ that contains B. However, the other co-bipartite range of its co-component can be computed in $O(1)$ time by Propositions 6.3.3 and 6.3.5. Step 1 defines the functions $Prev$ and $Next$ as in Proposition 6.3.5. Step 3 checks whether B is a universal semiblock and, if so, it outputs the co-bipartite range $\mathcal{X} = [B, B]$. When B is not a universal semiblock, then no semiblock in $\mathcal{X} \cup \mathcal{Y}$ is universal throughout the execution of the algorithm. Therefore, $F_l(B_r)$ is not the semiblock that appears immediately to the right of $F_r(B_l)$. Hence, by Proposition 6.3.3, $\overline{N}(\mathcal{X})$ is equal to $(F_r(B_l), F_l(B_r))$ which, in turn, is equal to $[Next(F_r(B_l)), Prev(F_l(B_r))]$.

For the implementation, the co-bipartite chunks $\mathcal{X}$ and $\mathcal{Y}$ can be represented by a pair of pointers, pointing to the leftmost and rightmost semiblocks in the range. Of course, the empty range is implemented as a pair of pointers to $NULL$. Clearly, the $Next$ and $Prev$ functions run in $O(1)$ time. Therefore, the main loop takes $O(d_{\mathcal{C}}(B))$ time, because at least one neighbor of B is inserted into $\mathcal{X}$ at each iteration. Moreover, the main cycle of Algorithm 6.2 is executed for at most $d_{\mathcal{C}}(B)$ times.

To invoke Algorithm 6.2 we first need to make sure that the input semiblock ring corresponds to a co-bipartite graph. Of course, Algorithm 6.2 can be executed on semiblock rings of graphs that are not co-bipartite. However, the algorithm fails to terminate in this case, unless an end semiblock is found.

Algorithm 6.2 Co-component of B in a semiblock ring.
Input: A semiblock ring Φ of a co-bipartite graph G, and a semiblock $B \in \Phi$.
Output: The co-bipartite range $\mathcal{X}$ that contains B.

1. Define the function $Next : \Phi \to \Phi$ such that $Next(B) = N_r(B)$ when $N_r(B) \neq B$, and $Next(B) = F_l(F_l(F_l(B)))$ when $N_r(B) = B$. Similarly, define $Prev : \Phi \to \Phi$ such that $Prev(B) = N_l(B)$ when $N_l(B) \neq B$, and $Prev(B) = (F_r(F_r(F_r(B)))$ when $N_l(B) = B$.
2. Set $\mathcal{X} := [B, B]$, $\mathcal{Y} := \emptyset$.
3. If $Next(F_r(B)) = F_l(B)$ then output $\mathcal{X}$.
4. Define the function $\overline{N}([B_l, B_r]) := [Next(F_r(B_l)), Prev(F_l(B_r))]$.
5. Perform the following operation while $\mathcal{Y} \neq \overline{N}(\mathcal{X})$.
6. Set $\mathcal{Y} := \overline{N}(\mathcal{X})$.
7. Set $\mathcal{X} := \overline{N}(\mathcal{Y})$.
8. Output $\mathcal{X}$.

Proposition 6.3.6. *If Algorithm 6.2 terminates with output $[B_l, B_r]$ when it is applied to a semiblock ring Φ of a graph G, then either G is a co-bipartite graph or one among $R = F_r(F_r(F_r(B_l)))$ and $L = F_l(F_l(F_l(B_r)))$ is an end semiblock.*

Proof. Suppose that Algorithm 6.2 terminates with output $[B_l, B_r]$ when it is applied to the semiblock ring $\Phi = B_1, \ldots, B_k$ and the block $B \in \Phi$. If B is universal then G is co-bipartite by Proposition 6.3.1. Suppose then that B is not universal, so the main loop of the algorithm is executed for at least one iteration. Let $[B_a, B_b]$ be the range corresponding to variable $\mathcal{Y}$ after the last iteration of the main loop is executed. By definition, $B_l = Next(F_r(B_a))$ and $B_r = Prev(F_l(B_b))$. By the loop stop condition, $B_a = Next(F_r(B_l))$ and $B_b = Prev(F_l(B_r))$. Here $Prev$ and $Next$ are the functions defined as in Algorithm 6.2, *i.e.*, $Next(B') = F_l(F_l(F_l(B')))$ when B' is the rightmost end semiblock, and $Prev(B') = F_r(F_r(F_r(B')))$ when B' is the leftmost end semiblock. If $F_r(B_a)$ is the rightmost end semiblock B', then $B_l = F_l(F_l(F_l(B')))$, thus R is the end semiblock B'. Similarly, if $F_l(B_b)$ is the leftmost end semiblock, then L is an end semiblock. On the other hand, if $F_r(B_l)$ is an end semiblock, then $R = F_r(B_l)$, and thus it is an end semiblock. Similarly, L is an end semiblock when $F_l(B_r)$ is an end semiblock. Finally, if none of $F_r(B_a)$, $F_l(B_b)$, $F_r(B_l)$, and $F_l(B_r)$ are end semiblocks, then $B_{l-1} = F_r(B_a)$ and $B_{a-1} = F_r(B_l)$ (see Figure 6.4). So, every semiblock of Φ belongs to $[B_l, F_r(B_l)] \cup [B_a, F_r(B_a)]$, *i.e.*, $\langle [B_l, F_r(B_l)], [B_a, F_r(B_a)] \rangle$ is a co-bipartition of G. □

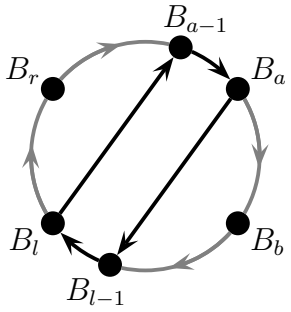


Figure 6.4: Proof of Proposition 6.3.6.

When Algorithm 6.2 finishes with input Φ, we can check whether G is co-bipartite or not as in the following proposition.

Proposition 6.3.7. *Let $\Phi = B_1, \ldots, B_k$ be a semiblock contig of a graph G. Then G is co-bipartite if and only if $B_1 = F_l(N_l(F_l(B_k)))$ and $B_k = F_r(N_r(F_r(B_1)))$.*

Proof. Suppose first that G is co-bipartite. If $N_l(F_l(B_k)) \neq B_1$, then $N_l(F_l(B_k))$ is not adjacent to B_k, thus $N_l(F_l(B_k))$ must be adjacent to B_1. That is, $B_1 = F_l(N_l(F_l(B_k)))$. Similarly, either $N_r(F_r(B_1)) = B_k$ or $F_r(N_r(F_r(B_1))) = B_k$.

For the converse, observe that both $\mathcal{X} = [B_1, F_r(B_1)]$ and $\mathcal{Y} = [F_l(B_k), B_k]$ are complete sets, and that all the semiblocks in Φ belong to at least one of $\mathcal{X}$ or $\mathcal{Y}$ because $N_r(F_r(B_1)) \in [F_l(B_k), B_k]$. □

The above propositions allow us to compute the co-component that contains B even when G is not co-bipartite, as follows. First run Algorithm 6.2 for at most $d(B)$ iterations. If the algorithm fails to terminate, then G is co-connected and not co-bipartite, by Proposition 6.3.1. Otherwise, we obtain a range $\mathcal{X}$ and we can check whether G is co-bipartite by combining Propositions 6.3.1 and 6.3.7.

The following proposition is useful to describe the number of neighbors that a vertex v of a graph H has in a semiblock ring of $H \setminus \{v\}$.

Proposition 6.3.8. *Let Φ be a semiblock ring of a co-bipartite graph G and $B \in \Phi$. If the main loop of Algorithm 6.2 is executed for p iterations when it is applied to Φ and B then $\overline{G}$ contains an induced path $B_1, \ldots, B_{2p}$, where $B = B_1$.*

Proof. The proposition is clearly true for $p = 0$. Suppose then that $p > 0$, and call $L_0 = R_0 = B$, $L_{i+1} = Next(F_r(L_i))$, and $R_{i+1} = Prev(F_l(R_i))$, for $0 \leq i \leq p$. We will prove that either $L_0, \ldots, L_{2p-1}$ or $R_0, \ldots, R_{2p-1}$ is an induced path of $\overline{G}$.

By definition, variable $\mathcal{X}$ is equal to $\mathcal{X}_i = [L_{2i}, R_{2i}]$ and variable $\mathcal{Y}$ is equal to $\mathcal{Y}_i = [L_{2i-1}, R_{2i-1}]$ after the i-th iteration of the main loop of Algorithm 6.2. Recall that, by Corollary 6.3.4, $\mathcal{X}_i$ and $\mathcal{Y}_i$ are co-bipartite chunks, and $\mathcal{X}_{i-1} \subseteq \mathcal{X}_i$ and $\mathcal{Y}_{i-1} \subseteq \mathcal{Y}_i$. By the stop condition, $\mathcal{Y}_p = \overline{N}(\mathcal{X}_{p-1}) \neq \mathcal{Y}_{p-1}$, *i.e.*, $[L_{2p-1}, R_{2p-1}] \neq [L_{2p-3}, R_{2p-3}]$. Therefore, either $L_{2p-1} \neq L_{2p-3}$ or $R_{2p-1} \neq R_{2p-3}$. Assume, w.l.o.g., that $L_{2p-1} \neq L_{2p-3}$. Hence, $L_i \neq L_{i-2}$ for every $1 \leq i \leq 2p-1$. Since also $[L_{i-2}, R_{i-2}] \subset [L_i, R_i]$, we obtain that $L_{i+2j} \notin [L_i, R_i]$, for every $0 \leq i < i + 2j \leq 2p-1$.

Fix some pair of values i, j such that $0 \leq i < j \leq 2p-1$. If i and j have the same parity, then $L_i \in [L_j, R_j]$, thus L_i is adjacent to L_j. If $i = j-1$, then L_i is not adjacent to L_j because otherwise L_i would be universal. Finally, if $i \neq j-1$ and i and j have different parity, then $L_j \notin [L_{i+1}, R_{i+1}] = \overline{N}([L_i, R_i])$, so L_i is adjacent to L_j. Summing up, $B = L_0, \ldots, L_{2p-1}$ is an induced path of $\overline{G}$. □

To end this section, we show how to use Algorithm 6.2 so as to obtain all the co-components of a PCA graph together with its co-bipartitions. The input is a round semiblock enumeration Φ, and the output is Φ when Φ is not co-bipartite, or a list $\mathcal{X}_1, \ldots, \mathcal{X}_s$ of co-bipartite ranges otherwise. One interesting thing is that $\mathcal{X}_i \cup \mathcal{X}_{i+1}$ is a range of Φ for every $1 \leq i \leq s$. The whole procedure is summarized in Algorithm 6.3. Recall that the only disconnected co-bipartite graph is the graph formed by the union of two complete sets. So, the only round semiblock enumeration Φ of a disconnected co-bipartite graph has two contigs, each of which is a co-bipartite range. Steps 1 and 3 find these co-bipartite ranges. On the other hand, observe that Steps 4–6 are used to check whether G is co-bipartite when G is connected.

6.3.1 Co-components of an incremental PCA graph

In this part of the section we deal with the problem of finding all the co-components of a graph G when a vertex v is to be inserted into G. The key idea is to prove that, for $H = G \cup \{v\}$ to be a PCA graph, v has to be adjacent to a large number of vertices of G. Even more, v does not have non-neighbors in more than two non-universal co-components of G. We present an $O(d_H(v))$ time algorithm that, given a round semiblock enumeration Φ of G, outputs all the co-components of G when G is co-bipartite and H is PCA. When either G is not co-bipartite or H is not PCA, it halts with output Φ.

For a family $\mathcal{C}$ of subsets of $V(G)$, denote by $d_{\mathcal{C}}(v)$ the number of members of $\mathcal{C}$ that are adjacent to v. The next propositions show that v is of high degree when H is a PCA graph and G is co-bipartite.

Proposition 6.3.9. *Let v be a vertex of a PCA graph H such that $H \setminus \{v\}$ is co-bipartite, Φ be a semiblock ring of $H \setminus \{v\}$, $B \in \Phi$, and $\mathcal{C}$ be the co-component of $H \setminus \{v\}$ that*

Algorithm 6.3 Co-components of a PCA graph G

Input: A round semiblock enumeration $\Phi = B_1, \ldots, B_k$ of a graph G.
Output: If G is not co-bipartite, then Φ is the output. Otherwise, a list $\mathcal{X}_1, \ldots, \mathcal{X}_s$ with the co-bipartite ranges of Φ is obtained, where $\mathcal{X}_i$ and $\mathcal{X}_{i+1}$ is a range for every $1 \leq i \leq s$.

1. If Φ has three components, then output Φ.
2. If Φ is composed by two contigs $\Phi_1 = B_1, \ldots, B_j$ and $\Phi_2 = B_{j+1}, \ldots, B_k$, then:
3. If $F_r(B_1) = B_j$ and $F_r(B_{j+1}) = B_k$, then output $\{\Phi_1, \Phi_2\}$. Otherwise, output Φ.
4. Apply Algorithm 6.2 to Φ and $B \in \Phi$ for at most $2d(B)$ steps. If Algorithm 6.2 fails to terminate, then output Φ. Otherwise, a range $\mathcal{X}_1 := [B_l, B_r]$ is obtained.
5. If $R := F_r(F_r(F_r(B_l)))$ is an end semiblock and $F_l(N_l(F_l(R)))$ is not, then output Φ.
6. If $L := F_l(F_l(F_l(B_l)))$ is an end semiblock and $F_r(N_r(F_r(L)))$ is not, then output Φ.
7. Set $i := 1$ and define $Next$ as in Algorithm 6.2.
8. While $Next(B_r) \neq B_l$ then:
9. Apply Algorithm 6.2 to Φ and $Next(B_r)$, to obtain $\mathcal{X}_{i+1} := [Next(B_r), B_b]$.
10. Set $i := i + 1$ and $B_r := B_b$.
11. Output $\{\mathcal{X}_1, \ldots, \mathcal{X}_i\}$.

contains B. If the main loop of Algorithm 6.2 takes p iterations to terminate when it is applied to B and Φ, then $p \leq d_{\mathcal{C}}(v) + 2$.

Proof. Recall that the Φ-reduction of $H \setminus \{v\}$ is the graph obtained by contracting all the vertices of each semiblock B_i into one vertex v_i. Define G as the graph that is obtained from the insertion of $\{v\}$ into the Φ-reduction of $H \setminus \{v\}$, where v is adjacent to a vertex v_i if and only if v is adjacent to the corresponding semiblock B_i. Observe that G is an induced subgraph of H.

By Proposition 6.3.8, there is an induced path $B_1, \ldots, B_{2p}$ in $\overline{H \setminus \{v\}}$ where $B_i \in \mathcal{C}$ for every $1 \leq i \leq |\mathcal{C}|$. The semiblocks of this path correspond to some path $v_1, \ldots, v_{2p}$ of $\overline{G \setminus \{v\}}$. The proposition is trivially true when $p \leq 2$. Suppose, then, that $p \geq 3$. In this case, v_1, v_2, v_4 and v_5 induce a hole in G, thus v is adjacent to at least one of these vertices by Theorem 2.4.10. Let a be the minimum such that v_a is a neighbor of v. If $a \geq 4$ then v_1, v_2, v_a and v induce a $K_{1,3}$ in G, contradicting Theorem 2.4.10. Consequently $a \leq 3$.

If v is adjacent to v_i for every even $i > a$ or for every odd $i > a$, then the result follows.

Hence, suppose that v has two non-neighbors v_i and v_j in G, where $j - i > 0$ is odd and $i > a$. Of all the possible combinations, take i and j so that v is adjacent to v_h in G, for every $i < h < j$. By construction, $v_i, \ldots, v_j, v$ is an induced cycle of $\overline{G}$ with odd length, thus v_a cannot be adjacent to all these vertices in G by Theorem 2.4.10. Consequently, $i = a + 1$, and $h - j$ is even for every $j < h \leq p$ such that v_h is non-adjacent to v in G. Therefore, $p \leq d_{\mathcal{C}}(v) + 2$. □

Proposition 6.3.10. *Let v be a vertex of a graph H and $\mathcal{B}$ be a semiblock partition of $H \setminus \{v\}$. If H is a PCA graph then the non-universal semiblocks of $\mathcal{B}$ that are not adjacent to v lie in at most two co-components of $H \setminus \{v\}$.*

Proof. On the contrary, suppose that there are three non-universal semiblocks B_1, B_2, B_3 of $\mathcal{B}$ that lie in different co-components of $H \setminus \{v\}$. Call $B_{i+3} \in \mathcal{B}$ to a non-neighbor semiblock of B_i for $i \in \{1, 2, 3\}$. Let G be the graph with vertices $v_1, \ldots, v_6, v$, where v_i and v_j $(1 \leq i < j \leq 6)$ are adjacent if and only if B_i and B_j are adjacent, while v is adjacent to v_i if and only if v is fully adjacent to B_i $(1 \leq i \leq 6)$. Observe that G is an induced subgraph of H.

If v is adjacent to v_4, v_5 and v_6, then G is isomorphic to $\overline{H_5}$ in Figure 2.2, thus G is not a PCA graph by Theorem 2.4.10. If v is adjacent to v_i and not adjacent to v_j, for $i, j \in \{4, 5, 6\}$, then v_{j-3}, v_j, v and v_i induce a $K_{1,3}$ in G. Finally, if v is not adjacent to v_i and to v_j, for $4 \leq i < j \leq 6$, then $v_i, v_j, v_{i-3}, v_{j-3}, v$ induce a C_4 plus an isolated vertex, which is not PCA by Theorem 2.4.10. □

These propositions imply a lower bound condition for the degree of v in H, as it follows from the next corollary.

Corollary 6.3.11. *Let v be a vertex of a PCA graph H such that $H \setminus \{v\}$ is co-bipartite, Φ be a round semiblock enumeration of $H \setminus \{v\}$, and u be the number of universal semiblocks of Φ. Call s to the number of times that Algorithm 6.2 is invoked when Algorithm 6.3 is applied to Φ. For $i = 1, \ldots, s$, call p_i to the number of iterations that the main loop of Algorithm 6.2 requires for the i-th invocation. Then*

$$s + \sum_{i=1}^{s} p_i \leq u + 2d_H(v) + 12.$$

Proof. If $H \setminus \{v\}$ is disconnected, then $s = 0$ and the corollary is trivially true. When $H \setminus \{v\}$ is connected, a new co-bipartite range is found each time Algorithm 6.3 invokes Algorithm 6.2. Let $\mathcal{X}_1, \ldots, \mathcal{X}_s$ be the list of co-bipartite ranges of $G = H \setminus \{v\}$ that Algorithm 6.3 finds with these invocations. Without loss of generality, suppose that $\mathcal{X}_1, \ldots, \mathcal{X}_u$ contain universal semiblocks. Suppose also that v is adjacent to all the semiblocks of $G[\mathcal{X}_{u+1} \cup \ldots \cup \mathcal{X}_j]$, while v has co-adjacent semiblocks in $G[\mathcal{X}_{j+1}], \ldots, G[\mathcal{X}_s]$.

By definition, $p_i = 0$ for every $1 \leq i \leq u$, and $p_i \leq d_{\mathcal{X}_i}(v)$ for every $1 \leq i \leq j$. On the other hand, Proposition 6.3.9 implies that $p_i \leq d_{\mathcal{X}_i}(v) + 2$ for every $j < i \leq s$. Finally, by Proposition 6.3.10, $s - (j+1) \leq 4$. Therefore

$$s + \sum_{i=1}^{s} p_i = \sum_{i=1}^{s} (1 + p_i) \leq u + \sum_{i=u+1}^{j} 2d_{\mathcal{X}_i}(v) + \sum_{i=j+1}^{s} (2d_{\mathcal{X}_i}(v) + 3) \leq u + 2d_H(v) + 12.$$

□

Algorithm 6.4 takes a semiblock enumeration of H and finds all the co-components of $H \setminus \{v\}$, when H is a PCA graph and $H \setminus \{v\}$ is co-bipartite. The correctness of this algorithm follows from Corollary 6.3.11, and its time complexity is $O(d_H(v) + u)$ where u is the number of universal semiblocks of $H \setminus \{v\}$. Note that the algorithm may output the co-components of $H \setminus \{v\}$ even when H is not a PCA graph, and it may output Φ even when H is a PCA graph but $H \setminus \{v\}$ is not co-bipartite.

Algorithm 6.4 Co-components of a round semiblock enumeration

Input: A round semiblock enumeration Φ of a graph $H \setminus \{v\}$, and $d(v)$, for $v \in V(H)$.
Output: If H is a PCA graph and $H \setminus \{v\}$ is co-bipartite, then the output is a list with the co-bipartite ranges of Φ as in Algorithm 6.3. If the co-components are not found, then the algorithm answers Φ.

1. Set $s := p := 0$.
2. Apply Algorithm 6.3 while $s + p \leq 2d(v) + 12$. For each invocation of Algorithm 6.2 add 1 to s if the obtained co-bipartite range is not universal. Similarly, for each iteration of the main loop of Algorithm 6.2 add 1 to p.
3. If $s + p > 2d_H(v) + 12$ then output Φ. Otherwise, output the co-bipartite ranges obtained by Algorithm 6.3.

6.3.2 Round semiblock enumerations of co-bipartite graphs

In this last part of the section we present two algorithms that can be used to generate all the round semiblock enumerations of those co-bipartite graphs that contain at most one universal semiblock. These algorithms are based on the work by Huang [Hua92], who characterized all the round enumerations of reduced graphs. As we have seen in Chapter 3, the round enumerations with no universal vertices have some nice properties that are lost when universal vertices are permitted. Dealing with universal vertices is annoying and error-prone; round enumerations with universal vertices receive a special

treatment in [Hua92], and this section is not an exception to this "rule". Fortunately, the universal semiblocks can be removed easily from a round semiblock enumeration as follows.

Proposition 6.3.12. *Let $\Phi = B_1, \ldots, B_k$ be a round semiblock enumeration of a graph G and B_i be a universal semiblock of Φ. If $k > 1$, then $\Psi = B_1, \ldots, B_{i-1}, B_{i+1}, \ldots, B_k$ is a round semiblock enumeration of $G \setminus \{B_i\}$.*

Proof. Clearly Ψ is a round enumeration of the semiblocks of Φ that belong to $G \setminus \{B_i\}$. So, it is enough to show a round orientation of the semiblocks in Ψ. Define F_l^Ψ and F_r^Ψ for $B \in \Psi$ as follows.

- If $F_l^\Phi(B) \neq B_i$, then $F_l^\Psi(B) = F_l^\Phi(B)$; otherwise, $F_l^\Psi(B) = B_{i+1}$.
- If $F_r^\Phi(B) \neq B_i$, then $F_r^\Psi(B) = F_r^\Phi(B)$; otherwise, $F_r^\Psi(B) = B_{i-1}$.

Certainly, F_r^Ψ and F_l^Ψ are well defined for every $B \in \Psi$, and it is not hard to see that $B \longrightarrow_\Phi B'$ if and only if $B \longrightarrow_\Psi B'$ for every $B' \in \Psi$. □

To remove the universal semiblock B_i as in the above proposition, we could move $F_l(B)$ so as to point to B_{i+1} for every B such that $F_l(B) = B_i$. After that we could do the analogous operation with the far right pointers and then remove B_i. However, this is not a good idea since we would have to traverse many semiblocks of Φ. The other choice, using the ideas from [HSS01], is to move the self pointers of B_i. Recall that every semiblock B with $F_l(B) = B_i$ has its far pointer pointing to $S_l(B_i)$. Thus, by moving $S_l(B_i)$ so as to point to B_{i+1} we actually move all the far pointers in $O(1)$ time. The inconvenient with this approach is that all those semiblocks that where pointing to $S_l(B_{i+1})$ have to be updated so as to point to the new self pointer of B_{i+1}. However, this is not too much expensive, as it is shown in the following proposition.

Proposition 6.3.13. *Let $\Phi = B_1, \ldots, B_k$ be a round semiblock enumeration of a graph G, B_i be a universal semiblock of Φ, and $B \in \Phi \setminus \{B_i\}$. If $F_r(B_i) \neq B_i$, then*

(i) $F_l(B) = B_{i+1}$ if and only if B is universal and $B \in (F_r(B_i), F_r(B_{i+1})]$.

(ii) $F_r(B) = B_{i-1}$ if and only if B is universal and $B \in [F_l(B_{i-1}), F_l(B_i))$.

If $F_r(B_i) = B_i$, then G is a complete graph.

Proof. If $F_l(B) = B_{i+1}$, then $B_i \not\longrightarrow B$, which implies that $B \longrightarrow B_i$ and, therefore, B is universal. Moreover, $B \notin (B_i, F_r(B_i)]$ and $B \in [B_{i+1}, F_r(B_{i+1})]$, thus either $F_r(B_{i+1}) = F_r(B_i) = B_i$ and G is a complete graph, or $B \in (F_r(B_i), F_r(B_{i+1})]$. The converse of (i) is trivial, and (ii) follows analogously when $F_r(B) = B_{i-1}$. □

Algorithm 6.5 can be used to remove a universal semiblock from Φ. The algorithm shows only the case in which B_i is not an end semiblock, for the sake of simplicity. Steps 2–4 update the self pointers and far pointers as in Propositions 6.3.12 and 6.3.13. Once the far pointers are updated, the algorithm updates the near and end pointers. For this, observe that B_{i-1} is an end semiblock if and only if $F_r(B_{i-1}) = B_{i-1}$. When B_{i-1} is not an end semiblock then its near right semiblock is B_{i+1} and vice versa. When B_{i-1} is an end semiblock then we need to update the end pointers. Here we take advantage that the far pointers are already updated so as to obtain the other end semiblock of the component as in Proposition 6.3.7. Clearly, this algorithm runs in $O(u)$ time where u is the number of universal semiblocks.

Algorithm 6.5 Removal of a universal semiblock from a round enumeration Φ

Input: A round semiblock enumeration Φ with at least two semiblocks, and a universal block B_i.

Output: Φ is transformed into a round semiblock enumeration of $\Phi \setminus \{B_i\}$ and a round semiblock enumeration containing only B_i.

1. Define $Prev$ and $Next$ as in Algorithm 6.2 and set $B_{i-1} := Prev(B_i)$ and $B_{i+1} := Next(B_i)$.

2. For each $B \in (F_r(B_i), F_r(B_{i+1})]$, set $F_l(B_i) := S_l(B_i)$.

3. For each $B \in [F_l(B_{i-1}), F_l(B_i))$, set $F_r(B_i) := S_r(B_i)$.

4. Swap $S_l(B_{i+1})$ with $S_l(B_i)$ and $S_r(B_{i-1})$ with $S_r(B_i)$.

5. Set $N_r(B_i) := N_l(B_i) := E(B_i) := B_i$ and $F_r(B_i) := S_r(B_i)$ and $F_l(B_i) := S_l(B_i)$.

6. If $F_r(B_{i-1}) \neq B_{i-1}$ then set $N_r(B_{i-1}) := B_{i+1}$ and $N_l(B_{i+1}) := B_{i-1}$

7. Else:

8. Set $N_r(B_{i-1}) := B_{i-1}$ and $N_l(B_{i+1}) := B_{i+1}$.

9. Set $E(B_{i-1}) := F_l(F_l(F_l(B_{i-1})))$ and $E(B_{i+1}) := F_r(F_r(F_r(B_{i+1})))$.

10. Set $E(E(B_{i-1})) := B_{i-1}$ and $E(E(B_{i+1})) := B_{i+1}$.

From now on we deal with round semiblock enumerations that contain no universal semiblocks. We will show two algorithms that can be used together to generate all the round semiblock enumerations of a given graph. The first algorithm splits a round semiblock enumeration of a graph $G + H$ into two round semiblock enumerations, one of G and one of H. The second algorithm joins two round semiblock enumerations of graphs G and H into one round semiblock enumeration of $G + H$. We consider the split algorithm first.

Let Φ be a round semiblock enumeration of a graph G. Say that a range $\mathcal{X} \subseteq \Phi$ is

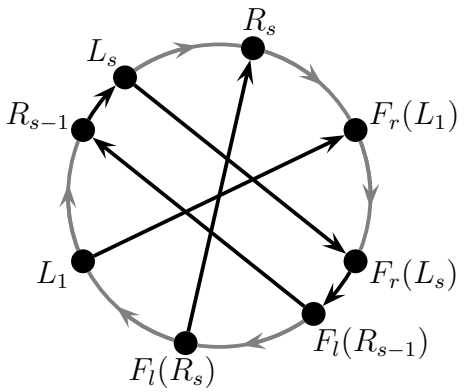


Figure 6.5: Proof of Proposition 6.3.14. If $F_r(L_s)$ is immediately to the left of $F_l(R_{s-1})$, then $\overline{\mathcal{X}} = (F_r(L_1), F_l(R_s))$.

a *range of co-components* if $\mathcal{X}$ is a complete set and it is the union of one or more co-bipartite ranges. In other words, $\langle \mathcal{X}, \overline{N}(\mathcal{X}) \rangle$ is a co-bipartition of a subgraph of G induced by several co-components. The following proposition, which follows from the work in [Hua92], is the generalization of Proposition 6.3.3 to ranges of co-components.

Proposition 6.3.14. *Let Φ be a round semiblock enumeration of a co-bipartite graph G and $\mathcal{X} = [B_l, B_r]$ be a range of co-components. If Φ contains no universal semiblocks, then $\overline{N}(\mathcal{X}) = (F_r(B_l), F_l(B_r))$.*

Proof. By definition, $\mathcal{X}$ is the union of s co-bipartite ranges $\mathcal{X}_1, \ldots, \mathcal{X}_s$. For $i = 1, \ldots, s$, let L_i and R_i be respectively the leftmost and rightmost semiblocks in $\mathcal{X}_i$, where $L_1 = B_l$, $R_s = B_r$, and $Next(R_i) = L_{i+1}$ if $i < s$. If $s = 1$, then the proposition follows from Proposition 6.3.3. For the other case, suppose that $s > 1$ and that the proposition is true for $[L_1, R_{s-1}]$. Observe that it is enough to prove that $\overline{N}(\mathcal{X}_s) = [F_l(R_{s-1}), F_l(R_s))$. By proposition 6.3.3, all we have to prove is that $F_r(L_s)$ is immediately to the left of $F_l(R_{s-1})$ (see Figure 6.5). Call B to the semiblock that is immediately to the left of $F_l(R_{s-1})$. Observe first that L_s is adjacent to all the semiblocks in $(L_s, F_r(L_1)]$ since $L_s \in (L_1, F_r(L_1)]$. Also, L_s is adjacent to all the semiblocks in $\overline{\mathcal{X}_i}$, for $1 \leq i < s$, because L_s does not belong to the co-component that contains $\mathcal{X}_i$. That is, L_s is adjacent to every block in $(L_s, B]$. Now, if $B \longrightarrow L_s$ then L_s is universal, a contradiction. Therefore, $L_s \longrightarrow B$, thus either $B = F_r(L_s)$ or $L_s \longrightarrow F_l(R_{s-1})$. The latter would imply that $B' \longrightarrow F_l(R_{s-1})$ for every $B' \in [L_s, F_l(R_{s-1}))$. But this is impossible because $F_l(R_{s-1}) \longrightarrow R_{s-1}$ and $F_l(R_{s-1})$ is not universal. Hence, $B = F_r(L_s)$. □

In view of the previous proposition, we will represent by $\overline{\mathcal{X}}$ the range $(F_r(B_l), F_l(B_r))$, for each range of co-components $\mathcal{X} = [B_l, B_r]$. The following proposition, which also follows from [Hua92], shows how can a round semiblock enumeration Ψ of $G[\mathcal{X} \cup \overline{N}(\mathcal{X})]$

be extracted from Φ. This proposition is rather similar to Proposition 6.3.12. However, instead of removing one co-component it allows the removal of several co-components at the same time. The main advantage of removing several co-components at the same time is that we also know how to obtain round semiblock enumeration of $G \setminus (\mathcal{X} \cup \overline{N}(\mathcal{X}))$ by symmetry.

Proposition 6.3.15. *Let $\Phi = B_1, \ldots, B_k$ be a round semiblock enumeration of a graph G and $\mathcal{X}$ be range of co-components of Φ. If Φ contains no universal semiblocks then $\Psi = \mathcal{X}, \overline{\mathcal{X}}$ is a round semiblock enumeration of $H = G[\mathcal{X} \cup \overline{N}(\mathcal{X})]$.*

Proof. By Proposition 6.3.14, Ψ is a round enumeration of the semiblocks of Φ that belong to H. So, it is enough to show a round orientation of the semiblocks in Ψ. Let $\mathcal{X} = [B_l, B_r]$ and $\overline{\mathcal{X}} = [B_a, B_b]$, and define F_l^Ψ and F_r^Ψ for $B \in \Psi$ as follows.

- If $F_l^\Phi(B) \in \mathcal{X} \cup \overline{\mathcal{X}}$, then $F_l^\Psi(B) = F_l^\Phi(B)$.
- If $F_r^\Phi(B) \in \mathcal{X} \cup \overline{\mathcal{X}}$, then $F_r^\Psi(B) = F_r^\Phi(B)$.
- If $B \in \mathcal{X}$ and $F_l^\Phi(B) = B_{b+1}$, then $F_l^\Psi(B) = B_l$.
- If $B \in \mathcal{X}$ and $F_r^\Phi(B) = B_{a-1}$, then $F_r^\Psi(B) = B_r$.
- If $B \in \overline{\mathcal{X}}$ and $F_l^\Phi(B) = B_{r+1}$, then $F_l^\Psi(B) = B_a$.
- If $B \in \overline{\mathcal{X}}$ and $F_r^\Phi(B) = B_{l-1}$, then $F_r^\Psi(B) = B_b$.

Observe that $F_r^\Phi(B_l)$ is immediately to the left of B_a, while $F_r^\Phi(B_r) \in [B_{a-1}, B_b]$. Therefore, $F_r^\Phi(B) \in [B_{a-1}, B_b]$ for every $B \in \mathcal{X}$. Similarly, $F_l^\Phi(B) \in [B_a, B_{b+1}]$ for every $B \in \mathcal{X}$, while $F_l^\Phi(B) \in [B_l, B_{r+1}]$ and $F_r^\Phi(B) \in [B_{l-1}, B_r]$ for every $B \in \overline{\mathcal{X}}$. Therefore, F_r^Ψ and F_l^Ψ are well defined, and it is not hard to see that $B \longrightarrow_\Phi B'$ if and only if $B \longrightarrow_\Psi B'$ for every, $B, B' \in \mathcal{X} \cup \overline{\mathcal{X}}$. □

To update the far pointers we are going to use the technique of nested pointers once again. The following proposition, which is the analogous of Proposition 6.3.13, shows that we can swap some self pointers in order to move all the far pointers efficiently.

Proposition 6.3.16. *Let $\Phi = B_1, \ldots, B_k$ be a round semiblock enumeration of a graph G and $\mathcal{X} = [B_l, B_r]$ be a range of co-components of Φ. If Φ has no universal semiblocks and $B_{l-1}, B_{r+1} \notin \overline{\mathcal{X}}$, then:*

(i) If $F_l(B) = B_l$, then $B \notin \mathcal{X} \cup \overline{\mathcal{X}}$ and

(ii) If $F_r(B) = B_r$, then $B \notin \mathcal{X} \cup \overline{\mathcal{X}}$.

Let $\mathcal{X} = [B_l, B_r]$ be a range of co-components of $\Phi = B_1, \ldots, B_k$, and suppose that $\mathcal{X}, \overline{\mathcal{X}} \neq \Phi$. Call H_1 and H_2 to $G[\mathcal{X} \cup \overline{\mathcal{X}}]$ and $G \setminus (\mathcal{X} \cup \overline{\mathcal{X}})$, respectively. When Φ contains no universal semiblocks, then, by Proposition 6.3.14, $\overline{\mathcal{X}} = [B_a, B_b]$ where B_a is the block immediately to the right of $F_r(B_l)$ and B_b is the block immediately to the left of $F_l(B_r)$.

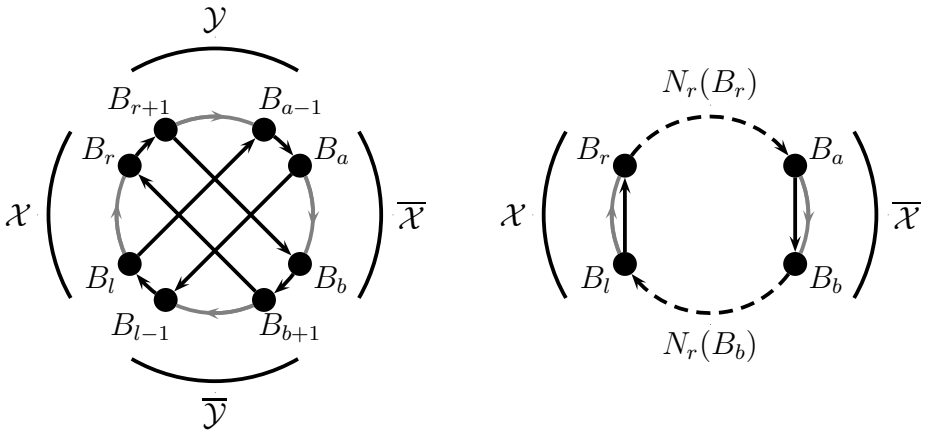


Figure 6.6: To the left, the orientation of the graph $G = H_1 + H_2$ with the round enumeration Φ is shown. To the right, the orientation of H_1 with the round enumeration $\mathcal{X}, \overline{\mathcal{X}}$, is shown. In H_1, either B_r (resp. B_b) is an end block or $N_r(B_r) = B_a$ (resp. $N_r(B_b) = B_l$).

Also, since $\Phi \neq \mathcal{X}, \overline{\mathcal{X}}$, there is some semiblock in (B_r, B_a) or in (B_b, B_l). Suppose that $(B_r, B_a) \neq \emptyset$, so the range $\mathcal{Y} = [B_{r+1}, B_{a-1}]$ is exactly (B_r, B_a). It turns out that $\mathcal{Y}$ is also a range of co-components. Indeed, $\mathcal{Y}$ is a complete set, and the only way to extend $\mathcal{Y}$ is by including some semiblock of $\mathcal{X} \cup \overline{\mathcal{X}}$. Furthermore, $\overline{\mathcal{Y}} = [B_{b+1}, B_{l-1}] = (B_b, B_l)$ because there are no universal semiblocks (see Figure 6.6). In other words, $H_2 = G[\mathcal{Y} \cup \overline{\mathcal{Y}}]$. So, by Proposition 6.3.14, $\Psi = \mathcal{X}, \overline{\mathcal{X}}$ is a round semiblock enumeration of H_1 and $\Gamma = \mathcal{Y}, \overline{\mathcal{Y}}$ is a round semiblock enumeration of H_2. Algorithm 6.6 can be used to split the graph G into H_1 and H_2. The input of the algorithm is Φ and the range of co-components $\mathcal{X}$. Note that Φ must be a semiblock ring without end semiblocks, since Φ has at least four co-bipartite ranges. To exchange all the far pointers in $O(1)$ time, we make use of the self pointers once again. Observe that, by Propositions 6.3.15 and 6.3.16, $F_l^\Phi(B) = B_l$ if and only if $F_l^\Gamma(B) = B_{r+1}$; $F_l^\Phi(B) = B_{r+1}$ if and only if $F_l^\Psi(B) = B_a$; $F_l^\Phi(B) = B_a$ if and only if $F_l^\Gamma(B) = B_{b+1}$; and $F_l^\Phi(B) = B_{b+1}$ if and only if $F_l^\Psi(B) = B_l$. So, the update of the left far pointers can be done as in Step 2. Finally, the end semiblocks are updated as in Algorithm 6.5. Clearly, this algorithm takes $O(1)$ time.

Proposition 6.3.15 can be used in the other direction, to obtain a round semiblock enumeration of G from a round semiblock enumeration of $G[\mathcal{X} \cup \overline{\mathcal{X}}]$ and a round semiblock enumeration of $G \backslash (\mathcal{X} \cup \overline{X})$. Suppose that $\Psi = B_l, \ldots, B_r, B_a, \ldots, B_b$ is a round semiblock enumeration of a co-bipartite graph H_1, where $\mathcal{X} = [B_l, B_r]$ is a range of co-components of Φ. Similarly, suppose that $\Gamma = B_{r+1}, \ldots, B_{a-1}, B_{b+1}, \ldots, B_{l-1}$ is round semiblock enumeration of the co-bipartite graphs H_2, where $\mathcal{Y} = [B_{a+1}, B_{b-1}]$ is a range of co-components of Γ. Define the enumeration $\langle \mathcal{X}, \mathcal{Y}, \overline{\mathcal{X}}, \overline{\mathcal{Y}} \rangle$ as the circular enumeration of blocks $\Phi = \mathcal{X}, \mathcal{Y}, \overline{\mathcal{X}}, \overline{\mathcal{Y}}$ where F_r^Φ and F_l^Φ are defined as below.

Algorithm 6.6 Split of a co-bipartite range

Input: A semiblock ring Φ, and a range of co-components $\mathcal{X} = [B_l, B_r]$ such that $\mathcal{X}, \overline{\mathcal{X}} \neq \Phi$.

Output: Φ is transformed into two round semiblock enumerations, one for $G[\mathcal{X} \cup \overline{\mathcal{X}}]$ and the other for $G \setminus (\mathcal{X} \cup \overline{\mathcal{X}})$.

1. Let $B_{l-1} := N_l(B_l)$, $B_{r+1} := N_r(B_r)$, $B_{b+1} := F_l(B_r)$, $B_{a-1} := F_r(B_l)$, $B_a := N_r(B_{a-1})$, and $B_b := N_l(B_{b+1})$.

//Self pointers

2. Simultaneously, set $S_l(B_{r+1}) := S_l(B_l)$, $S_l(B_a) := S_l(B_{r+1})$, $S_l(B_{b+1}) := S_l(B_a)$, and $S_l(B_l) := S_l(B_{b+1})$.

3. Simultaneously, set $S_r(B_{l-1}) := S_r(B_r)$, $S_r(B_b) := S_r(B_{l-1})$, $S_r(B_{a-1}) := S_r(B_b)$, and $S_r(B_r) := S_r(B_{a-1})$.

//Near and end pointers

4. Define the functions $F_l^3(B) = F_l(F_l(F_l(B)))$ and $F_r^3(B) = F_r(F_r(F_r(B)))$, for every $B \in \Phi$.

5. If $F_l(B_l) \neq B_l$, then set $N_l(B_l) := B_b$ and $N_r(B_b) := B_l$. Otherwise, set $N_l(B_l) := B_l$, $N_r(B_b) := B_b$, $E(B_l) := F_r^3(B_l)$, $E(B_b) := F_l^3(B_b)$, $E(E(B_l)) := B_l$, and $E(E(B_b)) := B_b$.

6. If $F_r(B_r) \neq B_r$, then set $N_r(B_r) := B_a$ and $N_l(B_a) := B_r$. Otherwise, set $N_r(B_r) := B_r$, $N_l(B_a) := B_a$, $E(B_r) := F_l^3(B_r)$, $E(B_a) := F_r^3(B_a)$, $E(E(B_r)) := B_r$, and $E(E(B_a)) := B_a$.

7. If $F_l(B_{r+1}) \neq B_{r+1}$, then set $N_l(B_{r+1}) := B_{l-1}$ and $N_r(B_{l-1}) := B_{r+1}$. Otherwise, set $N_l(B_{r+1}) := B_{r+1}$, $N_r(B_{l-1}) := B_{l-1}$, $E(B_{r+1}) := F_r^3(B_{r+1})$, $E(B_{l-1}) := F_l^3(B_{l-1})$, $E(E(B_{r+1})) := B_{r+1}$, and $E(E(B_{l-1})) := B_{l-1}$.

8. If $F_r(B_{a-1}) \neq B_{a-1}$, then set $N_r(B_{a-1}) := B_{b+1}$ and $N_l(B_{b+1}) := B_{a-1}$. Otherwise, set $N_r(B_{a-1}) := B_{a-1}$, $N_l(B_{b+1}) := B_{b+1}$, $E(B_{a-1}) := F_l^3(B_{a-1})$, $E(B_{b+1}) := F_r^3(B_{b+1})$, $E(E(B_{a-1})) := B_{a-1}$, and $E(E(B_{b+1})) := B_{b+1}$.

- If $B \in \mathcal{X}$ and $F_r^\Psi(B) = B_r$, then $F_r^\Phi(B) = B_{a-1}$.
- If $B \in \overline{\mathcal{X}}$ and $F_r^\Psi(B) = B_b$, then $F_r^\Phi(B) = B_{l-1}$.
- If $B \in \mathcal{X} \cup \overline{\mathcal{X}}$ satisfies none of the two previous conditions, then $F_r^\Phi(B) = F_r^\Psi(B)$.
- If $B \in \mathcal{X}$ and $F_l^\Psi(B) = B_l$, then $F_l^\Phi(B) = B_{b+1}$.
- If $B \in \overline{\mathcal{X}}$ and $F_l^\Psi(B) = B_a$, then $F_l^\Phi(v) = B_{r+1}$.
- If $B \in \mathcal{X} \cup \overline{\mathcal{X}}$ satisfies none of the two previous conditions, then $F_l^\Phi(B) = F_l^\Psi(B)$.
- The symmetric definitions hold for the semiblocks of Γ.

A proposition similar to Proposition 6.3.15 holds for the join operation. That is, the

enumeration $\langle \mathcal{X}, \mathcal{Y}, \overline{\mathcal{X}}, \overline{\mathcal{Y}} \rangle$ defined as above is a round semiblock enumeration of $G = H_1 + H_2$.

The last operation that we discuss is the insertion of one universal semiblock. Suppose that $\Psi = B_1, \ldots, B_{i-1}, B_{i+1}, \ldots, B_k$ is a round semiblock enumeration of a co-bipartite graph G, where $\mathcal{X} = [B_1, B_{i-1}]$ is a range of co-components Ψ. Define the enumeration $\langle \mathcal{X}, \{B_i\}, \overline{\mathcal{X}}, \emptyset \rangle$ as the circular enumeration $\Phi = [B_1, B_{i-1}], B_i, [B_{i+1}, B_k]$, where F_r^Φ and F_l^Φ are defined for as below for $B \in \Psi$.

- $F_r^\Psi(B_i) = B_{i-1}$ and $F_l^\Phi(B_i) = B_{i+1}$.
- If $F_l^\Psi(B) \neq B_{i+1}$ then $F_l^\Phi(B) = F_l^\Psi(B)$; otherwise $F_l^\Phi(B) = B_i$.
- If $F_r^\Phi(B) \neq B_{i-1}$ then $F_r^\Phi(B) = F_r^\Psi(B)$; otherwise $F_r^\Phi(B) = B_i$.

Define $\overline{\{B_i\}} = \emptyset$ when B_i is a universal semiblock. Say that two round semiblocks enumerations Ψ and Γ are *joinable* when at most one of Ψ and Γ contains a universal semiblock, and this universal semiblock is the unique semiblock of such enumeration. The inverse of Propositions 6.3.12 and 6.3.15 is as follows.

Proposition 6.3.17. *Let Ψ and Γ be round semiblock enumerations of co-bipartite graphs H_1 and H_2, and let $\mathcal{X}$ and $\mathcal{Y}$ be ranges of co-components of Ψ and Γ, respectively. If Ψ and Γ are joinable, then $\langle \mathcal{X}, \mathcal{Y}, \overline{\mathcal{X}}, \overline{\mathcal{Y}} \rangle$ is a round semiblock enumeration of $H_1 + H_2$.*

Proposition 6.3.17 yields an algorithm to join Ψ and Γ into a round semiblock enumeration of G. The details of this algorithm are very similar to those in Algorithms 6.5 and 6.6, and it can be implemented so as to run in $O(1)$ time.

Observe that $\langle \mathcal{X}, \mathcal{Y}, \overline{\mathcal{X}}, \overline{\mathcal{Y}} \rangle$ and $\langle \mathcal{X}, \overline{\mathcal{Y}}, \overline{\mathcal{X}}, \mathcal{Y} \rangle$ are different round semiblock enumerations of G. So, we can generate many round semiblock enumerations of G by combining the round semiblock enumerations of its co-components in different ways. It turns out that all the round semiblock enumerations of G can be generated in this way, as it was noted by Huang.

Theorem 6.3.18 ([Hua92])**.** *Let G be a co-bipartite PCA graph, $\mathcal{B}$ be a semiblock partition of G that contains at most one universal semiblock, and Φ be a round semiblock enumeration of G that contains exactly the semiblocks of $\mathcal{B}$. If G is not co-connected, then there are two joinable round semiblock enumerations Ψ and Γ such that:*

- $\mathcal{B} = \Psi \cup \Gamma$, $\Psi \cap \Gamma = \emptyset$, *and*
- $\Phi = \langle \mathcal{X}, \mathcal{Y}, \overline{\mathcal{X}}, \overline{\mathcal{Y}} \rangle$ *for some range of co-components $\mathcal{X}$ of Ψ and some range of co-components $\mathcal{Y}$ of Γ.*

There are some facts about the algorithms of this section that we would like to highlight, and a few details that we intentionally avoided for the sake of simplicity. The first fact to keep in mind is that the algorithms of this section may generate round semiblock

enumerations of PIG graphs that contain no end semiblocks. This is a desired property and we should restore the straightness of the enumeration, if required. The second fact is that the graphs H_1 and H_2 need not be connected when they are obtained from splitting G, nor when they are joined to generate G. When G is split into two disjoint graphs H_1 and H_2, then we should update the set of rings that represents the round semiblock enumeration so as to include the pointers to the leftmost end semiblocks of the contigs of H_1 and H_2. The last detail is that we should update the PHCA flag whenever one of the obtained semiblock enumerations is locally straight. This can be done in $O(1)$ time as in the following proposition.

Proposition 6.3.19. *Let $\Phi = B_1, \ldots, B_k$ be a round semiblock enumeration, $[B_l, B_r]$ be a range of co-components of Φ, and $B_{a-1} = F_r(B_l)$. Then Φ is locally straight if and only if*

- *Φ is a straight, or*
- *$F_r(F_r(B_{a-1})) = B_{l-1}$ and $F_r(F_r(B_{l-1})) = B_{a-1}$.*

Proof. Suppose first that Φ is locally straight. If $F_r(B_{a-1}) = B_{a-1}$, then Φ is straight. Otherwise, $B_{a-1} \longrightarrow B_a$ and $B_l \not\longrightarrow B_a$ because $B_{a-1} = F_r(B_l)$. Observe that if B_l is universal, then $B_a = F_l(B_l)$. On the other hand, if B_l is not universal then B_a is the leftmost semiblock in $\overline{N}([B_l, B_r])$ and $B_a \longrightarrow B_{l-1}$, by Proposition 6.3.14. Whichever the case, $[B_l, B_{a-1}]$ and $[B_a, B_{l-1}]$ are two complete sets, where $B_l \longrightarrow B_{a-1}$ and $B_a \longrightarrow B_{l-1}$. Hence, $F_r(B_{a-1}) \longrightarrow B_{l-1}$ which implies that $F_r(F_r(B_{a-1})) \in [B_{l-1}, B_{a-1}]$. If $F_r(F_r(B_{a-1})) \neq B_{l-1}$, then $B_l, B_{a-1}, F_r(B_{a-1})$ is a directed triangle. Similarly, if $F_r(B_{l-1}) \neq B_{l-1}$ and $F_r(F_r(B_{l-1})) \neq B_{a-1}$, then $B_a, B_{l-1}, F_r(B_{l-1})$ is a directed triangle.

For the converse, suppose that Φ is not locally straight, so it contains three semiblocks B_i, B_j, and B_k that form a directed triangle. Thus, Φ is not straight and, as before, $[B_l, B_{a-1}]$ and $[B_a, B_{l-1}]$ are two complete sets where $B_l \longrightarrow B_{a-1}$ and $B_a \longrightarrow B_{l-1}$. By exchanging $[B_l, B_{a-1}]$ with $[B_a, B_{l-1}]$ if required, assume that $[B_l, B_{a-1}]$ contains at least two semiblocks of B_i, B_j, B_k, say B_i and B_k. So, $[B_l, B_{a-1}]$ does not contain B_j since otherwise $B_{a-1} \longrightarrow B_l$. Then, since $B_i \longrightarrow B_j \longrightarrow B_k$, we obtain that B_{a-1}, B_j, B_l is a directed triangle. This implies that $F_r(B_{a-1}) \in [B_a, B_{l-1}]$ and $F_r(F_r(B_{a-1})) \in [B_l, B_r)$, so $F_r(F_r(B_{a-1})) \neq B_{l-1}$. □

6.4 The vertex-only incremental algorithm

In this section we develop an efficient algorithm for the PCA and PHCA vertex-only incremental recognition problems. The input of these problems is a graph G and a vertex $v \notin V(G)$ together with its set of neighbors $N(v) \subseteq V(G)$. The goal is to update the representation of G into a representation of $H = G \cup \{v\}$ where $N_H(v) = N(v)$, if

possible. This section is divided into two subsections. In the first subsection we show how to insert a vertex into a round block enumeration. In the second subsection we show the complete vertex-incremental algorithms for both PCA and PHCA graphs.

6.4.1 The refinement procedure

In this first part we consider a partial version of the vertex insertion problem. Roughly speaking, given a vertex v of a universal-free graph H and a round block enumeration Φ of $H \setminus \{v\}$, we are to find a round block enumeration Ψ of H such that $\Psi \setminus \{v\}$ looks similar to Φ. By similar, we mean that the semiblocks of $\Psi \setminus \{v\}$ appear in the same order as in Φ. A rigorous definition of "similar" for contigs can be found in Section 3.2 of [HSS01]. Its generalization to rings is in the next lemma.

Lemma 6.4.1 (Decremental Refinement Lemma)**.** *Let $\Psi = B_1, \ldots, B_k$ be a non-straight ring of a graph H, and $v \in B_p$. Call $B_l = F_l^{\Psi}(B_p)$ and $B_r = F_r^{\Psi}(B_p)$, and define L, R, and Φ_p according to rules (i) to (iii). Then,*

$$\Phi = B_l \cup L, \Phi_p, B_r \cup R, (B_r, B_l) \setminus \{R, L\}$$

is a (possibly straight) semiblock ring of $H \setminus \{v\}$ that contains at most two semiblocks that are not blocks of $H \setminus \{v\}$. Furthermore, one of these semiblocks is the universal semiblock B of Ψ, and the other semiblock is a twin of B in $H \setminus \{v\}$.

(i) If B_l is twin of B_{l-1} in $H \setminus \{v\}$, then $L = B_{l-1}$; otherwise, $L = \emptyset$.

(ii) If B_r is twin of B_{r+1} in $H \setminus \{v\}$, then $R = B_{r+1}$; otherwise, $R = \emptyset$.

(iii) If $|B_p| = 1$, then $\Phi_p = (B_l, B_p), (B_p, B_r)$; if not, $\Phi_p = (B_l, B_p), B_p \setminus \{v\}, (B_p, B_r)$.

Proof. Suppose first that $|B_p| > 1$, *i.e.*, v has twins in H. Then, $H \setminus \{v\}$ has the same blocks as H, with the exception of B_p that is replaced by $B_p \setminus \{v\}$. Consequently, Φ as defined in the lemma statement is equal to $B_p \setminus \{v\}, (B_p \circ B_p)$, and it is clearly a round block enumeration of $H \setminus \{v\}$. For the rest of the proof, suppose that $B_p = \{v\}$, *i.e.*, v has no twins in H.

Clearly, $\mathcal{B} = B_1, \ldots, B_{p-1}, B_{p+1}, \ldots, B_k$ is a semiblock partition of $H \setminus \{v\}$. Not all the semiblocks in $\mathcal{B}$ are necessarily blocks of $H \setminus \{v\}$, but all of them are included in some block of $H \setminus \{v\}$, by definition. Those members of $\mathcal{B}$ that are not blocks of $H \setminus \{v\}$ need to be joined to form a block. Let $B \notin \mathcal{B}$ be a block of $H \setminus \{v\}$. Since B is not a block of H, then v is neither fully adjacent nor non-adjacent to B. Consequently, $B_a = (B \cap N(v))$ and $B_b = (B \setminus N(v))$ are nonempty blocks of H and $B = B_a \cup B_b$. Observe that $B_a \in [B_l, B_r]$, $B_b \in (B_r, B_l)$, and that B_a and B_b are twin semiblocks of $H \setminus \{v\}$. Furthermore, B is a universal block of $H \setminus \{v\}$ if and only if B_a is the universal block in H.

We claim that B_b and B_a are consecutive in Ψ when B_a is not universal. On the contrary, suppose that there is some block $B_c \in (B_b, B_a)$ when $B_b \longrightarrow_\Psi B_a$. By definition, B_a is adjacent to all the blocks in $[F_l(B_b), B_a)$. Since B_a is not universal then $B_a \not\longrightarrow F_l(B_b)$, thus $F_l(B_a) = F_l(B_b) = F_l(B_c)$. But then, in Ψ, B_c properly dominates B_b and B_a properly dominates B_c because B_a, B_b and B_c are blocks of Ψ. This is impossible given that $N_H[B_a] \setminus N_H[B_b] = \{B_p\}$. Therefore, when $B_b \longrightarrow_\Psi B_a$, we obtain that $a = l$ and $b = l - 1$. The same arguments can be used to conclude that if $B_a \longrightarrow_\Psi B_b$ then $a = r$ and $b = r + 1$. That is, the only possible pairs of non-universal blocks of H that are not blocks of $H \setminus \{v\}$ are B_l with B_{l-1} and B_r with B_{r+1}.

If B_l and B_{l-1} are twins in $H \setminus \{v\}$, then $B_l \cup B_{l-1}$ is a block of $H \setminus \{v\}$. Similarly, if B_r and B_{r+1} are twins of $H \setminus \{v\}$, then $B_r \cup B_{r+1}$ is a block of $H \setminus \{v\}$. Note that if $l - 1 = r + 1$, then either B_l is not a twin of B_{l-1} or B_r is not a twin of B_{r+1} since otherwise B_r would be a twin of B_l, contradicting the fact that B_l and B_r are different blocks of Ψ. Finally, note that B_l and B_r are different from B_p because B_p is not an end block of Ψ. Therefore, the enumeration Φ defined in the lemma's statement is a semiblock ring of $H \setminus \{v\}$, where $\longrightarrow_\Phi$ is the same as the restriction of $\longrightarrow_\Psi$ to the semiblocks of Φ. Furthermore, Φ contains at most two semiblocks that are not blocks, one is the universal block B of H and the other is the twin of B in $H \setminus \{v\}$. □

Let $\Psi = B_1, \ldots, B_k$ be a non-straight ring of a graph H and $v \in B_p$. By the lemma above, $\Phi = B_l \cup L, \Phi_p, B_r \cup R, (B_r, B_l) \setminus \{R, L\}$ is a ring of $H \setminus \{v\}$. Say that such Φ is *refinable*, and that Ψ is a *refinement* of Φ.

The Decremental Refinement Lemma is useful to find out a refinable ring Φ of $H \setminus \{v\}$, when a ring Ψ of H is given. However, we would like to answer the opposite question. Given a ring Φ of $H \setminus \{v\}$, is Φ refinable? If so, we would also like to obtain a refinement Ψ of Φ. The answer when Ψ is a contig is given in Propositions 4.1 and 4.2 of [DHH96]. The generalization of these propositions to non-straight rings that contain no universal blocks is given in the following Incremental Refinement Lemma. We try to use the same subindices that are used in [DHH96] to highlight the similarities.

Lemma 6.4.2 (Incremental Refinement Lemma)**.** *Let v be a vertex of a universal-free graph H and, Φ be a ring of $H \setminus \{v\}$. If H is not a PIG graph, then Φ is refinable if and only there are two (possibly equal) blocks B_a and B_c that are co-adjacent to v and such that conditions (i) to (iv) are met.*

(i) v is fully adjacent to all the blocks in $(B_a \circ B_c)$ and it is not adjacent to all the blocks in (B_c, B_a).

(ii) $F_r(B_a) \in [B_a \circ B_c]$ and $F_r(B_a) \notin [F_l(B_c), B_c]$. In other words, if $B_a \neq B_c$ then B_a, $F_r(B_a)$, $F_l(B_c)$, B_c appear in this order in a traversal of $[B_a, B_c]$, where possibly $B_a = F_r(B_a)$ or $B_c = F_l(B_c)$.

(iii) *If either $B_a \neq B_c$ or v is not adjacent to B_a, then every block in $(B_a \circ B_c)$ is adjacent or equal to either B_l or B_r, where B_l and B_r are the leftmost and rightmost blocks of $[B_a \circ B_c]$ that are adjacent to v, respectively.*

(iv) *If $B_a = B_c$ and v is adjacent to B_a, then either $F_r(B_a)$ is immediately to the left of $F_l(B_{a-1})$ or $F_r(B_{a+1})$ is immediately to the left of $F_l(B_a)$.*

Proof. $\Longrightarrow$) By definition, Φ has some refinement $\Psi = B_1^\Psi, \ldots, B_k^\Psi$, where $v \in B_p^\Psi$. Call $B_l^\Psi = F_l^\Psi(B_p^\Psi)$ and $B_r^\Psi = F_r^\Psi(B_p^\Psi)$. If v has some twin in H then, by the definition of refinable, $\Phi = (B_p^\Psi \circ B_p^\Psi), B_p^\Psi \setminus \{v\}$. It is not hard to see that (i) to (iv) follow by taking $B_a = N_l^\Psi(B_l^\Psi)$ and $B_c = N_r^\Psi(B_r^\Psi)$. Suppose from now on that v has no twins in H, *i.e.*, $B_p^\Psi = \{v\}$.

By the definition of refinable, $\Phi = B_l^\Psi \cup L, \Phi_p, B_r^\Psi \cup R, (B_r^\Psi, B_l^\Psi) \setminus \{R, L\}$, where L, R, and Φ_p are defined from Ψ as in the Decremental Refinement Lemma. Consider the following two cases.

Case 1: $B_{l-1}^\Psi \neq B_{r+1}^\Psi$. If B_l^Ψ and B_{l-1}^Ψ are twins in $H \setminus \{v\}$, then take $B_a = B_l^\Psi \cup B_{l-1}^\Psi$; otherwise, take $B_a = B_{l-1}^\Psi$. Similarly, if B_r^Ψ and B_{r+1}^Ψ are twins in $H \setminus \{v\}$, then take $B_c = B_r^\Psi \cup B_{r+1}^\Psi$; otherwise, take $B_c = B_{r+1}^\Psi$. Observe that $B_a \neq B_c$, and that, by the Decremental Refinement Lemma, B_a and B_c are blocks of $H \setminus \{v\}$. Moreover, v is fully adjacent precisely to the blocks in (B_a, B_c) and it is adjacent to none of the blocks in (B_c, B_a). That is, conditions (i) and (iv) are satisfied by B_a and B_c.

By the way we defined B_a and B_c, we know that $B_{l-1}^\Psi = B_a \setminus N(v)$ and $B_{r+1}^\Psi = B_c \setminus N(v)$. Call $B_b = F_r^\Phi(B_a)$ and $B_d = F_l^\Phi(B_b)$.

Consider the statement (ii). Since $B_{l-1}^\Psi \not\longrightarrow_\Psi B_p^\Psi$ then $B_a \not\longrightarrow_\Phi B_c$, thus B_a, B_b, B_c appear in this order in Φ, where possibly $B_a = B_b$. Similarly, B_a, B_d, B_c appear in this order in Φ where possibly $B_c = B_d$. If either $B_a = B_b$ or $B_c = B_d$, then (ii) follows. For the other case suppose, in contradiction to (ii), that B_a, B_b, B_d, and B_c do not appear in this order in Φ. Hence, B_a, B_d, B_b, B_c appear in this order in Φ and, so, $B_a \longrightarrow_\Phi B_b \longrightarrow_\Phi B_c$. Then, by the Decremental Refinement Lemma, $B_b = B_s^\Psi$ for some $B_s^\Phi \in [B_l^\Psi, B_r^\Psi]$, so either $B_p^\Psi \in [B_s^\Psi, B_{r+1}^\Psi]$ or $B_p^\Psi \in [B_{l-1}^\Psi, B_s^\Psi]$. In the former case, $B_p^\Psi \longrightarrow_\Psi B_{r+1}^\Psi$ while in the latter case $B_{l-1}^\Psi \longrightarrow_\Psi B_p^\Psi$, because $B_{l-1}^\Psi \longrightarrow_\Psi B_s^\Psi \longrightarrow_\Psi B_{r+1}^\Psi$. Both are impossible, so B_a, B_b, B_d and B_c appear in this order in Φ.

For statement (iii), observe that $B_l = B_a$ when v is adjacent to B_a, while $B_l = B_{a+1}$ when v is not adjacent to B_a. Whichever the case, $B_l^\Psi \subseteq B_l$ and, similarly, $B_r^\Psi \subseteq B_r$. If $(B_a \circ B_c) = \emptyset$, then (iii) is vacuously true. Suppose, then, that there is some $B_i \in (B_a \circ B_c)$. Such B_i is equal to some block $B_s^\Psi \in [B_l^\Psi, B_r^\Psi]$, by the Decremental Refinement Lemma. If $B_s^\Psi \in (B_l^\Psi, B_p^\Psi)$, then $B_l^\Psi \longrightarrow_\Psi B_s^\Psi$, while if

$B_s^\Psi \in (B_p^\Psi, B_r^\Psi)$, then $B_s^\Psi \longrightarrow_\Psi B_r^\Psi$. Consequently, by the Decremental Refinement Lemma, either $B_l \longrightarrow_\Phi B_i$ or $B_i \longrightarrow_\Phi B_r$, *i.e.*, (iii) is true.

Case 2: $B_{l-1}^\Psi = B_{r+1}^\Psi$. If B_l^Ψ and B_{l-1}^Ψ are twins, then take $B_a = B_c = B_l^\Psi \cup B_{l-1}^\Psi$. If B_r^Ψ and B_{l-1}^Ψ are twins, then take $B_a = B_c = B_r^\Psi \cup B_{r+1}^\Psi$. Finally, if B_{l-1}^Ψ is a twin of neither B_l^Ψ nor B_r^Ψ, then take $B_a = B_c = B_{l-1}^\Psi$. Observe that, by the Decremental Refinement Lemma, $B_a = B_c$ is a block of $H \setminus \{v\}$. Moreover, v is fully adjacent to the blocks in $(B_a \circ B_c)$, thus condition (i) is satisfied by B_a and B_c. Condition (ii) is clearly true for $B_a = B_c$. When v is not adjacent to B_a, then $B_l = B_l^\Psi$ and $B_r = B_r^\Psi$ by the Decremental Refinement Lemma. So, we can prove that statement (iii) is true as in Case 1.

Finally, suppose that v is adjacent to B_a for statement (iv). By definition, either B_l^Ψ is a twin of B_{l-1}^Ψ or B_r^Ψ is a twin of B_{l-1}^Ψ. Without loss of generality, assume the former and fix $B_i \in (B_a, B_{a-1})$. By the Decremental Refinement Lemma, B_i corresponds to some block B_s^Ψ, and $B_{a-1} = B_r^\Psi$. If $B_s^\Psi \in [B_l^\Psi, B_p^\Psi)$ then $B_l^\Psi \longrightarrow_\Psi B_s^\Psi$, thus $B_a \longrightarrow_\Phi B_i$. Otherwise, $B_s^\Psi \in (B_p^\Psi, B_r^\Psi)$ because $B_r^\Psi = B_{a-1}$, so $B_i \longrightarrow_\Psi B_{a-1}$. Consequently, every block of Φ in $(B_a \circ B_a)$ is adjacent to either B_a or B_{a-1}. Then, $F_r(B_a)$ is immediately to the left of $F_l(B_{a-1})$ since otherwise there would be a block B in Φ such that $B_a \longrightarrow_\Phi B \longrightarrow_\Phi B_{a-1}$, implying that $B \cap N(v) \neq \emptyset$ is a universal block of H.

$\Longleftarrow$) All we have to do is to show how to build a refinement Ψ of Φ. Consider the following cases.

Case 1: v is adjacent to B_a and $B_a \neq B_c$ (see Figure 6.7 (a) and (c)). Define $\Psi_c = B_c \cap N(v), B_c \setminus N(v)$ when v is adjacent to B_c, and $\Psi_c = B_c$ when v is not adjacent to B_c. Call $B_b = F_r(B_a)$ and define

$$\Psi = (B_c, B_a), B_a \setminus N(v), B_a \cap N(v), (B_a, B_b], \{v\}, [B_{b+1}, B_c), \Psi_c.$$

By (i), all the elements of Ψ are blocks of H, thus Ψ is a circular enumeration of the blocks of H. To see that Ψ is a ring we have to show the orientation of the blocks of H. Let $B^\Psi \in \Psi \setminus \{v\}$ and $B \in \Phi$ be such that $B^\Psi \subseteq B$ and define F_l^Ψ and F_r^Ψ for B^Ψ according to the following rules (see Figure 6.7 (b) and (d)).

- If $F_l(B) = B_{b+1}$ and v is adjacent to B^Ψ, then $F_l^\Psi(B^\Psi) = \{v\}$.
- If $F_l(B) = B_{b+1}$ and v is not adjacent to B^Ψ, then $F_l^\Psi(B^\Psi) = B_{b+1} \cap N(v)$.
- If $F_l(B) \in N(v) \setminus \{B_{b+1}\}$, then $F_l^\Psi(B) = F_l(B) \cap N(v)$.
- If $F_l(B) \notin N(v)$, then $F_l^\Psi(B) = F_l(B)$.
- $F_l^\Psi(\{v\}) = B_a \cap N(v)$.
- If $F_r(B) = B_b$ and v is adjacent to B^Ψ, then $F_r^\Psi(B^\Psi) = \{v\}$.
- If $F_r(B) = B_b$ and v is not adjacent to B^Ψ, then $F_r^\Psi(B^\Psi) = B_b \cap N(v)$.
- If $F_r(B) \in N(v) \setminus \{B_b\}$, then $F_r^\Psi(B) = F_r(B) \cap N(v)$.

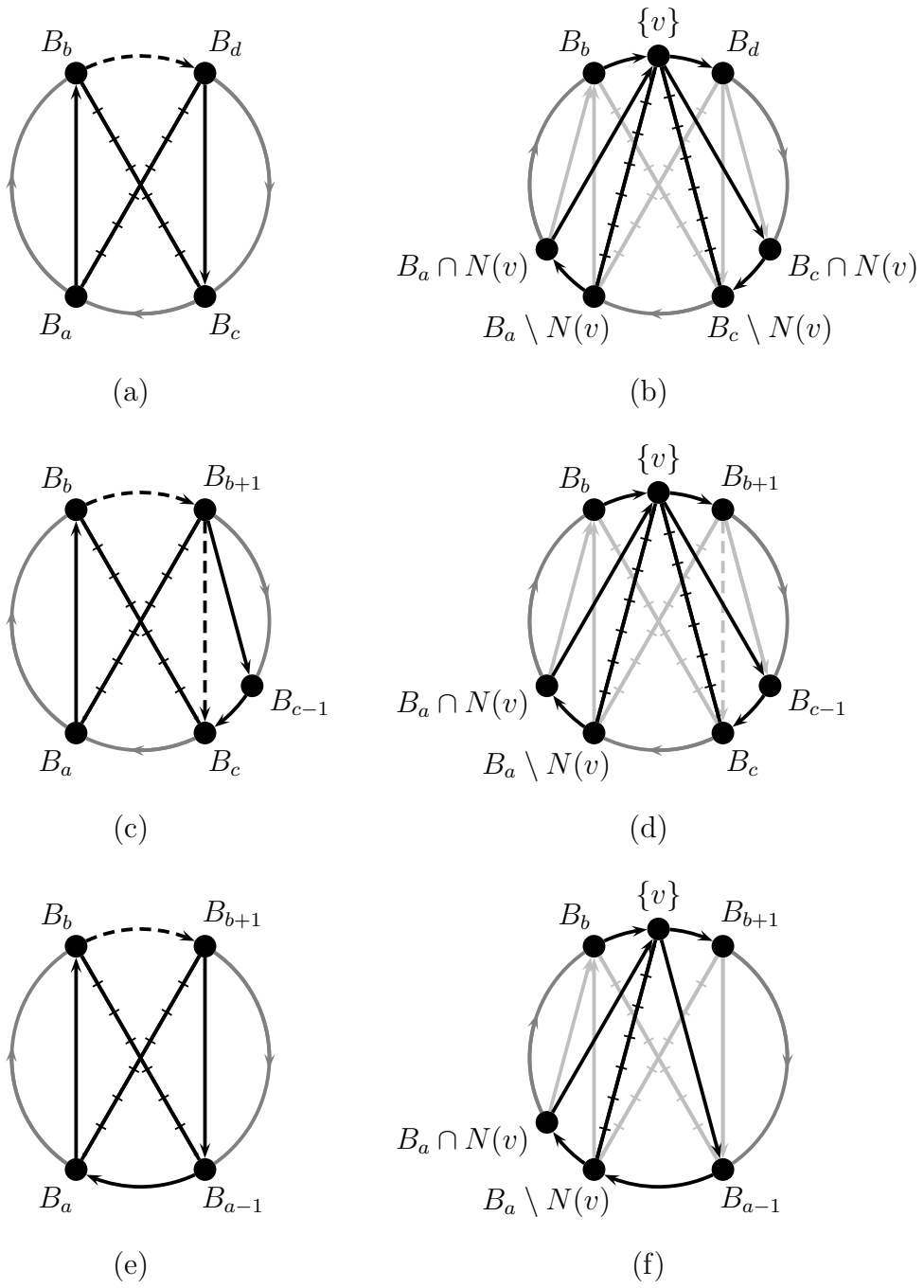


Figure 6.7: The effects of inserting a new vertex into a round enumeration Φ. Dashed lines indicate that there could be or not an edge between the two joined vertices, while those marked lines indicate that there is no edge between the two joined vertices. Figures (a) and (b) depict the case v adjacent to both B_a and B_c with $B_a \neq B_c$; (c) and (d) show the case v adjacent to B_a and not to B_c; and (e) and (f) represent the case v adjacent to $B_a = B_c$.

- If $F_r(B) \notin N(v)$, then $F_r^\Psi(B) = F_r(B)$.
- If v is adjacent to B_c, then $F_r^\Psi(\{v\}) = B_c \cap N(v)$; otherwise, $F_r^\Psi(\{v\}) = B_{c-1}$.

We claim that $F_l^\Psi(\{v\}) = B_l \cap N(v)$ and $F_r^\Psi(\{v\}) = B_r \cap N(v)$, where B_l and B_r are defined as in (iii). The former is clearly true because $B_l = B_a$. To see why is the latter true, consider whether B_c is the leftmost end block of Φ. If B_c is the leftmost end block, then v is adjacent to B_c or otherwise we can obtain a straight block enumeration of H as in [DHH96]; just replace the last rule with $F_r^\Psi(\{v\}) = \{v\}$ in the otherwise case. If B_c is not the leftmost end block, then $B_{c-1} \longrightarrow_\Phi B_c$, and so, by (ii), $B_{c-1} \in (B_b, B_c)$ in Φ. Therefore, $F_r^\Psi(\{v\}) = B_r \cap N(v)$ by the last rule.

To prove that F_l^Ψ and F_r^Ψ define a round block enumeration of Ψ, we ought to see that $B \in [B', F_r^\Psi(B')]$ if and only if $B' \in [F_l^\Psi(B), B]$ for every $B, B' \in \Psi$, and that $N[B] = [F_l^\Psi(B), F_r^\Psi(B)]$ for every $B \in \Psi$. This is easy to prove with a case analysis according to the position of B in Ψ (see [DHH96] for instance). In particular, observe that every block of Φ in (B_a, B_c) is adjacent to either B_l or B_r by (iii). In this case $B_l = B_a$, so all the blocks in (B_b, B_c) are adjacent or equal to B_r in Φ.

Case 2: v is adjacent to B_c and $B_a \neq B_c$. This case is analogous to Case 1.

Case 3: v is adjacent to B_a and $B_a = B_c$. By (iv), either $F_r(B_a)$ is immediately to the left of $F_l(B_{a-1})$ or $F_r(B_{a+1})$ is immediately to the left of $F_l(B_a)$. Suppose, w.l.o.g., that $F_r(B_a)$ is immediately to the left of $F_l(B_{a-1})$ (see Figure 6.7 (e)). Call $B_b = F_r(B_a)$ and define

$$\Psi = B_a \setminus N(v), B_a \cap N(v), (B_a, B_b], \{v\}, [B_{b+1}, B_a).$$

Again, by (i), Ψ is a circular enumeration of the blocks of H. Define F_l^Ψ and F_r^Ψ as in Case 1, with the only exception that the last rule is replaced with the rule $F_r^\Psi(\{v\}) = B_{a-1}$ (see Figure 6.7 (f)).

By (iv), $B \longrightarrow_\Phi B_{a-1}$ for every $B \in (F_r(B_a), B_{a-1})$, so it can be proved that F_l^Ψ and F_r^Ψ define a round block enumeration. Note that in this case B_{a-1} plays the role of B_r in Case 1.

(This case is important because it shows that the refinement of a refinable ring Φ is not unique. By (iv), either $F_r(B_a)$ is immediately to the left of $F_l(B_{a-1})$ or $F_r(B_{a+1})$ is immediately to the left of $F_l(B_a)$. We assumed, w.l.o.g., that the former is true, but the latter could be simultaneously true. So, there are two possible refinements, the enumeration Ψ defined above or the enumeration $\Gamma = B_a \cap N(v), B_a \setminus N(v), (B_a, B_{d-1}], \{v\}, [B_d, B_a)$, where $B_d = F_l(B_a)$. There is a good reason behind this ambiguity: $B_{b+1} = B_{d-1}$ and $\langle \{B_a\}, \{B_{b+1}\} \rangle$ is a co-component of $H \setminus \{v\}$ (the reader can check these conditions). Then H is co-bipartite and $B_a \cap N(v)$, $B_a \setminus N(v)$, B_{b+1} and $\{v\}$ form a co-component of H whose reduced graph is isomorphic to P_4. The difference between Ψ and Γ is that the co-component containing $\{v\}$ is reversed from Ψ to Γ.)

Case 4: v is adjacent to neither B_a nor B_c. The construction of Ψ in this case is as in [DHH96]. By (i), v is adjacent only to the blocks in $(B_a \circ B_c)$. Since H is connected then $(B_a \circ B_c) \neq \emptyset$, thus $N(v)$ is precisely $[B_{a+1}, B_{c-1}]$, $B_l = B_{a+1}$, and $B_r = B_{c-1}$. Call $B_b = F_r(B_a)$ and $B_d = F_l(B_c)$. By (ii), $(B_b, B_d) \subseteq [B_l, B_r]$. If there is some block $B_p \in (B_b, B_d)$ such that $B_l \longrightarrow B_p \longrightarrow B_r$, then, as in [DHH96], v is a twin of all the vertices in B_p. In this case, define $\Psi = (B_p, B_p), B_p \cup \{v\}$ where F_r and F_l are not changed. Finally, if there is no $B_p \in (B_b, B_d)$ such that $B_l \longrightarrow B_p \longrightarrow B_r$, then take B_u as the leftmost block in $(B_b, B_d]$ such that $B_u \longrightarrow B_r$, and B_w as the rightmost block in $[B_b, B_d)$ such that $B_l \longrightarrow B_w$. Note that B_w, B_u appear in this order in $[B_b, B_d]$ since otherwise $B_u \in (B_b, B_d)$ and $B_l \longrightarrow B_u \longrightarrow B_r$. Also, by (iii), $(B_w, B_u) = \emptyset$, *i.e.*, $u = w + 1$. In this case, take $\Psi = [B_u, B_w], \{v\}$, where F_l^Ψ and F_r^Ψ are defined by the rules below for $B \in \Psi \cap \Phi$.

- If v is not adjacent to B or $F_l^\Phi(B) \neq B_u$, then $F_l^\Psi(B) = F_l^\Phi(B)$.
- If v is adjacent to B and $F_l^\Phi(B) = B_u$, then $F_l^\Psi(B) = \{v\}$.
- If v is not adjacent to B or $F_r^\Phi(B) \neq B_w$, then $F_r^\Psi(B) = F_r^\Phi(B)$.
- If v is adjacent to B and $F_r^\Phi(B) = B_w$, then $F_r^\Psi(B) = \{v\}$.
- $F_l^\Psi(\{v\}) = B_l$ and $F_r^\Psi(\{v\}) = B_r$.

It is not hard to see that Ψ is refinement of Φ (see also [DHH96]).

□

Now we describe the algorithm to determine if a non-universal vertex can be inserted into a round semiblock enumeration Φ, without changing the order of the blocks of Φ. The first step of the algorithm is to check whether H is a PIG graph by calling the HSS algorithm. When the HSS algorithm succeeds in the insertion of v, then H is a PIG graph and there in nothing else to do. When the HSS algorithm fails, we know that H is not a PIG graph and Φ is a ring, so we check the conditions of the Incremental Refinement Lemma to see whether Φ is refinable. If Φ satisfies all the conditions then we can insert v in the same way as it is described in the Incremental Refinement Lemma. Observe that Φ is transformed into a straight enumeration whenever H is PIG. Algorithm 6.7 implements the insertion of v for the case in which the blocks B_a and B_c of the lemma are not equal. The variable New is just a new self pointer pointing to the block. Operations $Prev$ and $Next$ are used as usual but the reader can observe that these operations are not really required (recall that we are allowed to use the extreme pointers in this algorithm). Variable B_{p-1}^Ψ is used to mark the position in which $\{v\}$ is going to be created, when v has no twin vertices in H. Similarly, B_p^Ψ is the block that contains v, while B_l^Ψ and B_r^Ψ are used to mark the leftmost and rightmost neighbors blocks of B_p^Ψ in H. Finally, observe that B_{p-1}^Ψ and $N_r(B_{p-1}^\Psi)$ are the only blocks of Φ that can be end blocks of $H \setminus \{v\}$. We clear their end pointers at Step 22.

All the operations that are required to obtain a refinement of Φ take $O(d(v))$ time. In particular, observe that for the test of Conditions (ii) and (iii) we can first mark all the

Algorithm 6.7 Refinement of a round block enumeration

Input: A vertex v of a universal-free graph H, and a round block enumeration $\Phi = B_1, \ldots, B_k$ of $H \setminus \{v\}$.

Output: If H is a PIG graph, then a straight enumeration of H is obtained. When H is not a PIG graph and Φ is refinable, then Φ is updated into a refinement Ψ; otherwise, Φ is not modified and the algorithm reports an error.

1. If Φ is straight, then apply the HSS algorithm. If successful then halt.

//Test of the conditions of the Incremental Refinement Lemma.

2. If the blocks fully adjacent to v do not form a range, then halt in error.

3. Let B_a and B_c be such that v is fully adjacent precisely to (B_a, B_c) (assume $B_a \neq B_c$).

4. If v is adjacent to some block in (B_c, B_a), then halt in error.

5. If $F_r(B_a) \notin [B_a, B_c]$ or $F_r(B_a) \in [F_l(B_c), B_c]$, then halt in error.

6. Let B_l (resp. B_r) be the leftmost (resp. rightmost) block adjacent to v in (B_a, B_c).

7. If $F_l(B) \in (B_l, B)$ or $F_r(B) \in (B, B_r)$ for $B \in (F_r(B_a), F_l(B_c))$, then halt in error.

//Insertion of v

8. If v is adjacent to B_a, then:

9. Move the vertices of $B_a \cap N(v)$ into a new block B_l^Ψ between B_a and B_{a+1}.

10. Set $F_l(B_l^\Psi) := F_l(B_a)$, $F_r(B_l^\Psi) := F_r(B_a)$, $S_r(B_l^\Psi) := S_r(B_a)$, and $S_r(B_a) := New$.

11. Set $B_{p-1}^\Psi := F_r(B_a)$.

12. Otherwise, set $B_l^\Psi := B_{a+1}$ and $B_{p-1}^\Psi :=$ last in $[F_r(B_a), F_l(B_c))$ s.t. $B_{a+1} \longrightarrow B_{p-1}^\Psi$.

13. If v is adjacent to B_c, then:

14. Move the vertices of $B_c \cap N(v)$ into a new block B_r^Ψ between B_{c-1} and B_c.

15. Set $F_l(B_r^\Psi) := F_l(B_c)$, $F_r(B_r^\Psi) := F_r(B_c)$, $S_l(B_r^\Psi) := S_l(B_c)$, and $S_l(B_c) := New$.

16. Set $B_{p-1}^\Psi := Prev(F_l(B_c))$.

17. Otherwise, set $B_r^\Psi := B_{c-1}$.

18. If some $B_p^\Psi \in [B_l^\Psi, B_r^\Psi]$ has $F_l(B_p^\Psi) = B_l^\Psi$ and $F_r(B_p^\Psi) = B_r^\Psi$, then set $B_p^\Psi := B_p^\Psi \cup \{v\}$ and halt.

19. Create a new block $B_p^\Psi := \{v\}$, and insert it between B_{p-1}^Ψ and $Next(B_{p-1}^\Psi)$.

20. For every $B \in [B_l^\Psi, B_p^\Psi]$ such that $F_r(B) = B_{p-1}^\Psi$, set $F_r(B) := S_r(B_p^\Psi)$.

21. For every $B \in [B_p^\Psi, B_r^\Psi]$ such that $F_l(B) = Next(B_{p-1}^\Psi)$, set $F_l(B) := S_l(B_p^\Psi)$.

22. Set $F_l(B_p^\Psi) := S_l(B_l^\Psi)$ and $F_r(B_p^\Psi) := S_r(B_r^\Psi)$, $E(B_{p-1}^\Psi) := E(Next(B_{p-1}^\Psi)) := NULL$.

blocks of $[B_a, B_c]$ so as to solve the membership questions in $O(1)$ time per block. The final step is to update the PHCA flag. By the lemma below, we can traverse the range $[B_l^\Psi, B_r^\Psi]$ so as to test whether $F_r(F_r(B_r^\Psi))) \in [B_l^\Psi, B_r^\Psi]$. If true, then $F_r(B_r^\Psi) \longrightarrow B_l^\Psi$ which implies that H is not PHCA. If false, then the PHCA flag needs not to be updated. This update takes $O(d(v))$ time.

Lemma 6.4.3. *Let Ψ be a non-straight ring of a universal-free graph H and B_p be the block of Ψ that contains v, for $v \in H$. If Ψ is a refinement of Φ then Ψ is locally straight if and only if Φ is locally straight and $F_r^\Psi(B_p) \not\longrightarrow F_l^\Psi(B_p)$.*

Proof. If Φ is not locally straight then neither is Ψ by definition. Also, if $F_r^\Psi(B_p) \longrightarrow F_l^\Psi(B_p)$, then $F_l^\Psi(B_p)$, B_p, $F_r^\Psi(B_p)$ is a directed triangle of Ψ.

For the converse, if Ψ is not locally straight, then either Φ is not locally straight or B_p belongs to a directed triangle. In the latter case, $F_l^\Psi(B_p)$, B_p, $F_r^\Psi(B_p)$ is one such triangle. □

6.4.2 The impact of a new vertex

In this section we combine all the tools that we developed so far, so as to insert a new vertex into a PCA graph. That is, we are given a graph G and a vertex $v \notin V(G)$ together with a set $N(v) \subseteq V(G)$, and we ought to update the representation of G into a representation of $H = G \cup \{v\}$, where $N_H(v) = N(v)$. For the sake of simplicity, throughout this section we allow graphs to be empty, *i.e.*, graphs with empty vertex set. Of course, their unique round block enumeration is simply the empty set as well.

The vertex-insertion algorithm, summarized in Algorithm 6.8, is really simple once all the tools have been developed. Let $\Phi = B_1, \ldots, B_k$ be a round block enumeration of G. The first step is to divide the universal block B, if any, into the two semiblocks $B \cap N(v)$ and $B \setminus N(v)$ so as to obtain a round semiblock enumeration Φ'. The reason for doing this is that $B \setminus N(v)$ and $B \cap N(v)$ get separated into different co-components of Φ'. For the implementation, the set $N(v)$ is traversed to test whether v is adjacent and co-adjacent to the universal block B. If affirmative, then all the neighbors of v in B are moved into a new block, and its self and far pointers are updated as we did in Algorithm 6.7 for B_l^Ψ and B_r^Ψ. Once the universal block is divided, we compute the co-components of G by running Algorithm 6.4 with input Φ'. This algorithm either outputs the list of co-bipartite ranges of Φ' or it halts with output Φ'.

Suppose first that Algorithm 6.4 halted without finding any co-bipartite range of Φ'. Then either G is not co-bipartite or H is not a PCA graph. We can figure out whether H is a PCA graph by running Algorithm 6.7 with input Φ' and v: if H is a PCA graph then Φ is updated into a round block enumeration Ψ, otherwise H is not a PCA graph.

Algorithm 6.8 Insertion of a vertex v into a PCA graph G

Input: A round block enumeration Φ of a graph G, and a vertex $v \notin V(G)$ with $N(v) \subset V(G)$.
Output: If $G \cup \{v\}$ is a PCA graph, then Φ is updated into a round block enumeration of $G \cup \{v\}$; otherwise, the algorithm halts in error.

1. If there is a universal block B_i in $N(v)$ and v is co-adjacent to B_i, then:

2. Move the vertices of $B_i \cap N(v)$ into a new block B between B_i and B_{i+1}.

3. Set $F_l(B) := F_l(B_i)$, $F_r(B) := F_r(B_i)$, $S_r(B) := S_r(B_i)$, and $S_r(B_i) := New$.

4. Apply Algorithm 6.4 to find the co-components of Φ.

5. If Algorithm 6.4 halts before it terminates, then:

6. Run Algorithm 6.7 with input Φ and v. In case of errors, inform that G is not PCA; otherwise Φ was modified into a round block enumeration of $G \cup \{v\}$. Halt.

7. Let $\mathcal{X}_1, \ldots, \mathcal{X}_s$ be the list of co-bipartite ranges of Φ found by Algorithm 6.4.

8. If v is universal and there is a universal block $B \in \Phi$, then insert v into B and halt.

9. If Φ has some universal semiblock adjacent to v, then remove it with Algorithm 6.5.

10. Let, w.l.o.g, $\mathcal{X}_1, \ldots, \mathcal{X}_r$ be the co-bipartite ranges with semiblocks co-adjacent to v, and call $G_1 := G[\mathcal{X}_1 \cup \overline{\mathcal{X}_1} \cup \ldots \cup \mathcal{X}_r \cup \overline{\mathcal{X}_r}]$ and $G_2 := G_1 \setminus G$.

11. Run Algorithms 6.5 and 6.6 to obtain a round block enumeration Γ_i of $G[\mathcal{X}_i \cup \overline{\mathcal{X}_i}]$ for $1 \leq i \leq r$, and a round block enumeration Φ_2 of G_2.

12. For every round block enumeration Φ_1 of G_1, computed from $\{\Gamma_i\}_{1 \leq i \leq r}$ as in Theorem 6.3.18:

13. Run Algorithm 6.7 to Φ_1 and v.

14. If a round block enumeration Ψ of $G_1 \cup \{v\}$ is obtained, then:

15. If $\Phi_2 = \emptyset$, then take Ψ as the round block enumeration of $G \cup \{v\}$ and halt.

16. Run Algorithm 6.3 to obtain a co-bipartite range $\mathcal{X}$ of Φ_1.

17. Let $\mathcal{Y}$ be a co-bipartite range of Φ_2, obtained at Step 11.

18. Compute $\langle \mathcal{X}, \mathcal{Y}, \overline{\mathcal{X}}, \overline{\mathcal{Y}} \rangle$ as in Theorem 6.3.18.

19. If a universal block B was removed at Step 9, then compute $\langle \mathcal{X}, \{B\}, \overline{\mathcal{X}}, \emptyset \rangle$.

20. Halt.

21. Inform that $G \cup \{v\}$ is not a PCA graph and halt.

Indeed, if H is a PIG graph, then the HSS algorithm updates Φ' into a straight block enumeration. On the other hand, if H is PCA but not PIG, then G is not co-bipartite and its unique ring $\Phi' = \Phi$ must be refinable.

Suppose now that Algorithm 6.4 gave as output the list of co-bipartite ranges $\mathcal{X}_1, \ldots, \mathcal{X}_s$ of Φ'. If v is a universal vertex and Φ' has some universal semiblock, say B, then v is inserted into B and the computation finishes. Suppose then that either v is not universal or Φ' has no universal semiblocks. The next step is to find those co-components that have blocks co-adjacent to v, if any. By Proposition 6.3.10, there are at most three such co-components in Φ, including the universal semiblock co-adjacent to v, if any. Let G_1 be the subgraph of G induced by the semiblocks of such co-adjacent co-components, and $G_2 = G \setminus G_1$. Observe that G_1 is empty when v is a universal vertex, while G_2 is empty when v has at least one co-adjacent semiblock of Φ in every co-component of G. By definition, $H = (G_1 \cup \{v\}) + G_2$, and a round block enumeration Φ_1 of G_1 and a round block enumeration of Φ_2 of G_2 can be computed with Algorithms 6.5 and 6.6. For H to be PCA, $H_1 = G_1 \cup \{v\}$ has to be PCA and, furthermore, H_1 has to be co-bipartite when G_2 is not empty. By the Decremental Refinement Lemma, either H_1 is a PIG graph or G_1 is connected and one of its rings is refinable. Since G_1 has only $O(1)$ co-components, then all its rings can be computed in $O(1)$ time from Φ_1, as in Theorem 6.3.18. Then, since H_1 is universal-free by construction, we can apply Algorithm 6.7 to each of these rings to figure out if H_1 is PCA and, if so, we obtain a ring Ψ_1 of H_1 as a by-product. Finally, if G_2 is not empty, then we run Algorithm 6.3 on Ψ_1 to find a range of co-components $\mathcal{X}_1$, and afterwards we compute $\langle \mathcal{X}_1, \mathcal{X}_2, \overline{\mathcal{X}_1}, \overline{\mathcal{X}_2} \rangle$ for any co-bipartition range $\mathcal{X}_2$ of Φ_2. The round block enumeration so obtained is a round block enumeration of H.

Observe that the join operation is always done with blocks of H, thus the PHCA flag is updated by the join algorithm when G_2 is not empty. Also observe that the round block enumeration computed for H_1 is unique, because H_1 is co-connected. (Recall that in Case 3 of the Incremental Refinement Lemma we showed that a ring may admit more than one refinement. In there, we argued that the graph is not co-connected for this to happen. However, in our construction H_1 is co-connected, so this ambiguity could never occur.) Finally, note that this algorithm can be used to obtain two round block enumerations, one the reversal of the other, by applying it to the reversal round block enumeration that is also maintained for G.

All the operations that are executed to obtain the round block enumeration of H take $O(d_H(v) + u)$ time, where $u = 2$ is the number of universal semiblocks of Φ'. As a consequence a linear-time incremental recognition algorithm for PCA graphs is obtained. This solves a problem stated in [DHH96], *i.e.*, can the PCA recognition problem be solved in linear time by an incremental algorithm?.

6.5 The vertex-only decremental algorithm

In this section we develop the vertex-only decremental algorithm that can be combined with the vertex-only incremental algorithm to obtain a vertex-only fully dynamic recognition algorithm, both for PHCA and for PCA graphs. The input for this problem is a representation of some graph H together with a vertex $v \in V(H)$, and the output is a representation of $H \setminus \{v\}$. The representation of $H \setminus \{v\}$ can always be obtained, because the class of PCA graphs is hereditary. Recall that we have to guarantee that the representation is locally straight when $H \setminus \{v\}$ is a PHCA graph, while it has to be straight when $H \setminus \{v\}$ is a PIG graph. Also, we need to update the PHCA flag when the representation is locally straight.

As mentioned in Section 6.2, the data structure for the decremental algorithm is the same as the data structure for the incremental algorithm, with the exception that end pointers are removed. The decremental algorithm is divided into two phases. The first phase computes a round block enumeration of $H \setminus \{v\}$, and the second phase restores the straightness invariant and updates the PHCA flag.

We begin by describing the first phase. Let $\Psi = B_1, \ldots, B_k$ be a round block enumeration of a PCA graph H and $v \in B_p$, and call $B_l = F_l(B_p)$ and $B_r = F_r(B_p)$. When Ψ is straight, we can build a straight block enumeration of $H \setminus \{v\}$ by calling the HSS algorithm. When Ψ is not straight, we build the semiblock ring $\Phi = B_l \cup L, \Phi_p, B_r \cup R, (B_r, B_l) \setminus \{R, L\}$ of $H \setminus \{v\}$, where L and R are defined as in the Decremental Refinement Lemma. By the Decremental Refinement Lemma, there are at most two semiblocks of Φ that are not blocks of $H \setminus \{v\}$, and both of these semiblocks are universal in Φ. So, we can obtain the desired ring by joining these universal semiblocks into one universal block.

Algorithm 6.9 implements the above procedure. Step 3 removes v from B_p when $|B_p| > 1$. Steps 4–6 remove the block $B_p = \{v\}$ and update the far pointers of those blocks that were pointing to B_p. From Steps 7–12 we join the blocks B_l and B_{l-1}, and the blocks B_r and B_{r+1}, when they are twins in $\Psi \setminus \{B_p\}$. Observe that no semiblock of $\Psi \setminus \{B_p\}$ has its right far pointer referencing B_{l-1} because $F_l(B_{l-1}) = F_l(B_l)$. Similarly, no semiblock of $\Psi \setminus \{B_p\}$ has its left far pointer referencing B_{r+1}. Therefore, by moving the self pointers, the algorithm correctly updates all the far pointers of those blocks that were pointing to B_l and B_r in Ψ. After this step, the input enumeration Ψ was transformed in the semiblock ring Φ of $H \setminus \{v\}$ that is defined by the Decremental Refinement Lemma. All that it is left to do is to join the universal semiblocks. To find the universal semiblocks, the algorithm invokes Algorithm 6.4 at Step 13 with input Φ and v. Note that this algorithm always outputs the co-bipartite ranges of Φ when $H \setminus \{v\}$ is co-bipartite because H is a PCA graph. When Φ has two universal semiblocks, one of these semiblocks is adjacent to v by the Decremental Refinement Lemma. The algorithm

Algorithm 6.9 Removal of a vertex v from a PCA graph H

Input: A round block enumeration Ψ of a graph H and a vertex $v \in V(H)$.
Output: Ψ is updated into a round block enumeration Φ of $H \setminus \{v\}$.

1. If Ψ is straight then apply the HSS algorithm and halt.

2. Let B_p be the block that contains v, $B_l := F_l(B_p)$ and $B_r := F_r(B_p)$.

3. If $|B_p| > 1$ then set $B_p := B_p \setminus \{v\}$ and halt.

4. For every $B \in [B_l, B_p)$ such that $F_r(B) = B_p$, set $F_r(B) := S_r(N_l(B_p))$.

5. For every $B \in (B_p, B_r]$ such that $F_l(B) = B_p$, set $F_l(B) := S_l(N_r(B_p))$.

6. Remove B_p from Ψ.

//Join of B_l with B_{l-1} and B_r with B_{r+1} when they are twins.

7. If $F_l(B_{l-1}) = F_l(B_l)$ and $F_r(B_{l-1}) = F_r(B_l)$ then:

8. Move the vertices from B_l to B_{l-1}.

9. Set $S_r(B_{l-1}) := S_r(B_l)$ and remove B_l from Ψ.

10. If $F_l(B_{r+1}) = F_l(B_r)$ and $F_r(B_{r+1}) = F_r(B_r)$ then:

11. Move the vertices from B_r to B_{r+1}.

12. Set $S_l(B_{r+1}) := S_l(B_l)$ and remove B_r from Ψ.

//Join of the universal semiblocks.

13. Run Algorithm 6.4 on Ψ and v to obtain the co-components of $H \setminus \{v\}$.

14. If there are two universal semiblocks B_i and B_j, where v is adjacent to B_i then:

15. Apply Algorithm 6.5 to remove B_i from Ψ and set $B_j := B_j \cup B_i$.

removes this semiblock by invoking Algorithm 6.5 and then it moves all the vertices from this block to the other universal semiblock.

It is not hard to see that every step of Algorithm 6.9 is executed in $O(d(v))$ time. In particular, observe that there are at most two universal semiblocks in Ψ at every step of the algorithm. So, vertex v is removed from Ψ in $O(d(v))$ time.

The round block enumeration Φ of $H \setminus \{v\}$ that is computed by Algorithm 6.9 may violate the dynamic algorithm invariant, since Φ may be non-straight while $H \setminus \{v\}$ may be a PIG graph. So, we need to transform Φ into a straight enumeration when $H \setminus \{v\}$ is a PIG graph. Suppose that $H \setminus \{v\}$ is in fact a PIG graph. Recall that, by Theorem 3.3.4,

every PHCA model of a PIG graph is an interval model. In terms of round enumerations, if Φ is locally straight, then Φ is also straight. Otherwise, Φ has some directed triangle and, as in Theorem 3.2.10, one of the blocks in this triangle is the universal block. The lemma below extracts all the important information of Theorem 3.2.10 that we require, translated to the language of enumerations. Observe that in the lemma, both $B_1, \ldots, B_{i-1}$ and $B_{i+1}, \ldots, B_k$ are ranges of co-components of $G \setminus B_i$.

Lemma 6.5.1 (Corollary of Theorem 3.2.10)**.** *Let $\Phi = B_1, \ldots, B_k$ be a ring of a PIG graph G. If Φ is not straight, then there is some universal block $B_i \in \Phi$ such that $F_r(B_{i-1}) = B_i = F_l(B_{i+1})$. Furthermore, $\langle [B_{i-1}, B_1], B_i, [B_k, B_{i+1}], \emptyset \rangle$ is a contig of G.*

If the input Ψ of Algorithm 6.9 is straight, then the output Φ is also straight. When Ψ is not straight, then Φ is straight if and only if $N_r(B_p)$ is the leftmost end block of $\Psi \setminus \{B_p\}$. Certainly, we can check this condition in $O(1)$ time. When Φ is not straight then, by the above lemma, $H \setminus \{v\}$ is a PIG graph only if Φ contains some universal block B_i that was computed at Step 13 of Algorithm 6.9. When there is such a universal block, we can check the other condition of the above lemma to determine if $H \setminus \{v\}$ is PIG. If affirmative, then $\Phi = \langle [B_1, B_{i-1}], B_i, [B_{i+1}, B_k], \emptyset \rangle$, where $B_1 = F_l(B_i)$ and $B_k = F_r(B_i)$. So, we need only to compute $\Gamma = \langle [B_{i+1}, B_k], B_i, [B_1, B_{i-1}], \emptyset \rangle$, which is a contig of $H \setminus \{v\}$ by Lemma 6.5.1. (The enumeration that appears in the statement of Lemma 6.5.1 is the reverse of Γ which can be computed from the reverse of Φ.) The computation of Γ takes $O(1)$ time as it was discussed in Section 6.3.

Finally, we have to update the PHCA flag when B_p is removed from Ψ. By definition, $\Psi \setminus \{B_p\}$ is a round semiblock enumeration of $H \setminus \{v\}$. By Corollary 3.2.12, $H \setminus \{v\}$ is a PHCA graph if and only if $H \setminus \{v\}$ is a PIG graph or $\Psi \setminus \{B_p\}$ is locally straight. We already discussed how to detect whether $H \setminus \{v\}$ is a PIG graph or not. The following lemma can be used to test in $O(d(v))$ time whether $\Psi \setminus \{B_p\}$ is locally straight.

Proposition 6.5.2. *Let $\Psi = B_1, \ldots, B_k$ be a round semiblock enumeration of a graph H, and suppose that $B_p = \{v\}$, for some $1 \leq p \leq k$. Then $\Psi \setminus \{B_p\}$ has a directed triangle if and only if there is some block $B \in [F_l(B_p), B_p)$ such that $F_r(F_r(F_r(B))) \in [B, F_r(B_p)]$.*

Proof. Let B_a, B_b, B_c be a directed triangle of $\Psi \setminus \{B_p\}$. Without loss of generality, suppose that $B_p \in (B_a, B_b)$ and let $B_x = F_r(B_a)$, $B_y = F_r(B_x)$ and $B_z = F_r(B_y)$. Since $B_a \longrightarrow B_b$ and $B_c \longrightarrow B_a$, then $B_x \in [B_b, B_c)$. On the other hand, since $B_a \longrightarrow B_p$, then $B_x \in [B_p, F_r(B_p)]$, thus $B_x \in [B_b, F_r(B_p)]$. Similarly, since $B_a \longrightarrow B_x$ and $B_x \longrightarrow B_c$, then $B_y \in [B_c, B_a)$, and $B_y \longrightarrow B_a$ because $B_c \longrightarrow B_a$. Finally, since $B_y \longrightarrow B_a$ and $B_x \longrightarrow B_y$, then $B_z \in [B_a, B_x]$, hence $B_z \in [B_a, F_r(B_p)]$. Summing up, $B_a \in [F_l(B_p), B_p)$ and $F_r(F_r(F_r(B_a))) \in [B_a, F_r(B_p)]$.

For the converse, let $B_a \in [F_l(B_p), B_p)$, and call $B_x = F_r(B_a)$ and $B_y = F_r(B_x)$. If $F_r(B_y) \in [B_a, F_r(B_p)]$, then $B_y \longrightarrow B_a$, thus B_a, B_x, B_y is a directed triangle. □

As a consequence of this section, we can update the round block enumeration of H into one of $H \setminus \{v\}$ in $O(d_H(v))$ time, for any $v \in V(H)$. When $H \setminus \{v\}$ is a PIG graph, the enumeration so obtained is straight. After the update, the queries of whether $H \setminus \{v\}$ is a PIG graph or a PHCA graph can be solved in $O(1)$ time.

6.6 The incremental algorithm for PHCA graphs

In this section we consider the edge insertion problem for PHCA graphs. That is, given two non-adjacent vertices v and w from a PHCA graph G, we have to update the representation of G into a representation of $G \cup \{vw\}$, whenever possible. Let $\Phi = B_1, \ldots, B_k$ be a locally straight block enumeration of G, and suppose that $v \in B_p$ and $w \in B_q$. The algorithm for inserting vw into Φ is divided into four parts. First, we consider the case in which Φ is straight and $N(B_p) \cap N(B_q) \neq \emptyset$. Next, we consider the case in which Φ is straight and $N(B_p) \cap N(B_q) = \emptyset$. The third considered case is when Φ is not straight and both $F_r(B_p) \longrightarrow B_q$ and $F_r(B_q) \longrightarrow B_p$. Finally, we consider the case in which Φ is not straight and $F_r(B_q) \not\longrightarrow B_p$. Before starting, we discard the case in which G is disconnected with the following lemma.

Lemma 6.6.1. *Let G be a PHCA graph and v, w be two non-adjacent vertices of G. If G is disconnected, then $G \cup \{vw\}$ is a PHCA graph if and only if $G \cup \{vw\}$ is a PIG graph.*

Proof. Suppose that $G \cup \{vw\}$ is a PHCA graph but not a PIG graph. Then, $G \cup \{vw\}$ has some hole C, and every vertex of $G \cup \{vw\}$ has some neighbor in C because $G \cup \{vw\}$ is a circular-arc graph. Consequently, every vertex has some neighbor in $C \setminus \{vw\}$, thus G is connected. □

The first case in which G is a PIG graph and $N(v) \cap N(w) \neq \emptyset$ is similar to the case in which G is disconnected.

Lemma 6.6.2. *Let G be a PIG graph and v, w be two non-adjacent vertices of G. If $N(v) \cap N(w) \neq \emptyset$, then $G \cup \{vw\}$ is a PHCA graph if and only if $G \cup \{vw\}$ is a PIG graph.*

Proof. Suppose that $G \cup \{vw\}$ is a PHCA graph but not a PIG graph. Then, $G \cup \{vw\}$ contains some hole C that must contain vw since G is a PIG graph. Consequently, in G there is an induced path P with at least four vertices between v and w. Let $u \in N(v) \cap N(w)$ and call ϕ to a straight enumeration of $P \cup \{u\}$. By Theorem 3.3.2, there are exactly two straight enumerations of P, and the extreme vertices of these straight enumerations are v and w. So, u has to be adjacent to all the vertices of P,

or otherwise ϕ is not straight. But then $P \cup \{u\}$ induces a k-wheel in $G \cup \{vw\}$, in contradiction to Corollary 3.2.11. □

The second case in which G is a PIG graph and $N(u) \cap N(v) = \emptyset$ is analyzed below. Recall that we assume that B_1 and B_k are respectively the leftmost and rightmost end blocks of a contig $B_1, \ldots, B_k$.

Lemma 6.6.3. *Let $\Phi = B_1, \ldots, B_k$ be a contig of a graph G, $v \in B_p$, and $w \in B_q$, for $1 \leq p < q \leq k$. If $N[B_p] \cap N[B_q] = \emptyset$, then $G \cup \{vw\}$ is a PHCA graph if and only if $p = 1$ and $q = k$. Furthermore, if $G \cup \{vw\}$ is a PHCA graph then $\Psi = \Psi_1, (B_1, B_k), \Psi_k$ is a locally straight ring of $G \cup \{vw\}$, where Ψ_1 and Ψ_k are defined by the rules below.*

(i) If $B_1 = \{v\}$, then $\Psi_1 = \{v\}$; otherwise, $\Psi_1 = \{v\}, B_1 \setminus \{v\}$.

(ii) If $B_k = \{w\}$, then $\Psi_k = \{w\}$; otherwise, $\Psi_k = B_k \setminus \{w\}, \{w\}$.

Proof. Suppose that $p > 1$. Call $R_0 = B_i$ and $R_{i+1} = F_r(P_i)$, for every $i \geq 1$. Since G is connected, there is a value $j \geq 1$ such that $R_j \longrightarrow B_q$ and $R_{j-1} \not\longrightarrow B_q$. By hypothesis, $R_1 \not\longrightarrow B_q$, so $j > 1$ and $B_p, R_1, \ldots, R_j, B_q$ is an induced path of G with at least four blocks. Also, $B_1 \not\longrightarrow B_q$ and $B_1 \not\longrightarrow R_i$ (for $i = 1, \ldots, j$) since otherwise B_1 would be a twin of B_p. If $B_1 \longrightarrow B_p$, then v and w together with a vertex in B_1 and a vertex in R_1 induce a $K_{1,3}$ in $G \cup \{vw\}$. Otherwise, v and w together with the set $\{u \in R_i \mid 1 \leq i \leq j\}$ induce a hole C in $G \cup \{vw\}$, and no vertex of B_1 has a neighbor in C. Therefore, $G \cup \{vw\}$ is not a PCA graph. Similarly, if $q < k$ then $G \cup \{vw\}$ is not a PCA graph.

For the converse, suppose that $p = 1$ and $q = k$. All we have to do is to show that Ψ is a locally straight ring of $G \cup \{vw\}$. Clearly, Ψ is a circular ordering of the blocks of $G \cup \{vw\}$. Let $B \in \Phi$ be such that $B \neq \{v\}$ and $B \neq \{w\}$, and define F_l^Ψ and F_r^Ψ with the rules below.

- If $F_l^\Phi(B) \neq B_1$ then $F_l^\Psi(B \setminus \{w\}) = F_l^\Phi(B)$.
- If $F_l^\Phi(B) = B_1$ then $F_l^\Psi(B \setminus \{v\}) = \{v\}$.
- $F_l^\Psi(\{v\}) = \{w\}$.
- If $F_r^\Phi(B) \neq B_k$ then $F_r^\Psi(B \setminus \{v\}) = F_r^\Phi(B)$.
- If $F_r^\Phi(B) = B_k$ then $F_r^\Psi(B \setminus \{w\}) = \{w\}$.
- $F_r^\Psi(\{w\}) = \{v\}$.

It is not hard to see that F_l^Ψ and F_r^Ψ define a round orientation of the blocks of $G \cup \{vw\}$, so Ψ is a ring of $G \cup \{vw\}$ (see Figure 6.8). Moreover, $\{v\}$ and $\{w\}$ belong to all the directed cycles of blocks in Ψ. Hence, Ψ is locally straight because there is no block B in Ψ such that $\{v\} \longrightarrow_\Psi B \longrightarrow_\Psi \{w\}$ or otherwise $B \in N[B_1] \cap N[B_k]$ in Φ. □

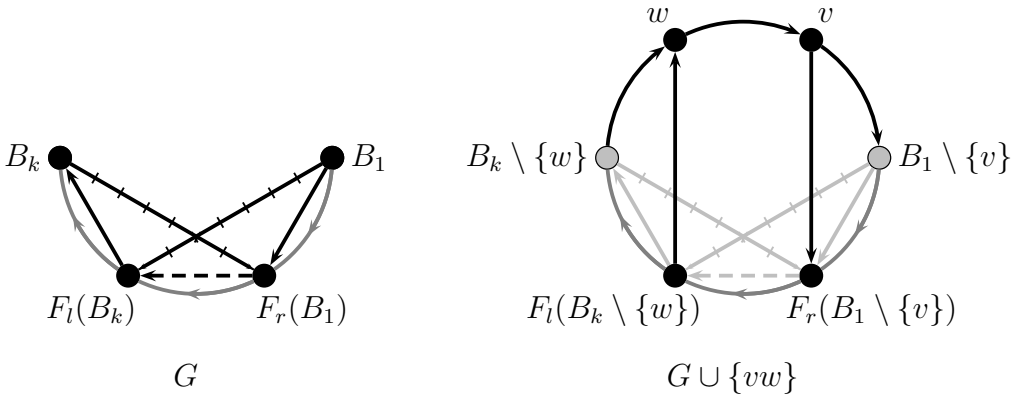


Figure 6.8: Insertion of an edge vw as in Lemma 6.6.3. Note that $B_1 \setminus \{v\}$ and $B_k \setminus \{v\}$ could be empty.

For the following case it is better to work with rings that may contain empty blocks. This is a contradiction by itself, because there are no empty blocks. So, we need to use some sort of technical trick that we have avoided so far. Given an ordering Ψ of sets, each of which is either a block or an empty set, we can filter the blocks of Ψ by removing $\emptyset$ from Ψ. That is, we can obtain an ordering of blocks by taking $\Psi \setminus \{\emptyset\}$. This is what we do in the next lemma.

Lemma 6.6.4. *Let Φ be a locally straight ring of a graph G, B_p and B_q be two non-adjacent blocks of Φ, $v \in B_p$, and $w \in B_q$. If $F_r(B_p) \longrightarrow B_q$ and $F_r(B_q) \longrightarrow B_p$ then $G \cup \{vw\}$ is a PHCA graph if and only if $V(G)$ can be partitioned into six sets $B_1, \ldots, B_6$ that satisfy conditions (i) to (vi). Furthermore, if $G \cup \{vw\}$ is a PHCA graph, then $\Psi \setminus \{\emptyset\}$ is a contig of $G \cup \{vw\}$, where Ψ is the linear ordering $B_6, B_5, B_1, B_4, B_2, B_3$.*

- *(i) B_1, B_2, B_4, and B_5 are blocks of G, while B_3 and B_6 are either blocks or empty sets.*
- *(ii) $B_1 = \{v\}$ and $B_4 = \{w\}$.*
- *(iii) B_2 and B_5 are adjacent to both B_1 and B_4.*
- *(iv) If $B_3 \neq \emptyset$, then B_3 is adjacent to both B_2 and B_4.*
- *(v) If $B_6 \neq \emptyset$, then B_6 is adjacent to both B_1 and B_5.*
- *(vi) There are no more adjacencies.*

Proof. Clearly, if $V(G)$ can be partitioned into six sets that satisfy conditions (i) to (vi), then $\Psi \setminus \emptyset$ is a contig of $G \cup \{vw\}$ (see Figure 6.9).

For the converse, suppose that $G \cup \{vw\}$ is a PHCA graph, and let $B_l = F_l(B_q)$, $B_r = F_r(B_p)$, $B_a = F_l(B_p)$, and $B_b = F_r(B_q)$. For the proof we will make some observations

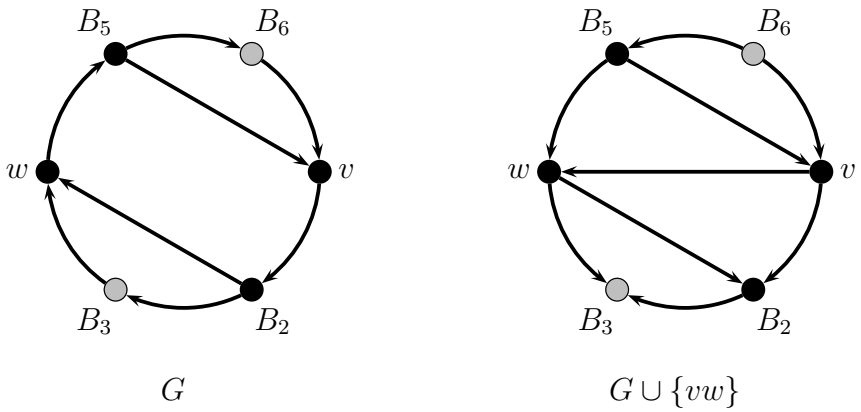


Figure 6.9: Insertion of an edge vw as in Lemma 6.6.4. In this case, B_3 and B_6 could be empty.

that we then combine to show that Φ is equal to $[B_1, B_6] \setminus \emptyset$, where B_1–B_6 are defined by the rules (i) to (vi).

Claim 1: no block in (B_p, B_q) is adjacent to a block in (B_q, B_p). By hypothesis $B_r \longrightarrow B_q$, so B_p, B_l, B_r, and B_q appear in this order in Φ, where possibly $B_l = B_r$. Similarly, B_q, B_a, B_b and B_p appear in this order in Φ, where possibly $B_a = B_b$. Then, no block in (B_p, B_l) is adjacent to a block in (B_q, B_a) and no block in (B_b, B_p) is adjacent to a block in (B_r, B_q). Also, $B_r \not\longrightarrow B_a$ or otherwise B_p, B_r, B_a would form a directed triangle. Similarly, $B_b \not\longrightarrow B_l$, thus no block in $[B_l, B_r]$ is adjacent to a block in $[B_a, B_b]$. If $B \in (B_b, B_p)$ is adjacent to B_l, then $B, B_l, \{w\}, B_b$ is a hole of semiblocks in $G \cup \{vw\}$, and the semiblock $\{v\}$ is adjacent to all these semiblocks in $G \cup \{vw\}$, a contradiction. So, no block in $[B_a, B_p)$ is adjacent to a block in $[B_l, B_q)$ and, similarly, no block in $(B_p, B_r]$ is adjacent to a block in $(B_q, B_b]$. Finally, if $B \in (B_b, B_p)$ is adjacent to $B' \in (B_p, B_l)$, then $B, B', B_l, \{w\}, B_b$ is an induced hole of semiblocks in $G \cup \{vw\}$, and the semiblock $\{v\}$ is adjacent to all these semiblocks in $G \cup \{vw\}$ which is impossible. Similarly, no block in (B_r, B_q) is adjacent to a block in (B_q, B_a). Summing up, no block in (B_p, B_q) is adjacent to a block in (B_q, B_p).

Claim 2: B_p and B_q are singletons, *i.e.*, $B_p = \{v\}$ and $B_q = \{w\}$. To obtain a contradiction, suppose that B_p contains a vertex $u \neq v$. By Claim 1, B_r is not adjacent to B_b, so $\{u, v\}, B_r, \{w\}$ and B_b induce a hole of semiblocks in G. But then, $\{v\}, \{u\}, B_r, \{w\}, B_b$ is an induced wheel of semiblocks in $G \cup \{vw\}$, a contradiction.

Claim 3: if $(B_p, B_l) \neq \emptyset$, then $(B_b, B_p) = \emptyset$. Suppose not, so there is a block $B \in (B_p, B_l)$ and there is another block $B' \in (B_b, B_p)$. By definition, $(B_b, B_l) = \overline{N}(B_q)$, so neither B nor B' are adjacent to B_q. Also, by Claim 1, B is not adjacent to B', so

$\{v\}, B, \{w\}, B'$ form an induced $K_{1,3}$ of semiblocks in $G \cup \{vw\}$, a contradiction.

Claim 4: if $(B_q, B_a) \neq \emptyset$, then $(B_r, B_q) = \emptyset$. The proof is analogous to Claim 3.

Claim 5: if $(B_p, B_l) \neq \emptyset$, then $(B_r, B_q) = \emptyset$. Suppose not, so there is a block $B \in (B_p, B_l)$ and there is another block $B' \in (B_r, B_q)$. If B is adjacent to B' then $\{v\}, B, B', \{w\}$ is a hole of semiblocks in $G \cup \{vw\}$ and B_l is adjacent to all these semiblocks in $G \cup \{vw\}$, a contradiction. When B is not adjacent to B' then $\{v\}, B, B_l, B', \{w\}, B_a$ induce a 3-sun of semiblocks in $G \cup \{vw\}$ since, by Claim 1, B_a is not adjacent to the blocks in (B_p, B_q). This is also impossible.

Claim 6: if $(B_q, B_a) \neq \emptyset$, then $(B_b, B_p) = \emptyset$. The proof is analogous to Claim 5.

By Claim 2, $B_p = \{v\}$ and $B_q = \{w\}$. By Claim 1, $F_l(B_l) = F_l(B_r) = B_p$ and $F_r(B_l) = F_r(B_r) = B_q$, so $B_l = B_r$ because both B_l and B_r are blocks. Similarly, $B_a = B_b$. By Claims 3–6, either (B_p, B_l) and (B_q, B_a) are empty or (B_l, B_q) and (B_a, B_p) are empty. Without loss of generality, assume that (B_p, B_l) and (B_q, B_a) are empty. But then, $F_l(B) = B_l$ for every block $B \in (B_l, B_q)$ while, by Claim 1, $F_r(B) = B_q$ for every $B \in (B_l, B_q)$. This implies that (B_l, B_q) is either empty or it has exactly one block. Similarly, (B_a, B_p) is either empty or it has exactly one block.

Define $B_1 = B_p$, $B_2 = B_l$, $B_4 = B_q$ and $B_5 = B_a$. If $(B_l, B_q) = \emptyset$, then define $B_3 = \emptyset$, otherwise define B_3 as the unique block in (B_l, B_q). Finally, if $(B_a, B_p) = \emptyset$, then define $B_6 = \emptyset$, otherwise define B_6 as the unique block in (B_a, B_p). Sets $B_1, \ldots, B_6$ form a partition of $V(G)$ that satisfies the conditions (i) to (vi). Moreover, $\Phi = [B_1, B_6] \setminus \emptyset$ (see Figure 6.6.4). □

The last case is very similar to Lemma 4.2 in [HSS01].

Lemma 6.6.5. *Let Φ be a locally straight ring of a graph G, B_p and B_q be two non-adjacent blocks of Φ, and $v \in B_p$ and $w \in B_q$. If Φ is not straight and $F_r(B_q) \not\longrightarrow B_p$, then $G \cup \{vw\}$ is a PHCA graph if and only if $F_r(B_p) = B_{q-1}$ and $F_l(B_q) = B_{p+1}$.*

Proof. Suppose that $F_r(B_p) \neq B_{q-1}$. By hypothesis $B_p \not\longrightarrow B_q$, so $F_r(B_p) \in [B_p, B_{q-1})$ and $F_l(B_{q-1}) \in (B_p, B_{q-1}]$. Consider the following alternatives for the rightmost neighbor of B_{p+1}.

Case 1: $B_{p+1} \longrightarrow B_q$. In this case, $F_l(B_{q-1}) = F_l(B_q) = B_{p+1}$ since $B_p \not\longrightarrow B_{q-1}$. Then, B_{q-1} is not adjacent to $F_r(B_q)$ and neither is B_p because $F_r(B_q) \not\longrightarrow B_p$. Consequently, $\{w\}, \{v\}, B_{q-1}$ and $F_r(B_q)$ form an induced $K_{1,3}$ of semiblocks in $G \cup \{vw\}$, so $G \cup \{vw\}$ is not a PCA graph.

Case 2: $B_p \longrightarrow F_r(B_{p+1})$. In this case, $B_{p+1} \not\longrightarrow B_q$, so B_{p+1} and B_q are not adjacent. As before, B_{p+1} is not adjacent to $F_l(B_p)$ and neither is B_q because $F_r(B_q) \not\longrightarrow B_p$. Then, $\{v\}, \{w\}, B_{p+1}, F_l(B_{p+1})$ form an induced $K_{1,3}$ of semiblocks in $G \cup \{vw\}$, so $G \cup \{vw\}$ is not a PCA graph.

Case 3: $B_{p+1} \not\longrightarrow B_q$ and $B_p \not\longrightarrow F_r(B_{p+1})$. Since Φ is not straight, there is an induced path of blocks between $F_r(B_{p+1})$ and B_q whose blocks all belong to $[F_r(B_{p+1}), B_q]$. Then, Φ contains an induced path $\mathcal{P} = B_p, B_{p+1}, \mathcal{P}', B_a, B_q$ that has at least four blocks and such that $\mathcal{P} \subseteq [B_p, B_q]$. On the other hand, since Φ is not straight and $F_r(B_q) \not\longrightarrow B_p$, Φ contains also an induced path $\mathcal{Q} = B_q, B_b, \mathcal{Q}', B_c, B_p$ that contains at least four blocks and such that $\mathcal{Q} \subseteq [B_q, B_p]$. If B_{p+1} is not adjacent to B_c, then $\{v\}, \{w\}, B_{p+1}, B_c$ form an induced $K_{1,3}$ of semiblocks in $G \cup \{vw\}$, hence $G \cup \{vw\}$ is not a PCA graph. Similarly, if B_a is not adjacent to B_b, then $\{w\}, \{v\}, B_a, B_b$ form an induced $K_{1,3}$ of semiblocks in $G \cup \{vw\}$. Suppose, for the last case, that B_{p+1} is adjacent to B_c and B_a is adjacent to B_b. If both $\mathcal{P}'$ and $\mathcal{Q}'$ are empty, then $\{v\}, B_{p+1}, B_a, \{w\}, B_b, B_c$ induce a $\overline{C_6}$ in $G \cup \{vw\}$, so $G \cup \{vw\}$ is not a PCA graph. If $\mathcal{Q}' \neq \emptyset$, then $\{v\}, B_{p+1}, \mathcal{P}', B_a, \{w\}$ is an induced hole of $G \cup \{vw\}$ and the blocks in $\mathcal{Q}'$ are not adjacent to the blocks of this hole, so $G \cup \{vw\}$ is not a PCA graph. Similarly, if $\mathcal{P}' \neq \emptyset$, then $G \cup \{vw\}$ is not a PCA graph.

Summing up, if $F_r(B_p) \neq B_{q-1}$, then $G \cup \{vw\}$ is not a PCA graph. The proof that $G \cup \{vw\}$ is not a PCA graph when $F_l(B_q) \neq B_{p+1}$ is analogous.

For the converse, suppose that $F_r(B_p) = B_{q-1}$ and $B_{p+1} = F_l(B_q)$. A locally straight ring of $G \cup \{vw\}$ can be obtained as in Lemma 4.2 of [HSS01]. For completeness, we show how can Φ be transformed into such a ring. Apply the following operations in order.

1. If $|B_p| = 1$, $F_r(B_q) = F_r(B_{q-1})$, and $F_l(B_{q-1}) = B_p$, then insert w into B_{q-1}. Otherwise, insert w into a new block immediately to the left of B_q.
2. If $|B_q| = 1$, $F_l(B_p) = F_l(B_{p+1})$, and $F_r(B_{p+1}) = B_q$, then insert v into B_{p+1}. Otherwise, insert v into a new block immediately to the right of B_p.
3. Remove v from B_p. If $B_p = \emptyset$, then remove B_p from Φ.
4. Remove w from B_q. If $B_q = \emptyset$, then remove B_q from Φ.

The resulting ordering of semiblocks Ψ is a ring of $G \cup \{vw\}$. The details are the same as in Lemma 4.2 of [HSS01]. Observe that the new blocks that now contain v and w do not belong to a directed triangle because otherwise there would be a block B such that $B_q \longrightarrow B \longrightarrow B_p$, contradicting the lemma's hypothesis. So, Φ is locally straight. □

6.6.1 The impact of a new edge

In this section we describe an algorithm that can be used to update the representation of the PHCA graph G when a new edge vw is inserted. The first step of the algorithm is to test if G is connected or not. Recall that G is connected if and only if its representation Φ is composed by a unique ring. If G is not connected, then $G \cup \{vw\}$ is a PHCA graph

if and only if $G \cup \{vw\}$ is a PIG graph, by Lemma 6.6.1. So, in this case we can test in $O(1)$ time whether $G \cup \{vw\}$ is a PHCA graph by running the HSS algorithm on Φ. Furthermore, Φ is updated into a representation Ψ of $G \cup \{vw\}$, when $G \cup \{vw\}$ is a PHCA graph. For the rest of this section we consider that G is connected, so Φ is a ring. Let B_p and B_q be the blocks of Φ that contain v and w, respectively.

The second step of the algorithm is to test whether Φ is straight or not, so as to evaluate which of the lemmas in the previous section applies for Φ. We have already discussed how can we test in $O(1)$ time whether a ring of G is straight or not. According to the output of this test, there are two alternatives.

Case 1: $\Phi = B_1, \ldots, B_k$ is straight. By exchanging v and w if required, assume that $p < q$. Suppose first that $1 < p$ or $q < k$, *i.e.*, one among B_p, B_q is not an end block. By Lemma 6.6.3, $G \cup \{vw\}$ is a PHCA graph only if $N(B_p) \cap N(B_q) = \emptyset$. So, by Lemma 6.6.2, $G \cup \{vw\}$ is a PHCA graph only if $G \cup \{vw\}$ is a PIG graph. We can figure out in $O(1)$ time whether $G \cup \{vw\}$ is a PIG graph by running the HSS algorithm on Φ. If the HSS algorithm is successful, then Φ is updated into a contig of $G \cup \{vw\}$. When the HSS algorithm is unsuccessful, we can conclude that $G \cup \{vw\}$ is not a PHCA graph. Suppose now that $p = 1$ and $q = k$. In this case, we can determine whether $N(B_p) \cap N(B_q) = \emptyset$ by testing whether $F_r(B_p) \longrightarrow B_q$. This test takes $O(1)$ time because $F_r(B_p) \longrightarrow B_q$ if and only if $F_r(F_r(B_1)) = B_k$. Again, by Lemma 6.6.2, if $F_r(B_p) \longrightarrow B_q$ then $G \cup \{vw\}$ is a PHCA graph if and only if $G \cup \{vw\}$ is a PIG graph. So, we can also call the HSS algorithm for this case. Finally, if $F_r(B_p) \not\longrightarrow B_k$, then $G \cup \{vw\}$ is PHCA by Lemma 6.6.3. Moreover, we can update Φ in $O(1)$ time into the representation Ψ of $G \cup \{v\}$ that appears in Lemma 6.6.3, as follows.

- **Near pointers.** If $|B_1| > 1$, then insert a new block $B_0 = \{v\}$ and set $N_r(B_0) := B_1$; otherwise, let $B_0 = B_1$. If $|B_k| > 1$, then insert a new block $B_{k+1} = \{w\}$ and set $N_l(B_{k+1}) = B_k$; otherwise, let $B_{k+1} = B_k$. Finally, set $N_l(B_0) := B_{k+1}$ and $N_r(B_{k+1}) := B_0$.
- **Far pointers.** Set $F_r(B_0) := F_r(B_1)$, $F_l(B_{k+1}) := F_l(B_k)$, $F_l(B_0) := S_l(B_{k+1})$, and $F_r(B_{k+1}) := S_r(B_0)$.
- **Self pointers.** If B is a block such that $F_l^{\phi}(B) = B_1$, then the far pointer of B has to be updated so as to point to $S_l(B_0)$ in Ψ. All these updates can be done in $O(1)$ time with the technique of nested pointer, by exchanging $S_l(B_0)$ and $S_l(B_1)$. Similarly, all the blocks whose right far pointer references B_k have to be updated so as to reference B_{k+1}. For this, swap $S_r(B_{k+1})$ with $S_r(B_k)$.
- **End pointers.** Set the end pointers of B_1 and B_k to NULL.

Case 2: $\Phi = B_1, \ldots, B_k$ is not straight. The first step of the algorithm in this case is to test whether Φ satisfies the conditions of Lemma 6.6.4. This check can be done in $O(1)$ time because Φ satisfies the conditions of Lemma 6.6.4 only if Φ has at most six blocks. When Φ satisfies the conditions of Lemma 6.6.4, $G \cup \{vw\}$

is a PHCA graph, and it is trivial to update Φ into a contig of $G \cup \{vw\}$ as in Lemma 6.6.4. When Φ does not satisfy the conditions of Lemma 6.6.4, $G \cup \{vw\}$ is a PHCA graph only if $F_r(B_p) \not\longrightarrow B_q$ or $F_r(B_q) \not\longrightarrow B_p$, by Lemma 6.6.4. If neither $F_r(B_p) = B_{q-1}$ and $F_l(B_q) = B_{p+1}$, nor $F_r(B_q) = B_{p-1}$ and $F_l(B_p) = B_{q+1}$, then we can conclude that $G \cup \{vw\}$ is not PHCA, by Lemma 6.6.5. We can check these conditions in $O(1)$ time by using the near and far pointers as usual. Suppose that $F_r(B_p) = B_{q-1}$ and $F_l(B_q) = B_{p+1}$, thus $F_r(B_p) \longrightarrow B_q$. Under these conditions, $G \cup \{vw\}$ is a PHCA graph if and only if $F_r(B_q) \not\longrightarrow B_p$, by Lemmas 6.6.4 and 6.6.5. Since Φ is locally straight then $F_r(B_q) \not\longrightarrow B_{p+1}$ or, otherwise, $B_q, F_r(B_q)$ and B_{p+1} would form a directed triangle in Φ. Then, $F_r(B_q) \longrightarrow B_p$ if and only if $F_r(F_r(B_q)) = B_p$. This condition is easily tested in $O(1)$ time and, when $F_r(B_q) \not\longrightarrow B_p$, we can update Φ into a ring of $G \cup \{vw\}$ in $O(1)$ time as in [HSS01].

Summing up, the update of Φ into a locally straight enumeration of $G \cup \{vw\}$ takes $O(1)$ time when $G \cup \{vw\}$ is a PHCA graph and Φ is equipped with the end pointers (the end pointers are used by the HSS algorithm). So, the incremental PHCA recognition problem, both for vertices and edges, can be solved in $O(1)$ time per edge inserted. When the end pointers of Φ are replaced with the connectivity structure then the insertion takes $O(\log n)$ time.

6.7 The decremental algorithm for PHCA graphs

The last operation that we consider is the removal of an edge from a PHCA graph. That is, given a PHCA graph H and an edge vw, the question is whether $H \setminus \{vw\}$ is a PHCA graph. If affirmative, then the representation Ψ of H has to be updated into a locally straight representation Φ of $H \setminus \{vw\}$. As in the previous section, the decremental algorithm is divided into four cases, according to the existence of common neighbors between v and w. These cases are somehow the inverse versions of the cases in the previous section. We begin with the case in which H is a PIG graph and v and w have no common neighbors.

Lemma 6.7.1. *Let H be a straight graph and v, w be two adjacent vertices of H. If $N(v) \cap N(w) = \emptyset$, then $H \setminus \{vw\}$ is a PHCA graph if and only if $H \setminus \{vw\}$ is a PIG graph.*

Proof. Suppose that $H \setminus \{vw\}$ is a PHCA graph but not a PIG graph, so $H \setminus \{vw\}$ has some induced hole C. Then, since H is a PIG graph, both v and w belong to C and, moreover, vw is the only chord of C in H. This means that C has exactly four vertices, so v and w have at least two common neighbors in H. The converse is trivial. □

The second case is when H is a PIG graph and v and w have some common neighbor.

Lemma 6.7.2. *Let H be a PIG graph and v, w be two adjacent vertices of H. If $N(v) \cap N(w) \neq \emptyset$, then $H \setminus \{vw\}$ is a PHCA graph if and only if:*

(i) $H \setminus \{vw\}$ is straight or

(ii) the vertices of H can be partitioned into six sets $B_1, \ldots, B_6$ that satisfy the conditions (i) to (vi) of Lemma 6.6.4 together with the condition that B_1 is adjacent to B_4. Furthermore, $\Phi = [B_1, B_6] \setminus \{\emptyset\}$ is a locally straight ring of $H \setminus \{vw\}$.

Proof. Suppose that $H \setminus \{vw\}$ is a PHCA graph. Let Φ be a locally straight block enumeration of $H \setminus \{vw\}$, $v \in B_p$, and $w \in B_q$. If Φ is straight, then (i) follows. Suppose, then, that Φ is not straight. If either $F_r(B_p) \not\longrightarrow B_q$ or $F_r(B_q) \not\longrightarrow B_p$, then, by Lemma 6.6.5, $H = (H \setminus \{vw\}) \cup \{vw\}$ admits a non-straight ring. So, by Theorem 3.3.4, H has no straight block enumerations, a contradiction. On the other hand, if both $F_r(B_p) \longrightarrow B_q$ and $F_r(B_q) \longrightarrow B_p$, then, by Lemma 6.6.4, the blocks of $H \setminus \{vw\}$ can be partitioned into six sets that satisfy the conditions of Lemma 6.6.4. Furthermore, by Theorem 3.3.4, Φ is equal to either $[B_1, B_6] \setminus \{\emptyset\}$ or to its reversal, so (ii) follows.

The converse is trivial since $H \setminus \{vw\}$ is a locally straight graph by (i) and (ii). □

The third case is when H is not a PIG graph and v and w have no common neighbors.

Lemma 6.7.3. *Let Ψ be a locally straight ring of a graph H, B_p and B_q be blocks such that either $B_p = B_q$ or $B_q \longrightarrow B_p$, and $v \in B_p$ and $w \in B_q$. If Ψ is not straight and $N(v) \cap N(w) = \emptyset$, then $H \setminus \{vw\}$ is a PIG graph. Furthermore, $[B_p, B_q]$ is a semiblock contig of $H \setminus \{vw\}$ whose semiblocks that are not blocks belong to $\{B_p, B_{p+1}, B_{q-1}, B_q\}$.*

Proof. Suppose that $H \setminus \{vw\}$ admits a locally straight ring Φ, and let B_p^Φ and B_q^Φ be the blocks of Φ that contain v and w, respectively. Certainly, $N[B_p^\Phi] \cap N[B_q^\Phi] = \emptyset$, thus $F_r^\Phi(B_p^\Phi) \neq N_l(B_q^\Phi)$ and $F_r^\Phi(B_q^\Phi) \neq N_l^\Phi(B_p^\Phi)$. Then, by Lemma 6.6.5, Φ must be a contig of $H \setminus \{vw\}$. Therefore, B_p^Φ and B_q^Φ are the end blocks of Φ by Lemma 6.6.3. Furthermore, by reversing Φ if required, we can assume that Ψ is obtained from Φ as in Lemma 6.6.3 since, by Theorem 3.3.4, the locally straight enumeration of H is unique up to full reversal. Then, B_p^Φ is the leftmost end block of Φ, B_q^Φ is the rightmost end block of Φ, and all the blocks of Ψ in (B_p, B_q) are in a one-to-one correspondence with the blocks of Φ in (B_p^Φ, B_q^Φ). That is, $[B_p, B_q]$ is a semiblock contig of $H \setminus \{vw\}$ whose only semiblocks that could not be blocks are B_p and B_{p+1} (when $B_p^\Phi = B_p \cup B_{p+1}$), and B_{q-1} and B_q (when $B_q^\Phi = B_{q-1} \cup B_q$). □

Finally, we consider the case in which H is not a PIG graph and v and w have common neighbors. This is the equivalent of Lemma 5.2 in [HSS01] for locally straight graphs.

Lemma 6.7.4. *Let Ψ be a locally straight ring of a graph H, B_p and B_q be blocks such that either $B_p = B_q$ or $B_p \longrightarrow B_q$, and $v \in B_p$ and $w \in B_q$. If Ψ is not straight and $N(v) \cap N(w) \neq \emptyset$, then $H \setminus \{vw\}$ is a PHCA graph if and only if $F_r(B_p) = B_q$ and $F_l(B_q) = B_p$.*

Proof. Suppose that $H \setminus \{vw\}$ admits a locally straight ring Φ, and let B_p^Φ and B_q^Φ be the blocks of Φ that contain v and w, respectively. By hypothesis, B_p^Φ and B_q^Φ have a common neighbor, so either $F_r^\Phi(B_p^\Phi) \longrightarrow B_q^\Phi$ or $F_r^\Phi(B_q^\Phi) \longrightarrow B_p^\Phi$. If $F_r^\Phi(B_p^\Phi) \longrightarrow B_q^\Phi$ and $F_r^\Phi(B_q^\Phi) \longrightarrow B_p^\Phi$, then H is a PIG graph by Lemma 6.6.4, contradicting the fact that Ψ and its reversal are the unique locally straight rings of H by Theorem 3.3.4. Otherwise, either $F_r^\Phi(B_p^\Phi) = N_l^\Phi(B_q^\Phi)$ and $F_l^\Phi(B_q^\Phi) = N_r^\Phi(B_p^\Phi)$, or $F_r^\Phi(B_q^\Phi) = N_l^\Phi(B_p^\Phi)$ and $F_l^\Phi(B_p^\Phi) = N_r^\Phi(B_q^\Phi)$ by Lemma 6.6.5. Furthermore, since Ψ is the unique ring of H up to full reversal, then we may assume that Ψ is obtained from Φ as in the proof of Lemma 6.6.5. Consequently, since $B_p \longrightarrow_\Psi B_q$, we obtain that $F_r^\Phi(B_p^\Phi) = N_l^\Phi(B_q^\Phi)$ and $F_l^\Phi(B_q^\Phi) = N_r^\Phi(B_p^\Phi)$, so $F_r^\Psi(B_p) = B_q$ and $F_l^\Psi(B_q) = B_p$.

For the converse, suppose that $F_r(B_p) = B_q$ and $F_l(B_q) = B_p$. A ring of $H \setminus \{vw\}$ can be found exactly as in Lemma 5.2 of [HSS01]. For completeness, we show how can Ψ be transformed into such a ring. Apply the following operations in order.

1. If $|B_p| = 1$, $F_l(B_{q+1}) = B_{p+1}$ and $F_r(B_q) = F_r(B_{q+1})$, then insert w into B_{q+1}. Otherwise, insert w into a new block immediately to the right of B_q.
2. If $|B_q| = 1$, $F_l(B_p) = F_l(B_{p-1})$ and $F_r(B_{p-1}) = B_{q-1}$, then insert v into B_{p-1}. Otherwise, insert v into a new block immediately to the right of B_p.
3. Remove v from B_p. If $B_p = \emptyset$, then remove B_p from Ψ.
4. Remove w from B_q. If $B_q = \emptyset$, then remove B_q from Ψ.

The resulting ordering Φ is a ring of $H \setminus \{vw\}$. The details are the same as in Lemma 5.2 of [HSS01]. Observe that no directed triangle could be created by this procedure, so Φ is locally straight. □

The algorithm to remove the edge $\{vw\}$ from H is similar to the incremental algorithm; first we need to find out which of the lemmas above applies, and then we have to modify the data structure accordingly. The first step is to check whether Φ is straight or not. As we already discussed, this test takes $O(1)$. The algorithm then splits in two cases.

Case 1: Ψ is straight. If H has at most six blocks then a ring of $H \setminus \{vw\}$ is easily obtained in $O(1)$ when $H \setminus \{vw\}$ is a PHCA graph. Otherwise, $H \setminus \{vw\}$ is a PHCA graph if and only if $H \setminus \{vw\}$ is a PIG graph, by Lemmas 6.7.1 and 6.7.2. We can figure out in $O(1)$ time if $H \setminus \{vw\}$ is a PIG graph by executing the HSS algorithm on Ψ. If affirmative, then a straight block enumeration of $H \setminus \{vw\}$ is obtained within the same time as a by-product.

Case 2: $\Psi = B_1, \ldots, B_k$ is not straight. Let B_p and B_q be the blocks that contain v and w, respectively. By exchanging v and w, assume that either $B_p = B_q$ or $B_p \longrightarrow B_q$. The first step of the algorithm in this case is to check whether v and w have a common neighbor. We claim that $N(v) \cap N(w) = \emptyset$ if and only if $B_p = \{v\}$, $B_q = \{w\}$, $N_r(B_p) = B_q$, $F_r(B_p) = B_q$, and $F_l(B_q) = B_p$. Indeed, if $B_p = B_q$, then all the vertices in B_{p+1} are common neighbors of v and w, while if $|B_p| > 1$ or $|B_q| > 1$, then v and w have common neighbors in $(B_p \cup B_q) \setminus \{v, w\}$. Similarly, if $B_p \neq B_q$ and $N_r(B_p) \neq B_q$, then all the vertices in B_{p+1} are common neighbors of v and w. When $N_r(B_p) = B_q$ and either $F_l(B_q) \neq B_p$ or $F_r(B_p) \neq B_q$, then v and w have common neighbors in $B_{p-1} \cup B_{q+1}$. Finally, observe that if $N_r(B_p) = B_q$, then $F_r(B_q) \not\longrightarrow B_p$ or, otherwise, B_p, B_q, and $F_r(B_q)$ would form a directed triangle. So, the claim is true, and we can test whether $N(v) \cap N(w) = \emptyset$ in $O(1)$ time. When $N(v) \cap N(w) \neq \emptyset$, we can determine if $H \setminus \{vw\}$ is a PHCA graph by checking whether $F_r(B_p) = B_q$ and $F_l(B_q) = B_p$, by Lemma 6.7.4. If affirmative, then we can update Ψ into a ring of $H \setminus \{vw\}$ in $O(1)$ time as in [HSS01]. Finally, when $N(v) \cap N(w) = \emptyset$ then $H \setminus \{vw\}$ is a PIG graph by Lemma 6.7.3. In this case, we can transform Ψ into a contig of $H \setminus \{vw\}$ as follows.

Near pointers. If $F_l(B_p) = F_l(B_{p-1})$, then move v from B_p to B_{p-1} and set $B_v := B_{p-1}$; otherwise, set $B_v := B_p$. If $F_r(B_q) = F_r(B_{q+1})$, then move w from B_q to B_{q+1} and set $B_w := B_{q+1}$; otherwise, set $B_w := B_q$. Set $N_r(B_p) := B_p$ and $N_l(B_q) := B_q$.

Far pointers. Set $F_r(B_v) := B_v$ and $F_l(B_w) := B_w$.

Self pointers. If B is a block such that $F_r^{\phi}(B) = B_p$, then the far pointer of B has to be updated so as to point to $S_r(B_v)$ in Ψ. Similarly, the right far pointers referencing B_q have to be updated so as to reference B_w. With this purpose, set $S_r(B_v) := S_r(B_p)$ and $S_l(B_w) := S_r(B_q)$.

Blocks. If $B_p = \emptyset$ (*i.e.*, v was moved to B_{p-1}), then remove B_p from Ψ. Similarly, remove B_q form Ψ when $B_q = \emptyset$.

Summing up, the removal of an edge in a PHCA graph takes $O(1)$ time. Thus, the decremental PHCA recognition problem, both for vertices and edges, can be solved in $O(1)$ time per edge removed.

6.8 Maintaining the connected components

Recall that the incremental and decremental algorithms use different data structures. Each block of the incremental data structure is equipped with two end pointers that link together the end blocks of a ring. These pointers are not present in the decremental data structure because their maintenance is expensive when a vertex or edge is removed. A replacement for the end pointers is required to combine the incremental and decremental

algorithms into one fully dynamic algorithm. It turns out that the only purpose of the end pointers is to evaluate whether two end blocks belong to the same ring or not. Is for this reason that Hell et al. replace the end pointers with an efficient connectivity data structure for PIG graphs. In this section we describe this data structure, and we adapt it for the PCA recognition algorithms.

Let G be a graph with a straight enumeration Φ. For the connectivity structure, the HSS algorithm stores a spanning path for each component of the block-reduction of G. Each of these paths is composed by one vertex for each block of the component and two blocks are adjacent in the path if and only if they are consecutive in Φ. That is, the connectivity structure is a graph $B(G)$ with one vertex v_i for each block $B_i \in \Phi$, and two different vertices v_i and v_j are adjacent in $B(G)$ if and only if $N_r^\Phi(B_i) = B_j$ or $N_l^\Phi(B_i) = B_j$. For the implementation, each component of $B(G)$ is stored as a balanced binary tree (e.g. a red-black tree), so the query of whether two blocks belong to the same component can be answered in $O(\log n)$ time.

The crucial observation is that only $O(1)$ vertices and edges are inserted to or removed from $B(G)$, for each of the operations supported by the incremental and decremental algorithms. Indeed, each insertion of a vertex in $B(G)$ corresponds to the creation of a new block in Φ, each removal of a vertex in $B(G)$ corresponds to the deletion of a block from Φ, each insertion of an edge of $B(G)$ corresponds to a new linking of two blocks via the near pointers of Φ, and each removal of an edge of $B(G)$ corresponds to the disappearance of the near pointer link between two blocks. In terms of the balanced trees data structure, all these operations correspond either to the creation and removal of trivial trees, or to the join and split of trees. So, $B(G)$ is updated in $O(\log n)$ time after each operation of the dynamic PIG recognition algorithm.

The adaptation of the connectivity data structure to the PCA case is really simple. Let G be a PCA graph and Φ be a round block enumeration of G. The connectivity data structure in this case is also the graph $B(G)$, defined exactly as for the PIG graphs. That is, there is one vertex per each block and two vertices of $B(G)$ are adjacent when their corresponding blocks are consecutive and adjacent in Φ. As we already argued, $O(1)$ updates of $B(G)$ are required for each operation of the dynamic algorithm when Φ is straight. Going through all the algorithms in this chapter, we can observe that there are also $O(1)$ updates in $B(G)$ for each operation supported by the dynamic algorithm when Φ is not straight. As for the implementation, when Φ is not straight then $B(G)$ is a cycle, so it can be implemented as a path $B(G) \setminus e$ and an edge e. The path $B(G) \setminus e$ is stored, exactly as in the PIG case, as a balanced binary tree. When Φ is straight then e is nullified and $B(G)$ is just as in the HSS algorithm. The following actions are taken each time a new edge e' is inserted into $B(G)$:

- If $B(G)$ is disconnected, then e' is inserted as in the HSS algorithm. Note that in this case e' must be an edge that joins two vertices of different components of $B(G)$.

- If $B(G)$ is connected, then e is set to e'. Observe that in this case $B(G)$ is not a cycle or otherwise e would not be inserted into $B(G)$.

Each time an edge e' is removed, the following actions are taken:

- If e is NULL, then e' is removed as in the HSS algorithm.
- If $e = e'$, then e is set to $NULL$.
- Otherwise, e' is removed from $B(G) \setminus \{e\}$ as in the HSS algorithm, then e is inserted to $(B(G) \setminus \{e\}) \setminus \{e'\}$, and, finally, e is nullified.

The connectivity query can then be answered according to whether e is $NULL$ or not. If $e = NULL$, then the query is answered, as in the HSS algorithm, with a traversal from the vertices of $B(G)$ to the root. If $e \neq NULL$, then Φ is not straight, so G is connected and the answer to the query is true.

Using this connectivity data structure, the vertex-only fully dynamic PCA recognition problem can be solved in $O(d+\log n)$ time per operation, while the fully dynamic PHCA recognition problem can be solved in $O(d+\log n)$ time per vertex operation and $O(\log n)$ time per edge operation.

7 Isomorphism of proper circular-arc graphs

Recall that two graphs are isomorphic when they have the same structure. The *isomorphism problem* is the problem of deciding whether two graphs G and H are isomorphic. Up to this date, no polynomial time algorithm was developed to solve the isomorphism problem, and it seems unlikely to find a proof that the problem is NP-hard [BHZ87, KST93, Sch88]. The conventional wisdom is that this problem lies in a class between the $\mathcal{P}$ and $\mathcal{NP}$-hard classes, if such a class exists. There is even a special complexity class that groups all the problems that are polynomially equivalent to the isomorphism problem.

A *labeled graph* is a graph G in which the vertices are numbers from 1 to n. Two labeled graphs G and H are *equal* when $ij \in E(G)$ if and only if $ij \in E(H)$ for every $1 \leq i, j \leq n$. The *canonization problem* consists of finding, for each graph G, a labeled graph $C(G)$ that is isomorphic to G in such a way that two graphs G and H are isomorphic if and only if $C(G)$ is equal to $C(H)$. The graph $C(G)$ is referred to as the *canonical form* of G. The equality of graphs is easily tested in polynomial time, so solving the canonization problem is enough to solve the isomorphism problem. In fact, many isomorphism testing algorithms build canonical forms of the input graphs.

Although no polynomial time algorithm is known for computing a canonical form of the input graph, there are many algorithms that compute a canonical labeling for random graphs with high probability [BK79, BES80, CP08]. The canonization problem can also be solved in polynomial time for several classes of graphs [GJ79]. In particular, labeled PQ-trees can be used to obtain a canonical form of an interval graph in linear time [LB79], labeled PC-trees can be used to obtain a canonical form of an HCA graph in linear time [Cur07], and a canonical form of a circular-arc graph can be obtained in $O(mn)$ time as in [Hsu95].

In this chapter we present a linear-time algorithm for the isomorphism problem restricted to PCA graphs. Our algorithm takes as input a PCA model $\mathcal{M}$ of a graph G, and it transforms $\mathcal{M}$ into a canonical PCA model of G. This canonical model is obtained by rotating, reflecting and sorting the components and co-components of the input model.

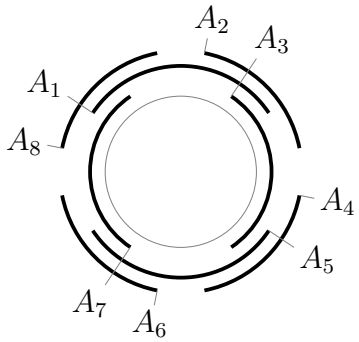
This chapter is organized in four sections. In Section 7.1 we discuss how to encode a PCA model as a string. In Section 7.2 we show how to compute a canonical encoding

of a given PCA model in linear time, while in Section 7.3 we show how to compute a canonical PCA model of a PCA graph. Finally, in Section 7.4 we combine all the previous results to compute a canonical encoding of a canonical PCA model of any PCA graph. The isomorphism problem is then reduced to testing whether two strings that encode PCA graphs are equal or not. In the rest of this section we introduce the terminology that we use for working with strings.

An *alphabet* Σ is a finite set of elements called *symbols*. Symbols are just unspecified objects with an equality operation. A *string* S *(over* Σ*)* is a sequence $s_0, \ldots, s_{|S|-1}$ of symbols of Σ, where $|S|$ is the *length* of S. The elements of $\{0, \ldots, |S|-1\}$ are the *positions* of S, and the symbol s_i is represented by $S(i)$. Recall that, as explained in Chapter 2, we may use negative numbers or numbers greater that $|S|-1$ to refer to the positions of S. In such cases, the position must be understood modulo $|S|$. Also, we extend the range notation to strings, *i.e.*, $[s_i, s_j]$ is the string $s_i, \ldots, s_j$ for every $1 \leq i, j \leq n$. As usual, if $j < i$, then $[s_i, s_j] = [s_i, s_n], [s_1, s_j]$. Given a total order $<$ over Σ, we denote by $<_{\text{lex}}$ the *lexicographic order* between strings, *i.e.*, $S <_{lex} T$ if and only if there exists a position $k \leq \min\{|S|-1, |T|-2\}$ such that $S(i) = T(i)$, for all $1 \leq i \leq k$, and either $k = |S|$ or $S(k+1) < T(k+1)$. For a position i, we represent by $S \ll i$ the i-th rotation of S, *i.e.*, $S \ll i = [s_i, s_{i-1}]$. A position i is *canonical (with respect to* $<$*)* when $S \ll i \leq_{\text{lex}} S \ll j$ for every position j of S. Observe that, since $<$ is a total order, $S \ll i = S \ll j$ for every pair i, j of canonical positions. The *minimum circular string problem* consists of finding every canonical position of S. For this it is enough to find one canonical position i and a period w such that $i + kw$ is a canonical position for every $k \geq 0$. This problem can be solved in $O(n)$ evaluations of $<$ [Boo80, Shi81].

7.1 PCA encodings

In the previous chapters we developed a lot of algorithms on circular-arc models. Except for the dynamic recognition algorithm, we did not discuss how can a circular-arc model be stored in the memory of a RAM machine; we took for granted that one data structure representing a circular-arc model is available, and that many operations were supported by this data structure. For instance, we assumed that a traversal of the extreme points of a circular-arc model takes $O(n)$ time, that given a beginning point we can compute its corresponding ending point in $O(1)$ time, that we can remove an arc from the circular-arc model in $O(1)$ time, etcetera. Of course, it is not difficult to implement a PCA model that supports these operations using basic data structures. However, we cannot expect that the end users of our algorithms provide such a data structure. The input model has to be encoded in the simpler and more general form as possible, and it should only contain the amount of information required so as to guess the positions of the beginning and ending points of each arc. Then, there should be some algorithms that transform these



$$R(A_{2k+1}) = \mathrm{a_1 b_7 b_8 a_2 a_3 b_1 b_2 a_4 a_5 b_3 b_4 a_6 a_7 b_5 b_6 a_8}$$
$$R(A_{2k}) = \mathrm{a_1 a_2 b_8 b_1 a_3 a_4 b_2 b_3 a_5 a_6 b_4 b_5 a_7 a_8 b_6 b_7}$$

$$\langle \mathrm{E(A_{2k+1})}, t(A_{2k+1})\rangle = \langle \mathrm{abbaabbaabbaabba}, 5\rangle$$
$$\langle \mathrm{E(A_{2k})}, t(A_{2k})\rangle = \langle \mathrm{aabbaabbaabbaabb}, 3\rangle$$

Figure 7.1: A circular arc model and all its arc and extreme encodings.

encodings into powerful data structures. In this section we show two simple encodings for circular-arc models that are commonly used for PCA graphs.

Let $\mathcal{M}$ be a circular-arc model with arcs $A_1, \ldots, A_n$ such that $s(A_1), \ldots, s(A_n)$ appear in this order in $\mathcal{M}$. The *arc encoding* $R(A_i, \mathcal{M})$ of $\mathcal{M}$ is the string obtained by traversing $C(\mathcal{M})$ from $s(A_i)$ and writing the symbol '$\mathrm{a_{j+1}}$' and '$\mathrm{b_{j+1}}$' when $s(A_{i+j})$ and $t(A_{i+j})$ are respectively reached (see Figure 7.1). Thus, the j-th beginning point that appears after $s(A_i)$ is designated with the symbol '$\mathrm{a_{j+1}}$' while the j-th ending point that appears after t_i is designated with the symbol '$\mathrm{b_{j+1}}$'. The alphabet used for the encoding of $R(A_i, \mathcal{M})$ has $2n$ symbols and, therefore, $\Theta(n \log n)$ bits are consumed by $R(A_i, \mathcal{M})$.

Usually, the input for the algorithms that work on circular-arc graphs is some sort of linear-time preprocessing of an arc representation; for example, it could be a round enumeration as in [BHH96]. In theory, each circular ordering of the beginning points is unique; the circular ordering $s(A_1)$, ..., $s(A_n)$ is equal to the circular ordering $s(A_2)$, ..., $s(A_n)$, $s(A_1)$ which in turn is equal to the circular ordering $s(A_3)$, ..., $s(A_n)$, $s(A_1)$, $s(A_2)$, and so on. However, in the computer's memory, the extreme points are stored in a fixed linear ordering, corresponding to the string that encodes the data structure. When arc encodings are used, this order is given by the appearance order of the 'a' and 'b' symbols, and algorithms usually assume that $s(A_j)$ corresponds to the symbol '$\mathrm{a_j}$' and $t(A_j)$ corresponds to the symbol '$\mathrm{b_j}$'. So, there could be n different arc encodings of a given circular-arc model $\mathcal{M}$, one for each arc A_i, and an algorithm can make no assumptions on which arc encoding of a model is received as input.

Although arc encodings are asymptotically the most space efficient representation for general circular-arc models, they are not space efficient for the representation of PCA models. In a PCA model, the order of the beginning points is the same as the order of the ending points, so we can reduce the alphabet of the model to just two symbols. For $A \in \mathcal{A}(\mathcal{M})$, the *extreme sequence* $E(A, \mathcal{M})$ of $\mathcal{M}$ is the string that is obtained by replacing in $R(A, \mathcal{M})$ each symbol '$\mathrm{a_j}$' with a symbol 'a' and each symbol '$\mathrm{b_j}$' with a

symbol 'b', for $1 \leq j \leq n$ (see Figure 7.1). In other words, $E(A, \mathcal{M})$ is the string obtained by traversing C from $s(A)$ and writing the symbols 'a' and 'b' when a beginning point and an ending point are respectively reached. The *mark point* $T(A, \mathcal{M})$ of $\mathcal{M}$ is the position where 'b_1' appears in $R(A, \mathcal{M})$, *i.e.*, $T(A, \mathcal{M})$ is the number of extreme points between $s(A)$ and $t(A)$ plus one.

By definition, the extreme sequence $E(A, \mathcal{M})$ and the mark point $T(A, \mathcal{M})$ are obtained from a PCA model $\mathcal{M}$ with a single traversal of $R(A, \mathcal{M})$. The arc encoding $R(A, \mathcal{M})$ can also be obtained from $E(A, \mathcal{M})$ and $T(A, \mathcal{M})$, by applying the inverse algorithm. That is, traverse $E(A, \mathcal{M})$ and replace the j-th symbol 'a' with the symbol 'a_j', and then traverse $E(A, \mathcal{M})$ in a circular manner from $T(A, \mathcal{M})$ and replace the j-th symbol 'b' with the symbol 'b_j'. This decoding is correct because if $s(A_1), \ldots, s(A_n)$ is the order of the beginning points of $\mathcal{M}$, then $t(A_1), \ldots, t(A_n)$ is the order of its ending points. We record this observation for future references.

Remark 7.1.1. For every PCA model $\mathcal{M}$ and every $A \in \mathcal{A}(\mathcal{M})$, the arc encoding $R(A, \mathcal{M})$ can be obtained from $E(A, \mathcal{M})$ and $T(A, \mathcal{M})$ by replacing the j-th symbol 'a' of $E(A, \mathcal{M})$ with the symbol 'a_j' and the j-th symbol 'b' that appears after $T(A, \mathcal{M})$, in a circular traversal of $E(A, \mathcal{M})$, with the symbol 'b_j'.

From now on we will not write the parameter A for R, E, and T when we want to refer to an arc encoding, extreme sequence, and mark point of an unspecified arc. We will also omit the parameter $\mathcal{M}$ when the model is understood. In view of the previous remark, we will refer to the pair $\langle E, T \rangle$ as an *extreme encoding* of $\mathcal{M}$. The advantage of the extreme encoding over the arc encoding for PCA models is that extreme encodings require only $2n + \Theta(\log n)$ bits instead of the $\Theta(n \log n)$ bits required by the arc encoding. Despite this advantage, extreme encodings are not frequently used by those algorithms that work with PCA models for two reasons. First, some operations, as taking the ending point of some arc when the beginning point is given, are hard to implement so that they run in $O(1)$ time without spending the $\Theta(n \log n)$ extra space. The second reason (see *e.g.* [BHH96]) is that the recognition algorithms build a powerful data structure within the same time that is required for the recognition (see Chapter 6).

Some caution about the terminology is required before we move on. By Remark 7.1.1 we can build the arc encoding from the extreme encoding with only $O(n)$ operations. However, the size of the arc encoding is not linear with respect to the size of the extreme encoding, so the algorithm is not linear, it just requires a linear amount of non-constant operations. Our canonization algorithm will use the arc encoding, or even the round enumeration data structure. So, we cannot claim that our isomorphism algorithm runs in linear time when the input models are encoded with extreme encodings. Of course, neither those algorithms that use a round enumeration data structure or an arc encoding for PCA models are strictly linear. Nevertheless, we will use the term linear to refer to those algorithms that require $O(n)$ operations on words of size $O(\log n)$, as it is the usual

meaning of this expression for circular-arc models[1] That is, an algorithm whose input is a circular-arc model is linear if it runs in a time proportional to the size of the arc encoding. We do not consider the problem of finding a succinct data structure for PCA models that supports the operations required in $O(1)$ time.

To end this section, we show how the arc encoding R can be transformed into a round enumeration data structure ϕ of G, and vice versa. Recall that the round enumeration data structure comprises a circular ordering $v_1, \ldots, v_n$ and two functions f_l and f_r such that v_i is adjacent to v_j if and only if $v_i \in [v_j, f_r(v_j)]$ and $v_j \in [f_l(v_i), v_i]$. Algorithm 7.1 can be used to compute the functions f_l and f_r in $O(n)$ time. The arrays computed by Algorithm 7.1 can be transformed into a round enumeration data structure (implemented over doubly-linked circular lists) in $O(n)$ time with the usual techniques.

Algorithm 7.1 Round enumeration of an arc encoding.

Input: An arc encoding R of a PCA model $\mathcal{M}$, for some $A \in \mathcal{A}(\mathcal{M})$.
Output: The functions f_l and f_r that define the round block enumeration corresponding to the intersection graph of $\mathcal{M}$.

1. Define two arrays f_l and f_r with n positions, and an integer $last$.

2. Traverse R from 'a_1' and apply the following operations for each symbol $e \in R$:

3. If $e =$ 'a_j' for some j, then set $last := j$.

4. If $e =$ 'b_j' for some j, then set $f_r(j) := last$.

5. Traverse R in reverse from 'b_1' and apply the following operations for each symbol $e \in R$:

6. If $e =$ 'b_j' for some j, then set $last := j$.

7. If $e =$ 'a_j' for some j, then set $f_l(j) := last$.

Finally, an extreme encoding can be obtained from a round enumeration by running Algorithm 7.2. The construction is based on the proof of Propositions 2.1 and 2.6 in [DHH96], and the time complexity is clearly $O(n)$.

7.2 Canonical encodings of PCA models

Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two circular-arc models. In this chapter we say that $\mathcal{M}_1$ is *equal* to $\mathcal{M}_2$ if there are two arcs $A_1 \in \mathcal{M}_1$ and $A_2 \in \mathcal{M}_2$ such that $R(A_1, \mathcal{M}_1) = R(A_2, \mathcal{M}_2)$.

[1] We have already done this when we discussed the complexity of the algorithm that transforms a PCA model into a PHCA model.

Algorithm 7.2 Extreme encoding of round enumeration.
Input: A round enumeration $\phi = v_1, \ldots, v_n$ of a PCA graph G.
Output: An extreme encoding of $\mathcal{M}$.

1. Define an array E with $2n$ positions, set $p := 1$, and set $s_r(v_i) := 0$ for $1 \leq i \leq n$.

2. For each $v_i \in \phi$, increase $s_r(f_r(v_i))$ by one.

3. For $p := 1, \ldots, n$ do:

4. Set $E[p]$:= 'a' and $E[p+i]$:= 'b' for every $1 \leq i \leq s_r(v_i)$.

5. Set $p := p + s_r(v_i) + 1$.

6. Set $T := 1 + \sum_{v \in [v_1, f_r(v_1))} (s_r(v) + 1)$.

7. Output $\langle E, T \rangle$

We write $\mathcal{M}_1 =_M \mathcal{M}_2$ to indicate that $\mathcal{M}_1$ and $\mathcal{M}_2$ are equal and $\mathcal{M}_1 \neq_M \mathcal{M}_2$ when $\mathcal{M}_1$ and $\mathcal{M}_2$ are not equal. This definition is equivalent to the equality definition given in Chapter 2, and it reflects what one would intuitively assume as equality of models, *i.e.*, do the extremes of $\mathcal{M}_1$ and $\mathcal{M}_2$ appear in the same order? Clearly, if two circular-arc models are equal then their intersection graphs are isomorphic, but the converse is not always true. The question of whether $\mathcal{M}_1$ and $\mathcal{M}_2$ are equal can be answered in $O(n^2)$ time by traversing every arc $A \in \mathcal{M}$, and checking whether $R(A, \mathcal{M}_1) = R(A_2, \mathcal{M}_2)$ for any fixed arc $A_2 \in \mathcal{M}_2$.

In this section we describe how to obtain a canonical encoding of a PCA model, so that the equality of models can be tested in $O(n)$ time by comparing the encodings. The goal is to find a linear-time function C, mapping models to arc encodings, so that $\mathcal{M}_1 =_M \mathcal{M}_2$ if and only if $C(\mathcal{M}_1) = C(\mathcal{M}_2)$. The idea is to take the "minimum" arc encoding of a PCA model as its canonical representation.

Let $<$ be the total ordering on the alphabet {'a', 'b'} where 'a' $<$ 'b'. Define the relation $\preceq$ between the arcs of a model so that $A \preceq A'$ if and only if either $E(A) <_{\text{lex}} E(A')$ or $E(A) = E(A')$ and $T(A) \leq T(A')$. Those arcs that are minimum under the $\preceq$ relation are called *canonical*. That is, A is a canonical arc if and only if $A \preceq A'$ for every arc $A' \in \mathcal{M}$. For a canonical arc A, we say that $R(A)$ is a *canonical* arc encoding and that $\langle E(A), T(A) \rangle$ is a *canonical* extreme encoding. By definition, the canonical extreme encoding of a PCA model is unique and, since a canonical encoding uniquely determines an arc encoding, the canonical arc encoding of a PCA model is also unique. The following lemma shows how can a canonical extreme encoding be obtained from any extreme encoding.

Proposition 7.2.1. *Let $\mathcal{M}$ be a PCA model with arcs $A_1, \ldots, A_n$ where $s(A_1), \ldots, s(A_n)$ appear in this order, and let $A_i \in \mathcal{A}$. If p is the position of the j-th symbol* 'a' *in $E(A_i)$, then $E(A_{i+j}) = E(A_i) \ll p$ for every $1 \leq j \leq n$.*

Proof. By definition, $E(A_i)$ is the string that is obtained by traversing $C(\mathcal{M})$ from $s(A_i)$, and writing the symbols 'a' and 'b' each time a beginning and an ending point are respectively reached. If $C(\mathcal{M})$ is traversed from $s(A_{i+j})$ and the same procedure is applied, then the string obtained is $E(A_i) \ll p$ which, by definition, is equal to $E(A_{i+j})$. □

Lemma 7.2.2. *Let $\mathcal{M}$ be a PCA model with arcs $A_1, \ldots, A_n$ where $s(A_1), \ldots, s(A_n)$ appear in this order, and let $A_i \in \mathcal{A}(\mathcal{M})$. Then the following are equivalent:*

(i) A_i *is a canonical arc.*

(ii) $E(A_i) \leq_{lex} E(A_j)$ *for every* $A_j \in \mathcal{A}(\mathcal{M})$.

(iii) For $A_j \in \mathcal{A}(\mathcal{M})$, *the position of the* $(i-j)$*-th symbol* 'a' *in* $E(A_j)$ *is canonical.*

Proof. $(i) \implies (ii)$. It follows from the definition.

$(ii) \implies (iii)$. Let q be any canonical position of $E(A_j)$, and define p as the position of the $(i-j)$-th symbol 'a' in $E(A_j)$. By definition, $E(A_j) \ll q \leq_{\text{lex}} E(A_j) \ll p$. Since $E(A_j) \ll p$ begins with a symbol 'a', then the q-th position of $E(A_j)$ corresponds to a symbol 'a'. So, the q-th position represents the beginning point of some arc, say A_k. By Proposition 7.2.1, $E(A_i) = E(A_j) \ll p$ and $E(A_k) = E(A_j) \ll q$, so either p is a canonical position of $E(A_j)$ (when $E(A_i) = E(A_k)$) or $E(A_k) <_{\text{lex}} E(A_i)$.

$(iii) \implies (i)$. Call $E = E(A_i)$, and let A_{i+j} be any canonical arc of $\mathcal{M}$ and p be the position of the $(j+1)$-th symbol 'a' in E. By Proposition 7.2.1, $E(A_{i+j}) = E \ll p$ and, by definition of canonical arc, $E \ll p \leq_{\text{lex}} E$. Since position 0 is a canonical position of E, it follows that $E \leq_{\text{lex}} E \ll p$, thus $E = E \ll p$. Consequently, $E = E \ll kp$ for every $k \geq 0$. So, for every position x, the number of symbols 'a' in $[E(x), E(p+x))$ is equal to the number of symbols 'b' in $[E(x), E(p+x))$, since there are exactly n symbols 'a' and n symbols 'b' in E. Furthermore, $p = 2j$ because there are exactly j symbols 'a' in $[E(0), E(p))$. So, the p-th symbol that appears after $t(A_i)$ in E is exactly the j-th symbol 'b' that appears after $T(A_i)$ in E. Since this symbol represents the ending point of A_j in E, then $T(A_j) = T(A_i)$, thus A_i is canonical. □

From now on, we denote by $C(\mathcal{M})$ the canonical extreme encoding of $\mathcal{M}$, by $E^*(\mathcal{M})$ the extreme sequence corresponding to $C(\mathcal{M})$, and by $T^*(\mathcal{M})$ the mark point corresponding to $C(\mathcal{M})$. That is, $E^*(\mathcal{M}) = E(A, \mathcal{M})$ and $T^*(\mathcal{M}) = T(A, \mathcal{M})$ for some canonical arc $A \in \mathcal{M}$. The canonization procedure for extreme encodings is presented in Algorithm 7.3. The input of the algorithm is any extreme encoding $\langle E, T \rangle$. The first step computes a

canonical position p of E that corresponds to the position of the i-th symbol 'a', for some $1 \leq i \leq n$. The value i is computed at the second step of the algorithm. By Lemma 7.2.2, A_i is a canonical arc of $\mathcal{M}$ and, by Proposition 7.2.1, $E(A_i) = E \ll p$. As already argued, the ending points of the arcs appear in the same order of as the beginning points in a PCA model. Thus, the i-th symbol 'b' after the position T is the symbol that represents $t(A_i)$. Its position q is found in the third step, so $T(A_i) = q - p$ by definition. Consequently, the output of Algorithm 7.3 is the canonical extreme encoding $C(\mathcal{M})$.

Algorithm 7.3 Canonization of an extreme encoding.

Input: An extreme encoding $\langle E, T \rangle$ of a PCA model $\mathcal{M}$, for some $A \in \mathcal{A}(\mathcal{M})$.
Output: The canonical extreme encoding of $\mathcal{M}$.

1. Compute a canonical position p of E with respect to the order 'a' < 'b'.
2. Count the number i of symbols 'a' in $[E(0), E(p))$.
3. Find the position q of the i-th symbol 'b' that appears in E after the position T.
4. Output $\langle E \ll p,\ q - p \rangle$

With respect to the time complexity, the first step takes $O(n)$ evaluations of $<$ [Shi81], and, since the alphabet has only two symbols, each evaluation of $<$ takes $O(1)$ time. The second and third steps consist of a single traversal of E, and thus require $O(n)$ time[2]. Therefore, $C(\mathcal{M})$ is computed in $O(n)$ time and, by Remark 7.1.1, the canonical arc encoding is also computed in $O(n)$ time.

7.3 Canonical models of PCA graphs

In the previous section we developed an algorithm to compute a canonical encoding of a PCA model. In this section we take one step further, to compute a canonical model of a PCA graph. The goal is to find a linear-time function M, mapping PCA graphs to PCA models, so that two PCA graphs G_1 and G_2 are isomorphic if and only if $M(G_1) =_M M(G_2)$. The ultimate goal is to test whether G_1 is isomorphic to G_2 by checking whether $C(M(G_1)) = C(M(G_2))$. As before, the idea to find the canonical model is to compute the "minimum" PCA model of a graph.

Define the relation $\preceq$ between PCA models where $\mathcal{M} \preceq \mathcal{M}'$ if and only if either $E^*(\mathcal{M}) <_{\text{lex}} E^*(\mathcal{M}')$ or $E^*(\mathcal{M}) = E^*(\mathcal{M}')$ and $T^*(\mathcal{M}) \leq T^*(\mathcal{M}')$. Relation $\preceq$ is

[2] recall the $\Theta(\log n)$ word size assumption.

the generalization to PCA models of the relation $\preceq$ between arcs. Note that two models $\mathcal{M}$ and $\mathcal{M}'$ are equal if and only if $\mathcal{M} \preceq \mathcal{M}'$ and $\mathcal{M}' \preceq \mathcal{M}$. Since $\preceq$ is also transitive, $\preceq$ is a total order between PCA models. The following proposition shows how can two PCA models be compared efficiently.

Proposition 7.3.1. *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two PCA models, and $<$ be the total order on the alphabet {'a','m','b'} where 'a' $<$ 'm' $<$ 'b'. Define the string S_i that is obtained from $E^*(\mathcal{M}_i)$ by replacing the symbol 'b' that appears at position $T^*(\mathcal{M}_i)$ with the symbol 'm', for $i \in \{1, 2\}$. Then, $\mathcal{M}_1 \preceq \mathcal{M}_2$ if and only if $S_1 \leq_{lex} S_2$.*

By the above proposition, we can sort a family of PCA models $\{\mathcal{M}_1, \ldots, \mathcal{M}_k\}$ in $O(\sum_{i=1}^{k} |E^*(\mathcal{M}_i)|)$ time as follows. First, compute the string S_i from $E^*(\mathcal{M}_i)$ by replacing the symbol at position $T^*(\mathcal{M}_i)$ with the symbol 'm', for every $1 \leq i \leq k$. Second, radix sort the set $\{S_1, \ldots, S_k\}$ to obtain the permutation $S'_1 \leq_{\text{lex}} \ldots \leq_{\text{lex}} S'_k$. Finally, translate the order between the strings to the family of models. We record this discussion as a remark.

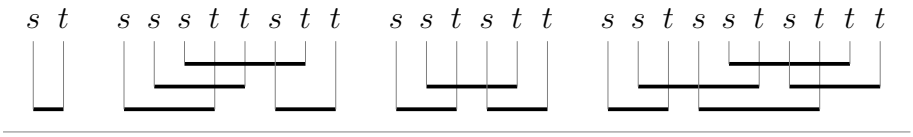
Remark 7.3.2. A family $\{\mathcal{M}_1, \ldots, \mathcal{M}_k\}$ of PCA models can be sorted with respect to the total order $\preceq$ in $O(\sum_{i=1}^{k} |E^*(\mathcal{M}_i)|)$ time.

We are now ready to define the canonical PCA model of a PCA graph. The canonical model is built differently according to whether the graph is a PIG graph or not. When the graph is not PIG, then we have to compute all the co-components of the graph. All the tools that we developed to work with co-components of PCA graph are described in terms of round enumerations, so we use the round enumeration terminology once again to describe our algorithms. We also use this terminology for PIG graphs, so as to highlight the similarities between the PIG and non-PIG cases. We note, however, that the transformations between PCA models and round enumerations are not strictly required (see [LSS08]).

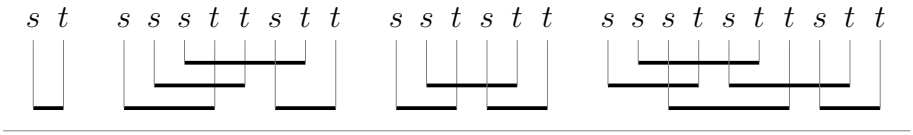
PIG graphs

Suppose that G is a PIG graph. By Theorem 3.3.2, each component H of G has at most two PIG orders, one the reverse of the other. Each of these PIG orders of H corresponds to one of the two PIG models of H, as in the proof of Proposition 2.1 in [DHH96]. Let $G_1, \ldots, G_k$ be the components of G. For $i = 1, \ldots, k$, let ϕ_i be the PIG order that corresponds to the PIG model $\mathcal{I}_i$ of G_i such that $\mathcal{I}_i \preceq \mathcal{I}_i^{-1}$ (see Figure 7.2 (b)). Furthermore, suppose that $\mathcal{I}_i \preceq \mathcal{I}_{i+1}$ for every $1 \leq i < k$ (see Figure 7.2 (c)). It is not hard to see that $\phi(G) = \phi_1, \ldots, \phi_k$ is a PIG order of G. Moreover, $\phi(G)$ is a canonical PIG order of G in the sense that $\phi(G) = \phi(H)$ for every graph H isomorphic to G. Indeed, $\mathcal{I}_i$ is uniquely determined for G_i because $\preceq$ is a total order between models; the

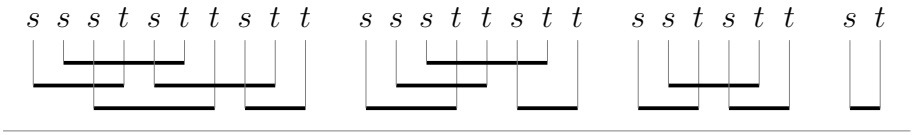
ordering $\mathcal{I}_1, \ldots, \mathcal{I}_k$ is unique because $\preceq$ is a total order; and ϕ_i is uniquely determined from $\mathcal{I}_i$. Therefore, $\phi(G) = \phi(H)$ for every pair of isomorphic PIG graphs G and H. The *canonical model* $M(G)$ of G is the PIG model corresponding to the PIG order $\phi(G)$.



(a) A PIG model $\mathcal{I}_a$.



(b) A model $\mathcal{I}_b$ with components $\mathcal{I}_1, \ldots, \mathcal{I}_k$ such that $\mathcal{I}_i \preceq \mathcal{I}_i^{-1}$.



(c) A model $\mathcal{I}_c$ with components $\mathcal{I}_1, \ldots, \mathcal{I}_k$ such that $\mathcal{I}_i \preceq \mathcal{I}_i^{-1}$ and $\mathcal{I}_i \preceq \mathcal{I}_{i+1}$.

Figure 7.2: Canonization of a PIG model. First, the components of the input model $\mathcal{I}_a$ are canonized so as to obtain an equivalent model $\mathcal{I}_b$. Then, the components of $\mathcal{I}_b$ are sorted so as to obtain an equivalent canonical model $\mathcal{I}_c$.

Algorithm 7.4 can be used to find the canonical model of a PIG graph G. The algorithm has three main steps. First, a canonical PIG order of each component of G is computed. Then, these PIG orders are sorted according to the operation $\preceq$ of their corresponding PCA models. Finally, these sorted PIG orders are concatenated into $\phi(G)$, and the corresponding model $M(G)$ is computed.

Consider the time complexity of Algorithm 7.4 when the input is a PIG order as in Step 1. A component of a PIG order ϕ is just a range $[v, w]$ of ϕ such that v is a leftmost end vertex (*i.e.*, $f_l(v) = v$), w is a rightmost end vertex (*i.e.*, $f_r(w) = w$), and (v, w) contains no end vertices. Thus, an $O(n)$ time traversal of ϕ is enough to compute all the components at Step 2. The comparison at Step 4 can be done as in Proposition 7.3.1, with a global cost of $O(n)$ time. Step 5 takes $O(n)$ time if a radix sort is used as in Remark 7.3.2. Therefore, $M(G)$ is computed in $O(n)$ time for PIG graphs.

Algorithm 7.4 Canonical model of a PIG graph.
Input: A PIG graph G.
Output: An extreme encoding of $M(G)$.

1. Let ϕ be a PIG order of G.
2. Compute the components $\phi_1, \ldots, \phi_k$ of ϕ.
3. Invoke Algorithm 7.2 to find the model $\mathcal{I}_i$ corresponding to ϕ_i.
4. Define $\mathcal{I}(i) := \min_{\preceq}\{\mathcal{I}_i, \mathcal{I}_i^{-1}\}$ for $1 \leq i \leq k$.
5. Sort the family $\{\mathcal{I}(1), \ldots, \mathcal{I}(k)\}$ so that $\mathcal{I}(i) \preceq \mathcal{I}(i+1)$, for every $1 \leq i < k$.
6. Invoke Algorithm 7.1 to find the round enumeration $\phi(i)$ corresponding to $\mathcal{I}(i)$.
7. Compute $\psi := \phi(1), \ldots, \phi(k)$.
8. Output the result of Algorithm 7.2 with input Ψ.

Non-interval PCA graphs

The procedure for the canonization of non-interval PCA graphs is similar to the procedure for the canonization of PIG graphs. The main difference is that the graph is split into co-components instead of being split into components.

Suppose that G is a non-interval universal-free graph, thus G is connected. By Proposition 6.3.1, either G is co-connected or each co-component is a co-bipartite graph. Hence, each co-component of G is either connected or it is the union of two complete sets. Whichever the case, each co-component H of G has at most two round enumerations, one the reverse of the other, by Theorem 3.3.3. Each of these round enumerations of H corresponds to one of the two PCA models of H, as in Proposition 2.6 of [DHH96]. Let $G_1, \ldots, G_k$ be the co-components of G. For $i = 1, \ldots, k$, let ϕ_i be the round enumeration that corresponds to the PCA model $\mathcal{M}_i$ of G_i such that $\mathcal{M}_i \preceq \mathcal{M}_i^{-1}$. Furthermore, suppose that $\mathcal{M}_i \preceq \mathcal{M}_{i+1}$ for every $1 \leq i < k$. Since ϕ_i $(1 \leq i \leq k)$ is co-bipartite, and G contains no universal vertices, then ϕ_i has two co-bipartite ranges, namely $\mathcal{X}_i$ and $\overline{\mathcal{X}_i}$. Without loss of generality, suppose that $\mathcal{X}_i$ contains a vertex that corresponds to a canonical arc of $\mathcal{M}_i$. Define the enumeration

$$\phi(G) = \mathcal{X}_1, \ldots, \mathcal{X}_k, \overline{\mathcal{X}_1}, \ldots, \overline{\mathcal{X}_k}.$$

By Proposition 6.3.17, the enumeration $\phi(G)$ is a round enumeration of G since $\phi(G)$ is equal to the enumeration ψ_k that is computed with the equations below.

- $\psi_1 = \phi_1$ and $\mathcal{Y}_1 = \mathcal{X}_1$.
- $\psi_i = \langle \mathcal{Y}_{i-1}, \mathcal{X}_i, \overline{\mathcal{Y}_{i-1}}, \overline{\mathcal{X}_i} \rangle$ and $\mathcal{Y}_i = \mathcal{Y}_{i-1}, \mathcal{X}_i$, for $1 < i \leq k$.

Furthermore, $\phi(G)$ is a canonical round enumeration of G, in the sense that $\phi(G) = \phi(H)$ for every graph H isomorphic to G. Indeed, the ordering $\phi_1, \ldots, \phi_k$ is unique as before; each ϕ_i has exactly two co-bipartite ranges because it is co-connected, and if both co-bipartite ranges contain a vertex corresponding to a canonical arc, then these ranges are equal because canonical arcs are indistinguishable. Therefore, $\phi(G) = \phi(H)$ for every pair of isomorphic universal-free non-interval PCA graphs G and H. The *canonical model* $M(G)$ of G is the PCA model corresponding to the round enumeration $\phi(G)$.

Algorithm 7.5 can be used to find a canonical PCA model $M(G)$ for every non-interval PCA graph G. The first step is to remove the universal vertices from a round enumeration ϕ of G. Then, we compute a round enumeration for each co-component of G, and we sort them according to the operation $\preceq$ of their corresponding PCA models. After these round enumerations are sorted, we compute $\phi(G)$ as in the above equation and we insert back all the universal vertices. Finally, we transform the obtained enumeration into an extreme encoding.

Consider the time complexity of Algorithm 7.5 when the input is a round enumeration as in Step 1. For Step 2, we can compute the universal vertices as in Lemma 2.4.9, or we can take advantage of the invocation of Algorithm 6.2 at Step 3 so as to remove the singleton co-components. Arguments similar to those in the previous paragraph are enough to conclude that Steps 4–6 take $O(n)$ time. Finally, note that the vertex $v(i)$ of Step 8 is just the first vertex of the enumeration $\phi(i)$ computed by Algorithm 7.2, because the extreme encoding $E^*(\mathcal{M}(i))$ is some encoding $E(A, \mathcal{M}(i))$ for a canonical arc A. Since all the other algorithms that are invoked by Algorithm 7.5 run in linear time, we obtain that Algorithm 7.5 finds a canonical model in $O(n)$ time.

7.4 Putting it all together

By construction, two PCA graphs G and H are isomorphic if and only if their canonical models $M(G)$ and $M(H)$ are equal. Moreover, $M(G)$ and $M(H)$ are equal if and only if their canonical extreme encodings $C(M(G))$ and $C(M(H))$ are equal. Thus, we can conclude that the isomorphism problem can be reduced to testing the equality of two strings.

Theorem 7.4.1. *Let G and H be two PCA graphs. Then the following are equivalent:*

1. *G and H are isomorphic,*
2. *$\mathcal{M}(G) =_M \mathcal{M}(H)$,*

Algorithm 7.5 Canonical model of a non-interval PCA graph.
Input: a non-interval PCA graph G.
Output: An extreme encoding of $M(G)$.

1. Let ϕ be a round enumeration of G.

2. Remove the universal vertices from ϕ and insert them into a block B.

3. Invoke Algorithm 6.2 to find the co-components $\phi_1, \ldots, \phi_k$ of ϕ.

4. Invoke Algorithm 7.2 to find the model $\mathcal{M}_i$ corresponding to ϕ_i.

5. Define $\mathcal{M}(i) := \min_{\preceq}\{M_i, M_i^{-1}\}$ for $1 \leq i \leq k$.

6. Sort the family $\{\mathcal{M}(1), \ldots, \mathcal{M}(k)\}$ so that $\mathcal{M}(i) \preceq \mathcal{M}(i+1)$, for every $1 \leq i < k$.

7. Invoke Algorithm 7.1 to find the round enumeration $\phi(i)$ corresponding to $\mathcal{M}(i)$.

8. Let $v(i)$ be the vertex of $\phi(i)$ corresponding to a canonical arc of $\mathcal{M}(i)$.

9. Invoke Algorithm 6.2 with input $\phi(i)$ and $v(i)$ to find the co-bipartite ranges $\mathcal{X}(i)$ and $\overline{\mathcal{X}(i)}$, for $1 \leq i \leq k$.

10. Let $\psi := \phi(1)$ and $\mathcal{Y} := X(1)$.

11. For $i := 1, \ldots, k$:

12. Compute $\psi := \langle \mathcal{Y}, \mathcal{X}(i), \overline{\mathcal{Y}}, \overline{\mathcal{X}(i)} \rangle$ as in Proposition 6.3.17 and set $\mathcal{Y} := \mathcal{Y}, \mathcal{X}(i)$.

13. Compute $\Psi := \langle \mathcal{Y}, B, \overline{\mathcal{Y}}, \emptyset \rangle$ as in Proposition 6.3.17.

14. Replace Ψ in B with $|B|$ universal vertices.

15. Output the result of Algorithm 7.2 with input Ψ.

3. $C(\mathcal{M}(G)) = C(\mathcal{M}(G))$.

Let $\mathcal{M}$ be a PCA model of a graph G. The canonical model $M(G)$ can be obtained from $\mathcal{M}$ in $O(n)$ time with a four step procedure. The first step is to transform $\mathcal{M}$ into a PIG model when G is a PIG graph. For this, we can run Algorithm 5.6 on $\mathcal{M}$; a PIG model of G is obtained if G is a PIG graph, while G is not a PIG graph otherwise. The second step is to run Algorithm 7.1 on $\mathcal{M}$ so as to obtain a round enumeration ϕ of G. Note that ϕ is a PIG order whenever $\mathcal{M}$ is a PIG model. The third step is to determine whether ϕ is a PIG order, by traversing ϕ so as to find out whether ϕ has an end vertex. The last step is to obtain the model $M(G)$ by running either Algorithm 7.4 or 7.5 with input ϕ, according to whether ϕ is a PIG order or not. Clearly, this procedure takes $O(n)$ time, so we obtain the main theorems of this chapter.

Theorem 7.4.2. *The canonical model of a PCA graph can be obtained in $O(n + m)$ time. Furthermore, if a PCA model of the input graph is given then the canonical model of the graph can be found in $O(n)$ time.*

Corollary 7.4.3. *The isomorphism problem for PCA graph can be solved in $O(n + m)$ time. Furthermore, if PCA models of the input graphs are given then the isomorphism problem can be solved in $O(n)$ time.*

8 The clique operator on circular-arc graphs

In this chapter we consider three problems on clique graphs of circular-arc graphs. The first problem is to characterize the class of clique graphs of HCA, NHCA, and PHCA graphs. The second problem investigates how to compute all the cliques of an HCA graph. The third problem is to characterize the graph to which a general circular-arc graph K-converges, if the graph is K-convergent. For the second problem we develop a linear-time algorithm, while for the other two problems we propose new characterizations that lead to linear-time algorithms for the associated recognition problems. To begin this chapter, we introduce the terminology of clique graphs and some notions that we require.

The *clique graph* $K(G)$ of a graph G is the intersection graph of the cliques of G. That is, $K(G)$ has one vertex for each clique of G and two vertices of $K(G)$ are adjacent when their corresponding cliques in G have nonempty intersection. A graph is a *clique graph* if it is isomorphic to $K(G)$, for some graph G [Ham68, RS71]. If $\mathcal{C}$ is a class of graphs, we denote by $K(\mathcal{C})$ the class of graphs which are clique graphs of the members of $\mathcal{C}$. One of the common questions on clique graphs is to characterize and recognize clique graphs of classes of graphs. In fact, clique graphs of several classes have been characterized and several algorithms are known for testing if a graph is a clique graph of some class (see [Szw03]). For many of these classes there are also polynomial-time recognition algorithms. However, recently the complexity of the recognition of clique graphs of arbitrary graphs was proved to be NP-Hard [AFdFG09]. A class of graphs $\mathcal{C}$ is *K-fixed* when $K(\mathcal{C}) = \mathcal{C}$, and it is *$K$-closed* when $K(\mathcal{C}) \subset \mathcal{C}$. In [Szw03] it is remarked that a large number of the classes whose clique graphs have been characterized so far are K-fixed or K-closed. This, for instance, is the case for interval graphs (cf. below).

The *iterated clique graph* is defined by $K^0(G) = G$ and $K^{i+1}(G) = K(K^i(G))$. The analysis of the *K-behavior* of a clique graph is one of the main topics about iterated clique graphs. A graph G is *K-null* if $K^i(G)$ is the trivial graph, for some $i \geq 0$. Say that G is *K-periodic* with period i if $K^i(G) = G$ for some $i > 0$. When the period is 1, the K-periodic graph is called *self-clique*. A graph is *K-convergent* when it is K-null or $K^i(G)$ is K-periodic for some $i \geq 0$. If G is not K-convergent, then $|V(K^i(G))|$ is unbounded when $i \longrightarrow \infty$; in this case G is *K-divergent*. To determine the K-behavior

of a graph G means to decide if G is K-null, K-convergent or K-divergent. Other questions related to iterated graphs include determining the speed of convergence, its period, or the speed of divergence (see [Szw03]). For the general case, the problem of determining the K-behavior of a graph is not known even to be computable. Nevertheless, polynomial-time algorithms to decide the K-behavior of a few classes are known. This is the case for cographs [LdMM+04], P_4-tidy graphs [dMML06], and complete multipartite graphs [NL78]. Clique-Helly graphs K-converge to graphs with period either 1 or 2 [Esc73], interval graphs are K-null, and octahedra of dimension at least three K-diverge (the octahedra of dimension n is the graph $\overline{nK_2}$). In this chapter we show how to combine the results by [FANLP04], [GH88], and [LNLP09] in order to obtain a linear-time algorithm for deciding the K-behavior of a circular-arc graph.

The *dismantling* of a graph G is the graph obtained by iteratively removing one dominated vertex of G, until no more dominated vertices remain. It is not hard to see that the dismantling of a graph is unique, up to isomorphism. Those vertices that are not removed while computing a dismantling of G form a *dismantling set*. Note that G could have many dismantling sets, although all of them induce the dismantling of G. In [FANLP04] it is proved that the K-behavior is the same for a graph and its dismantling, *i.e.*, they are both K-null, or they are both K-convergent and not K-null, or they are both K-divergent. By definition, the dismantling of any graph can be computed in polynomial-time. We show that a procedure similar to the one in [GH88] can be used to compute the dismantling in $O(n)$ time when a circular-arc model is given.

The first problem that we consider is the characterization and recognition of clique graphs of HCA, NHCA, and PHCA graphs. As we already mentioned, the characterization and recognition of clique graphs of interval and PIG graphs were solved by Hedman [Hed84], who proved that $K(\text{IG}) = K(\text{PIG}) = \text{PIG}$. In relation to HCA graphs, Durán and Lin proved in [DL01] that $K(\text{HCA}) \subset \text{PCA} \cap \text{HCA}$, thus the HCA class is K-closed. The same paper also describes characterizations for the clique graphs of HCA graphs, but these characterizations did not lead to a polynomial-time recognition algorithm, and the time complexity of recognizing clique graphs of HCA graphs remained so far open. We characterize the clique graph of an HCA graph G, by proving that either $K(G)$ is a PHCA graph or $K(G) \setminus U$ is a co-bipartite PHCA graph with $|U| \geq 2$, where U is the set of universal vertices of $K(G)$. In addition, we prove that $K(\text{NHCA}) = K(\text{PHCA}) = \text{PHCA}$, obtaining a result that resembles the one by Hedman. These characterizations lead to linear-time recognition algorithms for the classes of clique graphs of HCA, NHCA, and PHCA graphs.

The second problem that we consider in this chapter is the computation of the clique graph of an HCA graph. The number of cliques of a general circular-arc graph could be exponential with respect to the number of vertices of the graph. So, this problem is not tractable[1] for general circular-arc graphs. However, HCA graphs have $O(n)$ cliques, thus

[1] at least when the vertices and the edges have to be explicitly enumerated.

the lower bound for the time complexity is $\Omega(n)$ in this case. Durán et al. [DLMS06] developed a two phase algorithm to build the clique graph of an HCA graph. The first phase finds all the cliques in $O(n^2)$ time, and the second phase builds the clique graph in $O(n)$ time. We will improve the first phase of this algorithm so as to find all the cliques in $O(n)$ time.

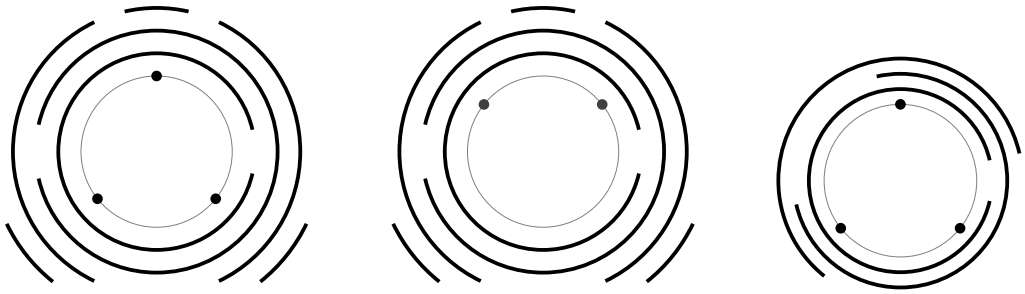
The last problem that we consider is the characterization of the graph to which a general circular-arc graph K-converges, when it does K-converge. This problem was partially solved by Bonomo [Bon06], who proved that an HCA graph G is K-periodic if and only if G is isomorphic to C_n^k with $n > 3k$. Moreover, K-periodic Helly circular-arc graphs are always self-clique. In a recent paper [LNLP09], Larrión et al. proved that the graph C_n^k is K-convergent if and only if $n > 3k$. By combining this result with Theorem 1 in [GH88] (see Theorem 4.1.1), we conclude that G is K-null if and only if its dismantling is K-null; G is K-convergent and not K-null if and only if its dismantling is C_n^k, with $n > 3k$; and G is K-divergent otherwise. We will prove that every K-convergent circular-arc graph always K-converges to its dismantling. Furthermore, we characterize the K-convergent circular-arc graphs which are not K-null, by showing that the class is exactly the class of NHCA graphs which are not interval graphs. Finally, we describe how the algorithm in [GH88] can be adapted to compute the dismantling of a circular-arc graph in $O(n)$ time, given a circular-arc model of it. This algorithm implies that the K-behavior of a circular-arc graph can be decided in linear time.

The characterizations of the clique graphs of HCA, NHCA, and PHCA are described in Section 8.1, the algorithm to compute all the cliques of an HCA graph appears in Section 8.2, and Section 8.3 contains the results on the K-behavior of circular-arc graphs. Section 8.2 is a joint work with Ross McConnell.

8.1 Characterization of clique graphs of HCA graphs

In this section we characterize the K(HCA), K(NHCA), and K(PHCA) classes, relating them to the PHCA class. In this sense, the characterizations are very similar to those given by Hedman [Hed84] for the characterization of the clique graphs of interval and PIG graphs. Recall that $K(\text{HCA}) \subset \text{PCA}$ [DL01]. For the characterizations we need to describe the PCA models of the clique graphs, both for HCA and for NHCA graphs.

Fix an HCA model $\mathcal{M}$. In this chapter we denote by $\mathcal{A}(p)$ the collection of arcs of $\mathcal{M}$ that contain the point $p \in C$. Clearly, $\mathcal{A}(p)$ is a complete set of $\mathcal{M}$, and it is a clique if and only if p is a clique point. For points $p \neq p'$ on the circle, say that p *dominates* p' if $\mathcal{A}(p') \subseteq \mathcal{A}(p)$. Say that p *properly dominates* p' when the inclusion is proper, while p is *equivalent* to p' when $\mathcal{A}(p) = \mathcal{A}(p')$. In $\mathcal{M}$, every non properly dominated point is a clique point and vice versa, therefore, there is a one-to-one correspondence between cliques of $\mathcal{M}$ and non-equivalent clique points (see Figure 8.1 (a)). An *intersection*



(a) Clique points (b) Non-clique intersection points (c) A clique model

Figure 8.1: Example of the construction of a clique model, given an HCA model $\mathcal{M}$. In (a), the clique points of $\mathcal{M}$ are shown, and they are all intersection points. In (b) there are two intersection points that are properly dominated and in (c) the clique model of $\mathcal{M}$ (w.r.t. the set of clique points) is shown.

segment (s, t) is a pair of consecutive extremes where s is a beginning point and t is an ending point. Points inside intersection segments are called *intersection points*. Every clique point of $\mathcal{M}$ must be an intersection point [DLMS06], but the converse is not necessarily true because there can be multiple intersection segments that are contained in exactly the same set of arcs (see Figure 8.1 (b)). However, when $\mathcal{M}$ is an NHCA model, then every intersection point is also a clique point because (s, t) is exactly the intersection between the arc whose beginning point is s and the arc whose ending point is t.

The *arc reduction* of a clique point p is the arc $(s, t(A_k))$ where s is the beginning point of the intersection segment that contains p, and $A_k \in \mathcal{A}(p)$ is the arc whose ending point is farthest from p when traversing $C(\mathcal{M})$. Observe that if $s = s(A_k)$, then the arc reduction of p is precisely A_k. In such cases, we say that p and its arc reduction A_k are *strong*. Non-strong clique points as well as non-strong arc reductions are referred to as *weak*. If $\mathcal{M}$ is an NPHCA model, then every clique point is strong, *i.e.*, all the arc reductions of $\mathcal{M}$ are arcs of $\mathcal{M}$. A *clique point representation* of $\mathcal{M}$ is a maximal set of non-equivalent clique points. Define the *clique model* of $\mathcal{M}$, with respect to a clique point representation Q, as the model formed by the arc reductions of Q (see Figure 8.1 (c)). Durán et al. [DLMS06] proved that any clique model of the intersection graph G of $\mathcal{M}$ is a PCA model of $K(G)$. Furthermore, when $\mathcal{M}$ is an NHCA graph, then all its clique models are PHCA because every arc reduction is included in some arc of $\mathcal{M}$. We sum up this discussion with two results for future reference.

Theorem 8.1.1 ([DLMS06])**.** *Let $\mathcal{M}$ be an HCA model of a graph G. Then, every clique model of $\mathcal{M}$ is a PCA model of $K(G)$. Furthermore, if $\mathcal{M}$ is an NHCA model, then every clique model of $\mathcal{M}$ is also NPHCA.*

Proposition 8.1.2. *The clique model of every NPHCA model $\mathcal{M}$ is the submodel of $\mathcal{M}$ induced by its strong arcs.*

Durán et al. [DLMS06] developed an $O(n^2)$ time algorithm for building the clique model of an HCA model $\mathcal{M}$. This algorithm is composed by two steps: first compute a clique point representation of $\mathcal{M}$ in $O(n^2)$ time, and then compute the arc reductions of all the clique points in $O(n)$ time. In the next section we show how to improve the first step so as to run in $O(n)$ time.

A graph G is *clique-Helly* if every family of pairwise intersecting cliques of G has nonempty intersection. Every clique-Helly graph that contains an induced 3-sun H must contain an induced $K_{1,3}$ [Dra89, Szw97]. In [DL01] it is not only proved that $K(\text{HCA}) \subset \text{PCA}$, but also it is proved that every graph in $K(\text{HCA})$ is clique-Helly. Therefore, graphs in $K(\text{HCA})$ contain no induced 3-sun, because $K_{1,3}$ is not a PCA graph. We remark this fact for future reference, and analyze how does the center of a 4-wheel look like in clique graphs of HCA graphs. The *center* of a 4-wheel is the vertex of degree four.

Lemma 8.1.3. *Graphs in $K(HCA)$ contain no 3-suns as induced subgraphs.*

Lemma 8.1.4. *Let $\mathcal{M}$ be an HCA model of a graph G. If $K(G)$ contains a 4-wheel as an induced subgraph, then*

(i) $\mathcal{M}$ has two arcs covering the circle,

(ii) $K(G)$ has at least two universal vertices, and

(iii) the center of every induced 4-wheel of $K(G)$ is universal.

Proof. Let v_1, v_2, v_3, v_4, u be vertices of $H = K(G)$ that induce a 4-wheel, where v_1, v_2, v_3 and v_4 induce a hole in that order and u is adjacent to all the vertices of the hole. Since $\mathcal{M}$ is HCA, it follows that v_i $(1 \leq i \leq 4)$ is represented by a clique point $p_i \in C(\mathcal{M})$, and u is represented by a clique point $q \in C(\mathcal{M})$. In every circular-arc model, p_1, p_2, p_3, and p_4 should be in that circular order, or in the reverse one. Without loss of generality, assume the former, and suppose that q lies between p_1 and p_2 (see Figure 8.2 (a)). Since u is adjacent to v_4 in H, then there exists some arc $U_1 \in \mathcal{A}$ crossing both q and p_4. But v_4 is not adjacent to v_2, so U_1 does not cross p_2. Then, U_1 crosses p_1 and, since v_1 is not adjacent to v_3, it follows that U_1 does not cross p_3. The same argument can be used to show that there is an arc $U_2 \in \mathcal{A}$ crossing q and p_3, but not p_1 and p_4 (see Figure 8.2 (b)). Since v_3 is adjacent to v_4 in H, then there is an arc $A_{34} \in \mathcal{A}$ crossing p_3 and p_4 and, since v_3 is not adjacent to v_1 and v_4 is not adjacent to v_2, it follows that A_{34} crosses neither p_1 nor p_2. Analogously, there is an arc $A_{12} \in \mathcal{A}$ crossing p_1 and p_2 that crosses neither p_3 nor p_4. As a consequence of these facts, U_1, U_2, A_{12}, and A_{34} are all different (see Figure 8.2 (c)). Now, since $\mathcal{M}$ is Helly and U_1, U_2, A_{34} cover the circle, then U_1, U_2 must share a common point inside A_{34}, *i.e.*, U_1, U_2 cover the circle. Also, A_{12} and A_{34}

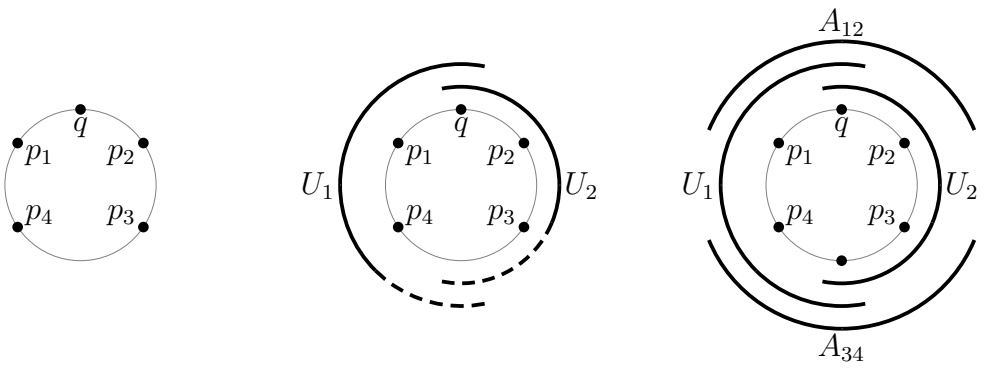


Figure 8.2: Proof of Lemma 8.1.4

do not intersect since otherwise A_{12}, A_{34}, U_1, and U_2 correspond to a complete set with no common point, which is impossible. Then there are at least two different cliques, one containing U_1, U_2, A_{12} and the other containing U_1, U_2, A_{34}, that are both universal, because every clique point is crossed by either U_1 or U_2. In other words, H has two universal vertices. Finally, point q is crossed by both U_1 and U_2, so u is universal in H. □

We are now ready to give both characterizations of $K(\text{PHCA})$ and $K(\text{HCA})$. As a corollary we will also obtain that $K(\text{NHCA}) = K(\text{PHCA})$ which resembles the fact that $K(\text{IG}) = K(\text{PIG})$.

Theorem 8.1.5. *Let H be a graph. Then $H = K(G)$ for some PHCA graph G if and only if H is a PHCA graph.*

Proof. Fix a PHCA graph G with a PHCA model $\mathcal{M}$. Graph $H = K(G)$ is PCA [DL01] and it does not contain 3-suns as induced subgraphs by Lemma 8.1.3. If two arcs of $\mathcal{M}$ cover the circle, then these two are universal arcs by Lemma 2.4.7, thus H is a clique which is clearly a PHCA graph. Otherwise, by Lemma 8.1.4, H has no 4-wheels as induced subgraphs. Hence, by Corollary 3.2.11, H is a PHCA graph.

For the converse, let H be a PHCA graph. If H is a PIG graph, then there exists a PIG graph G such that $K(G) = H$ [Hed84], and so the result follows. Suppose otherwise, and let $\mathcal{M}$ be an NPHCA model of H. Recall that such a model must always exist by Theorem 2.4.8. By Theorem 8.1.1, it suffices to find an NPHCA supermodel of $\mathcal{M}$ whose clique model is $\mathcal{M}$. Let $\mathcal{Q}$ be the set of arc reductions of $\mathcal{A}$ and $\mathcal{N} = \mathcal{A} \setminus \mathcal{Q}$. By Proposition 8.1.2, $\mathcal{Q}$ is a subset of arcs of $\mathcal{A}$. Note that, since H is not an interval graph, every arc of $\mathcal{A}$ contains at least one ending point of some other arc.

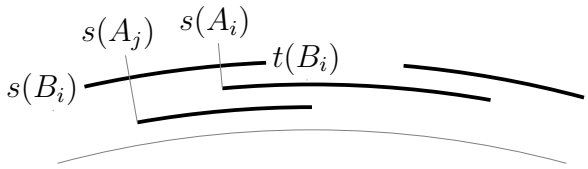


Figure 8.3: Example of the arc B_i associated to A_i in Theorem 8.1.5, where $A_j = NEXT_t(A_i)$.

Fix a small enough ϵ. For $A_i \in \mathcal{A}$, call $NEXT_t(A_i)$ to the arc whose ending point appears first when traversing C from $s(A_i)$. For each arc $A_i \in \mathcal{N}$ let B_i be the arc $(s(NEXT_t(A_i)) - \epsilon, s(A_i) + \epsilon)$ (see Figure 8.3). If two arcs B_i, B_j share their beginning points, then modify one of them so that none of them is included in the other. We claim that $\mathcal{M}' = (C, \mathcal{A} \cup \{B_i : A_i \in \mathcal{N}\})$ is an NPHCA model that has $\mathcal{M}$ as its clique model.

First we prove that $\mathcal{M}'$ is a proper model, *i.e.*, no arc of $\mathcal{M}'$ is contained in some other arc. Fix some B_i for $1 \leq i \leq n$. First observe that every arc A_j containing $s(B_i)$ does not cross $s(A_i)$, because the first arc crossing $s(A_i)$ is $NEXT_t(A_i)$. Thus $B_i \not\subset A_j$ for $1 \leq i, j \leq n$. Also $A_i \not\subset B_j$ for $1 \leq i, j \leq n$, because every beginning point that lies in B_j crosses $s(A_j)$, and ϵ is small enough. Finally, if B_i is contained in B_j, then $s(B_i)$ appears after $s(NEXT_t(A_j)) = s(B_j)+\epsilon$ because ϵ is small enough. Since $t(NEXT_t(A_j))$ appears after $t(B_j)$, we obtain that $NEXT_t(A_j)$ contains B_i, a contradiction. Thus, $\mathcal{M}'$ is a proper model.

We must now see that $\mathcal{M}'$ is Helly and normal, and for this it is enough to show that there are no two nor three arcs covering the circle. If B_i together with a set of arcs cover the circle, then $NEXT_t(A_i)$ with this set of arcs also cover the circle, because ϵ is small enough. Model $\mathcal{M}$ is Helly and normal, thus the smallest set of arcs covering the circle has size at least four and the same holds for $\mathcal{M}'$. Therefore, $\mathcal{M}'$ is an NPHCA model.

Finally, we have to prove that $\mathcal{M}'$ has $\mathcal{M}$ as its clique model. Every arc $A_i \in \mathcal{N}$ is strong, because $\mathcal{M}'$ is NPHCA and the first ending point that appears from $s(A_i)$ is $t(B_i)$. Also, the next ending point that appears from $s(B_i)$ is the beginning point of either $NEXT_t(A_i)$ or some B_j for $1 \leq j \leq n$, thus B_i is not strong. Last, the extreme that appears after $s(A_i)$ is not changed for $A_i \in \mathcal{Q}$, so it must be an ending point. Consequently, $\mathcal{M}$ is the submodel of $\mathcal{M}'$ induced by the strong arcs and the result follows from Proposition 8.1.2. □

Theorem 8.1.6. *Let H be a graph and U the set of universal vertices of H. Then, $H = K(G)$ for some HCA graph G if and only if:*

(i) H is a PHCA graph or

(ii) $H \setminus U$ is a co-bipartite PHCA graph and $|U| \geq 2$.

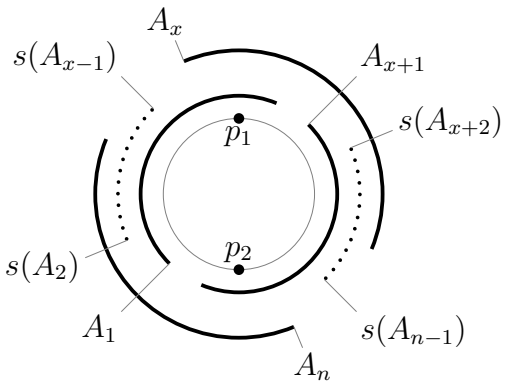


Figure 8.4: Positions of the beginning points in Theorem 8.1.6

Proof. Fix an HCA graph G with an HCA model $\mathcal{M}$. As in Theorem 8.1.5, H is a PCA graph and it contains no 3-suns as induced subgraphs. By condition (iii) of Theorem 8.1.4, the center of every induced 4-wheel of H is universal, thus $H \setminus U$ has no 4-wheels as induced subgraphs and consequently $H \setminus U$ is a PHCA graph, by Corollary 3.2.11. Then, if $U = \emptyset$, it follows that H is a PHCA graph. Otherwise, by conditions (i) and (ii) of Lemma 8.1.4, two arcs U_1, U_2 of $\mathcal{M}$ cover the circle, and H has at least two universal vertices. Then, every clique point of $\mathcal{M}$ is crossed by at least one of U_1, U_2, so the cliques of $\mathcal{M}$ can be partitioned into families $\mathcal{Q}_1$ and $\mathcal{Q}_2$ such that all the cliques in $\mathcal{Q}_i$ contain U_i, for $i \in \{1, 2\}$. In other words, the set of vertices of H corresponding to $\mathcal{Q}_i$ is a complete set for $i \in \{1, 2\}$, thus condition (ii) of this theorem holds.

For the converse, if H is a PHCA graph then the result follows from Theorem 8.1.5. Suppose, then, that $H \setminus U$ is a co-bipartite PHCA graph and that $|U| \geq 2$. Let $\langle V_1, V_2 \rangle$ be a co-bipartition of $H \setminus U$ and $\mathcal{M}_H$ be a PHCA model of $H \setminus U$. Each V_i is a complete set, therefore $\mathcal{M}_H$ has one point p_i that is crossed by all the arcs corresponding to the vertices in V_i, for $i \in \{1, 2\}$. Also, since $H \setminus U$ has no universal vertices, no arc crosses both points p_1 and p_2. Let $A_1, \ldots, A_n$ be the arcs of $\mathcal{M}$ such that $s(A_1), \ldots, s(A_n)$ appear in this order in a traversal of $C(\mathcal{M})$ where, w.l.o.g., p_2 lies in the segment $(s(A_n), s(A_1))$ and p_1 lies in the segment $(s(A_x), s(A_{x+1}))$, for some $1 \leq x \leq n$ (see Figure 8.4). Note that $A_1, \ldots, A_x$ are the arcs corresponding to V_1 and $A_{x+1}, \ldots, A_n$ are the arcs corresponding to V_2.

Fix some small enough ϵ, and define the arc $B_i = (s(A_i) - \epsilon, s(A_i) + \epsilon)$ for each arc $A_i \in \mathcal{A}(\mathcal{M})$. Let $\mathcal{M}_1$ be the model obtained from $\mathcal{M}_H$ by adding every B_i, for $A_i \in \mathcal{A}(\mathcal{M})$. Clearly, $\mathcal{M}_1$ is Helly and each of its intersection segments is of the form $(s(A_i), t(B_i))$. Also, every B_i contains only one intersection segment which is $(s(A_i), t(B_i))$. Then, it follows that every intersection point is a clique point, and the arc reduction of the points in $(s(A_i), t(B_i))$ is A_i. In other words, $\mathcal{M}_H$ is the clique model of $\mathcal{M}_1$. Therefore, by Theorem 8.1.1, the intersection graph G_1 of $\mathcal{M}_1$ is an HCA graph whose clique graph is

$H \setminus U$.

Let $U_1 = (s(A_1) - 3\epsilon, s(A_{x+1}) - 2\epsilon)$ and $U_2 = (s(A_{x+1}) - 3\epsilon, s(A_1) - 2\epsilon)$, and define $\mathcal{M}_2$ as the model obtained from $\mathcal{M}_1$ by adding U_1 and U_2. Suppose, to obtain a contradiction, that $\mathcal{M}_2$ is not Helly. By definition, U_i, A_j do not cover the circle for $i \in \{1, 2\}$ and $1 \leq j \leq n$. Then neither U_i and B_j can cover the circle. The only pair of arcs that cover the circle is U_1 and U_2, and, therefore, the minimal non-Helly complete sets have exactly three arcs by Theorem 3.0.3. If one of these arcs is B_i, then we can exchange it for A_i, for $1 \leq i \leq n$. Hence, we assume w.l.o.g. that one of the arcs is U_1 and the other two arcs are A_i, A_j, where A_i crosses $t(U_1)$ and A_j crosses $s(U_1)$. No arc in $A_1, \ldots, A_x$ crosses $s(A_1) - 3\epsilon = s(U_1)$, and no arc in $A_{x+1}, \ldots, A_n$ crosses $s(A_{x+1}) - 2\epsilon = t(U_1)$. Hence, $1 \leq i \leq x$ and $x + 1 \leq j \leq n$. Now, since ϵ is small enough and A_j crosses $s(A_1) - 3\epsilon$, we obtain that A_j intersects A_1. But then A_1, A_i, A_j cover the circle, which contradicts the fact that $\mathcal{M}_H$ is PHCA. Therefore, the intersection graph G_2 of $\mathcal{M}_2$ is HCA.

Let Q_1, Q_2 be two cliques of G_1, corresponding to vertices v_1, v_2 of $H \setminus U$. The clique points of $\mathcal{M}_1$ are still clique points in $\mathcal{M}_2$, because in $\mathcal{M}_2$ each B_i contains only the intersection segment $(s(A_i), t(B_i))$. So, Q_1 and Q_2 are also cliques of G_2. Even more, since U_i crosses only clique points corresponding to vertices of V_i ($i \in \{1, 2\}$), then Q_1 and Q_2 intersect if and only if they intersect in G_1. That is, Q_1 and Q_2 intersect in G_2 if and only if v_1 is adjacent to v_2 in $H \setminus U$, thus $H \setminus U$ is an induced subgraph of $K(G_2)$. Now, $\mathcal{M}_2$ has at most two more clique points than $\mathcal{M}_1$, because the inclusion of U_1 and U_2 has only two more intersection segments, $(s(U_1), t(U_2))$ and $(s(U_2), t(U_1))$. On the other hand, there is at least one more clique point in $\mathcal{M}_2$ because $\{U_1, U_2\}$ is contained in one universal clique, and G_1 has no universal cliques. Hence, G_2 is isomorphic to $H \setminus U$ plus one or two universal vertices. Adding at most $|U|$ pairwise disjoint arcs into $(s(U_1), t(U_2))$, we can obtain an HCA model of a graph G, where $K(G) = H$. □

Corollary 8.1.7. *Let H be a graph. Then, $H = K(G)$ for some NHCA graph G if and only if H is a PHCA graph.*

Proof. By Theorem 8.1.1, the clique model of any NHCA model of G is a PHCA model of $K(G)$. The converse follows from Theorem 8.1.5. □

Let $\mathcal{M}$ be a PCA model of a graph H. Recall that we can test whether $\mathcal{M}$ is equivalent to a PHCA model in $O(n)$ time, by running Algorithm 5.6 on $\mathcal{M}$. Thus, we can test whether $H \in K(\text{PHCA})$ in $O(n)$ time, when a PCA model of H is given. For the recognition of graphs in $K(\text{HCA})$ we need to take into account the universal arcs. By Lemma 2.4.9, the universal arcs of $\mathcal{M}$ are precisely those arcs that contain at least $n - 1$ extremes of the other arcs. Thus, the removal of the universal arcs from $\mathcal{M}$ can be done in $O(n)$ time, while counting how many universal arcs are there (see, *e.g.*, Algorithm 5.1). If $\mathcal{M}$ has exactly one universal arc, then $H \notin K(\text{HCA})$; otherwise, let $\mathcal{M}'$ be the obtained model. Graph H is in $K(\text{HCA})$ if and only if $H \setminus U$ is a PHCA graph, and either $U = \emptyset$ or $H \setminus U$

is co-bipartite. Testing if $H \setminus U$ is co-bipartite can be done in $O(n)$ time by running Algorithm 6.2 on (the corresponding round enumeration of) $\mathcal{M}'$, while testing if $H \setminus U$ is a PHCA graph is done in $O(n)$ time with an invocation of Algorithm 5.6 on $\mathcal{M}'$. To sum up, we can test whether $H \in K(\text{HCA})$ in $O(n)$ time when a PCA model of H is given. When H is given without a PCA model, we can run the algorithms in [DHH96, KN09] to test whether H is a PCA graph and obtain a PCA model as a by-product, when H is PCA.

Theorems 8.1.5 and 8.1.6 also yield procedures to compute an inverse graph with respect to the operator K. That is, given a graph H in $K(\text{HCA})$ or in $K(\text{PHCA})$, find a graph G such that $K(G) = H$. We omit the details here but it is not hard to see that both algorithms can be implemented so as to run in $O(n)$ time when the input is an HCA or PHCA model of H.

8.2 Cliques of an HCA graph

In [DLMS06], an $O(n^2)$ algorithm for constructing a clique model of an HCA model is described. The algorithm consists of two well defined procedures: 1) Find a clique point representation Q of the model, and 2) build the clique model with respect to Q. The first procedure is the bottleneck of the algorithm, and it takes $O(n^2)$ time, while the second procedure can be done in $O(n)$ time. In this section we develop a linear-time algorithm that reduces the time of the bottleneck step to $O(n)$. Given a set P of points in an arbitrary circular-arc model, our algorithm finds, in $O(n + |P|)$ time, a minimum set $P^* \subseteq P$ such that every point in $P \setminus P^*$ is dominated by some point in P^*. Letting P be one intersection point in each of the $O(n)$ intersection segments solves the bottleneck step.

Let $\mathcal{M}$ be a circular-arc model and $P = \{p_1, \ldots, p_k\}$ be a set of points of $C(\mathcal{M})$ where $p_1, \ldots, p_k$ appear in this order in a traversal of $C(\mathcal{M})$. Say that $P' \subseteq P$ is a *P-dominating* set if every point in $P \setminus P'$ is dominated by some point in P'. The purpose of our algorithm is to find a minimum P-dominating sequence. For this, the algorithm first traverses $C(\mathcal{M})$ so as to remove from P all those points p_i that are dominated by some point p_j, for $1 \leq i < j \leq k$. After these dominated points are removed, the algorithm traverses $C(\mathcal{M})$ once again so as to remove all those points p_j that are dominated by some point p_i, for $1 \leq i < j \leq k$. We define the semi-dominating sequences to formalize this idea.

The *ascendant semi-dominating sequence* of P is the subset

$$SD^+(P) = \{p_i \in P \mid \mathcal{A}(p_i) \not\subseteq \mathcal{A}(p_j) \text{ for all } p_j \in P \text{ such that } 1 \leq i < j \leq k\}.$$

In other words, $SD^+(P)$ is the set of points that survive the first traversal, *i.e.*, $SD^+(P)$ contains the points $p_i \in P$ that are not dominated by a point $p_j \in P$ such that $j > i$. Similarly, the *descendant semi-dominating sequence* of P is the subset

$$SD^-(P) = \{p_j \in P \mid \mathcal{A}(p_j) \not\subseteq \mathcal{A}(p_i) \text{ for all } p_i \in P \text{ such that } 1 \leq i < j \leq k\}.$$

The following is the key lemma that guaranties the correctness of our algorithm.

Lemma 8.2.1. *Let $\mathcal{M}$ be a circular-arc model and P be a set of points. Then, both $SD^-(SD^+(P))$ and $SD^+(SD^-(P))$ are minimum P-dominating sequences.*

Proof. We will only prove that $P^* = SD^+(SD^-(P))$ is a minimum P-dominating sequence since the proof for $SD^-(SD^+(P))$ can be obtained by taking the reverse model of $\mathcal{M}$. Let $p_1, \ldots, p_k$ be the points comprising P, where $p_1, \ldots, p_k$ appear in this order in a traversal of $C(\mathcal{M})$. We first prove that P^* is in fact a P-dominating sequence.

By definition, every point $p_j \in P \setminus SD^-(P)$ is dominated by some point $p_i \in P$ for some $1 \leq i < j$. If i is the minimum such that p_i dominates p_j, then no point $p \in \{p_1, \ldots, p_{i-1}\}$ can dominate p_i because otherwise p would dominate p_j, breaking the minimality of i. Therefore, every point in $P \setminus SD^-(P)$ is dominated by some point in $SD^-(P)$. We can apply the same arguments to $SD^-(P)$ and P^* to conclude that every point in $SD^-(P) \setminus P^*$ is dominated by some point in P^*. Since domination is a transitive relation, every point of P is also dominated by some point in P^*, *i.e.*, P^* is a P-dominating sequence.

We now show that P^* is minimum. Observe that it is enough to show that P^* is minimal because P^* is a P-dominating sequence and the domination relation is transitive. Suppose that $p_i \in P$ is dominated by $p_j \in P^*$. If $j < i$ then $p_i \notin SD^-(P)$, hence $p_i \notin P^*$. Otherwise, $p_i \notin P^*$ because it is dominated by $p_j \in SD^-(P)$ and $j > i$. Therefore, P^* is minimal and thus minimum. □

Algorithms to find SD^+ and SD^- are symmetric. We describe the one to find SD^+. The algorithm works by induction on the size of a $P_i = \{p_1, p_2, \ldots, p_i\}$. After the step i, the algorithm partitions the elements of $SD^+(P_i)$ into two sets D_i and Q_i such that:

- The points in D_i are crossed by some arc of $\mathcal{A}(\mathcal{M})$ that is entirely contained in the segment (p_k, p_i), hence these points cannot be dominated by a point in $\{p_{i+1}, p_{i+2}, \ldots, p_k\}$. These points are already known to be members of $SD^+(P)$.
- The set Q_i contains those points in $SD^+(P_i) \setminus D_i$, stored in clockwise order $q_1, q_2, \ldots, q_j$, where q_1 is the first point that appears from p_k in a traversal of $C(\mathcal{M})$. Their status as members of $SD^+(P)$ is uncertain; though they are in $SD^+(P_i)$, they might be dominated by points in $\{p_{i+1}, p_{i+2}, \ldots, p_k\}$.

After the step k, the algorithm returns $SD^+(P) = D_k \cup Q_k$. It remains to describe how to obtain D_{i+1} and Q_{i+1} from D_i and Q_i. The first step is to find those points in Q_i that must be added to D_i so as to obtain D_{i+1}. By definition, these points are crossed by some arc that is entirely contained in (p_k, p_{i+1}), but are crossed by no arc entirely contained in (p_k, p_i). So, the points in $D_{i+1} \setminus D_i$ are exactly those points of Q_i that are crossed by some arc whose beginning point lies in (p_k, p_{i+1}) and whose ending point lies in (p_i, p_{i+1}). Of all these arcs, let B be the one whose beginning point is closest to p_k. That is, B is the arc contained in (p_k, p_{i+1}) that crosses the most members of Q_i and, so, $D_{i+1} \setminus D_i$ is precisely the set of points in Q_i that are crossed by B.

The second step is to find those members of Q_i that belong to Q_{i+1}. By definition, $Q_{i+1} = SD^+(P_{i+1}) \setminus D_{i+1}$, and no point $q_a \in Q_i$ is dominated by a point $q_b \in Q_i$ such that $b > a$. Thus, Q_{i+1} is composed by those points of $Q_i \setminus D_{i+1}$ that are not dominated by p_{i+1}, plus p_{i+1}. So, a point $q \in Q_i \setminus D_{i+1}$ is in Q_{i+1} if and only if it is contained in some arc A that does not contain p_{i+1} (since otherwise it would be dominated by p_{i+1}) and that contains p_k (since otherwise it would already be identified as a member of D_{i+1}). Of all such arcs, let A be the one whose ending point is farthest from p_k. Since A is the arc that covers the most members of $Q_i \setminus D_{i+1}$, it follows that Q_{i+1} is just the points of $Q_i \setminus D_{i+1}$ that are contained in A.

The procedure to obtain $SD^+(P)$ from P is depicted as Algorithm 8.1. In the algorithm, Q is implemented as a stack of points $(q_1, \ldots, q_j)$, where $q_1, \ldots, q_j$ appear in this order in a traversal of $C(\mathcal{M})$, q_1 is the first point of Q that appears in a traversal of $C(\mathcal{M})$ from p_k, and q_j is the element at the top of Q. Each point $q \in Q$ is stored jointly with an arc $A(q)$. When $A(p_i)$ and p_i are inserted into Q at Step 5, $A(p_i)$ is the arc that crosses p_k, whose ending point lies in (p_i, p_{i+1}), and whose beginning point is closest to p_k (or empty if such an arc does not exist). Suppose that this condition also holds for all the other points of Q. That is, if q, q' are two consecutive points of Q, then $A(q)$ is the arc whose ending point lies in (q, q') and whose beginning point is closest to p_k. Observe that $A(q)$ crosses p_k, because otherwise q would not be in Q.

The loop from Steps 6 to 8 moves those points in Q that are covered by the arc B. To preserve the invariant conditions, at Step 8 we have to update the arc $A(q)$, corresponding to the point q at the top of the stack, so that its beginning point is the closest to p_k from those with ending point in (q, p_{i+1}). The loop from Steps 9 to 10 removes those points q that are dominated by p_{i+1}. Observe that if $A(q)$ crosses p_{i+1} then every arc that crosses q also crosses p_{i+1}. There is no need to update the arc corresponding to the point q' that remains at the top of the stack after q is popped, because if $s(A(q'))$ is not closer to p_k than $A(q)$ then q' is also dominated by p_{i+1}.

To sum up, Algorithm 8.1 correctly finds $SD^+(P)$. With respect to the time complexity, the main loop is executed $|P|$ times. Inside the main loop, Steps 3 and 4 traverse all the extremes of $\mathcal{M}$. Each extreme is traversed twice, the first time to find the arc B, and the second time to find the arc A_i. Hence, Steps 3 and 4 take overall $O(n)$ time. The

Algorithm 8.1 Ascending semi-dominating sequence

Input: An HCA model $\mathcal{M}$ and a set $P = \{p_1, \ldots, p_k\}$ of points of $C(\mathcal{M})$ such that $p_1, \ldots, p_k$ appear in this order in a traversal of $C(\mathcal{M})$.
Output: $SD^+(P)$.

1. Let $D := \emptyset$ be a set of points and $Q := \emptyset$ be a stack of pairs. Each item of Q is a point $q \in P$, together with an arc $A \in \mathcal{A}(\mathcal{M})$. Let $q(Q)$ be a function that returns the point at the top of Q, and $A(Q)$ be a function that returns the arc at the top of Q.
2. For $i := 1, \ldots, k-1$:
3. Find the arc B such that $t(B) \in (p_i, p_{i+1})$ and $s(B) \in (p_1, p_i)$ is closest to p_k. If such an arc does not exist, then set $B := \emptyset$.
4. Find the arc A_i such that $t(D) \in (p_i, p_{i+1})$, $p_k \in A_i$ and $s(A_i)$ is closest to p_k. If such an arc does not exist, then set $A_i := \emptyset$.
5. Push $\langle p_i, A_i \rangle$ into Q.
6. Do while $Q \neq \emptyset$ and $q(Q) \in B$:
7. Pop $\langle q, A \rangle := \langle q(Q), A(Q) \rangle$ from Q, and insert q into D.
8. If $Q \neq \emptyset$ and A is closer to p_k than $A(Q)$, then set $A(Q) := A$.
9. Do while $Q \neq \emptyset$ and $A(Q)$ crosses p_{i+1}:
10. Pop the pair at the top of Q.
11. Return $SD^+(P) := D \cup Q \cup \{p_k\}$.

operations at Steps 5, 6 and 9 take $O(1)$ time. Finally, the operations inside the inner loops are applied only to elements of Q that are popped from Q. Thus, executing these steps take a time proportional to the time that took all the insertions, which is $O(|P|)$. Hence, Algorithm 8.1 takes $O(n + |P|)$ time.

Once all the clique points are found, it is easy to find the maximum clique. If the arcs of $\mathcal{M}$ have weights, then the maximum weight (maximal) clique can also be found easily, even when some edge is negative. We record these facts as a simple corollary.

Theorem 8.2.2. *The maximum clique problem and the maximum weight clique problem can be solved in $O(n)$ time for HCA graph, when an HCA model is given as input.*

8.3 K-behavior of circular-arc graphs

In this section we develop an algorithm to find out to which graph does a circular-arc graph K-converge, when it does K-converge. Powers of cycles play an important role in this section. Recall that, by Theorem 1 in [GH88] (see also Theorem 4.1.1), the dismantling of a circular-arc graph is isomorphic either to the trivial graph or to the power of a cycle. This theorem can be combined with the following two results by Frias et al. and Larrión et al., to actually decide the K-behavior of a general circular-arc graph in polynomial time.

Theorem 8.3.1 ([FANLP04])**.** *A graph G has the same K-behavior as its dismantling H. That is, G is K-null if and only if H is K-null; G is K-convergent and not K-null if and only if H is K-convergent and not K-null; and G is K-divergent if and only if H is K-divergent.*

Theorem 8.3.2 ([LNLP09])**.** *Graph C_n^k is K-convergent if and only if it is a complete graph or $n > 3k$.*

By the theorems above, a circular-arc graph is K-null if its dismantling is the trivial graph; it K-converges to a graph which is not K-null if its dismantling is C_n^k with $n > 3k$; or it K-diverges otherwise. However, much more can be said about the graph to which G K-converges when it does, because this graph is self-clique and thus unique. We start with two useful propositions that are easy enough to prove.

Proposition 8.3.3. *For every graph G, there exists a dismantling set that contains no properly dominated vertex of G.*

Proposition 8.3.4. *Let W be a dismantling set of a graph G. If G' is an induced subgraph of G that contains every vertex of W and the vertices of $G \setminus G'$ are dominated by vertices of G', then W is a dismantling set of G'.*

Next, we show that every circular-arc graph that is neither K-null nor K-divergent must be NHCA. For this we need to show how can the dismantling of a circular-arc graph be computed when a circular-arc model $\mathcal{M}$ is given. The *CA-dismantling algorithm* can be divided into two steps. First remove every arc A_i that is contained in some arc A_j to obtain a PCA submodel $\mathcal{M}'$ of $\mathcal{M}$. Clearly, every removed arc corresponds to a dominated vertex. Second, iteratively remove every arc whose beginning point is immediately followed by some other beginning point. Since $\mathcal{M}'$ is PCA, all such arcs are dominated. At the end, every beginning point is followed by an ending point. If $\mathcal{M}'$ has some non-universal arc, then the model so obtained has no dominated arcs by Theorem 4.1.1. Otherwise, the dismantling of $\mathcal{M}$ is any trivial model.

Theorem 8.3.5. *A circular-arc graph G is K-convergent and not K-null if and only if G is a non-interval NHCA graph.*

Proof. Suppose first that G is K-convergent and not K-null, and let W be some of its dismantling sets. By Theorem 8.3.1 (see also [Pri92]) $|W| > 1$. Then, W is not a complete set and, by Theorem 4.1.1, $G[W]$ is isomorphic to C_n^k for some pair of values n, k. If $n \leq 3k$ then G is K-divergent by Theorem 8.3.2, which is also impossible, so $n > 3k$. Then G contains an induced cycle of length at least 4, that is, G is not an interval graph. Let $\mathcal{M}$ be a circular-arc model of G, and call $\mathcal{M}_W$ to the submodel of $\mathcal{M}$ induced by the arcs corresponding to vertices of W. We can assume, w.l.o.g., that $\mathcal{M}_W$ was computed by the CA-dismantling algorithm. Call $N_1, \ldots, N_s$ to the arcs of G that were removed by the CA-dismantling algorithm, where N_i was removed after N_{i+1} for $1 \leq i \leq s-1$. We prove by induction on i that $\mathcal{M}_i = \mathcal{M}_W \cup \{N_1 \ldots, N_i\}$ is an NHCA model.

For the base case $i = 0$, W is isomorphic to C_n^k with $n > 3k$ and so, by Theorem 4.1.1, $\mathcal{M}_H$ has no two nor three arcs that together cover the circle, *i.e.*, $\mathcal{M}_W$ is normal and HCA. For the inductive case, observe that by construction N_i is either properly contained in some arc $A \in \mathcal{M}_{i-1}$, or $\mathcal{M}_i$ is proper and $s(N_i)$ is followed by the beginning point of some arc $A \in \mathcal{M}_{i-1}$. In either case, if N_i together with a subset of arcs $\mathcal{A}$ in $\mathcal{M}_i$ cover the circle, then also A covers the circle with $\mathcal{A}$. Consequently, by the inductive hypothesis, $|\mathcal{A} \cup \{N_i\}| \geq 4$, *i.e.*, $\mathcal{M}_i$ is NHCA.

For the converse, assume that G is a non-interval NHCA graph. We employ induction to show that in every step of the dismantling process, the subgraph so far obtained contains an induced hole. Since G is NHCA and non-interval, G itself contains at least one hole. By the induction hypothesis the subgraph obtained after a certain number of removals of dominated vertices also contains a hole. Let $\mathcal{M}$ be the circular-arc model corresponding to this subgraph, and let $\mathcal{A} = \{A_1, \ldots, A_k\}$ be the set of arcs of some minimum hole, where A_i intersects A_{i-1} and A_{i+1} for $1 \leq i \leq k$. Examine the removal of the next dominated arc. If the removed arc is not one of $\mathcal{A}$, then the hole in $\mathcal{M}$ is preserved in the next step. Otherwise, some arc $A_i \in \mathcal{A}$ is either contained in an arc B_i or $\mathcal{M}$ is proper and $s(A_i)$ is followed by $s(B_i)$. In either case, B_i intersects A_{i-1} and A_{i+1}, and, since $\mathcal{A}$ induces a minimum hole, B_i is adjacent to either none or all of the arcs in $\mathcal{A} \setminus \{A_{i-1}, A_{i+1}\}$. In the former case, $(\mathcal{A} \setminus \{A_i\}) \cup \{B_i\}$ induces a hole and the invariant is preserved. In the latter case, $\mathcal{A}$ together with B_i induce a k-wheel, contradicting Theorem 3.2.9. Consequently, the dismantling of G contains a hole, meaning that it is neither a single vertex nor isomorphic to $C_{n,k}$ for $n \leq 3k$, *i.e.*, G is K-convergent and not K-null. □

Now we proceed to prove that every K-convergent circular-arc graph K-converges to its dismantling, which is the main theorem of this section.

Theorem 8.3.6. *If a circular-arc graph is K-convergent, then it K-converges to its dismantling.*

Proof. If G is K-null, then the dismantling of G is the trivial graph by Theorem 8.3.1 (see also [Pri92]), and thus G K-converges to its dismantling. So, assume that G is not K-null, which implies that G admits an NHCA model $\mathcal{M}$ by Theorem 8.3.5. We prove the theorem by induction on k, where k is the minimum number such that $K^k(G) = K^{k+1}(G)$. This induction is well defined because K-periodic circular-arc graphs are self-clique [Bon06].

In the base case, $K(G) = G$. By Theorem 8.1.1, the clique model $\mathcal{M}_Q$ of $\mathcal{M}$ is an NPHCA model of $K(G)$, thus $\mathcal{M} = \mathcal{M}_Q$ is a NPHCA model. Then, by Proposition 8.1.2, $\mathcal{M}_Q = \mathcal{M}$ has only strong arcs, implying that every beginning point of $\mathcal{M}$ is followed by an ending point. Hence, by Theorem 4.1.1, G contains no dominated arcs, *i.e.*, $G[W] = G = K(G)$ for any dismantling set W of G.

Now we proceed with the inductive case. Let W be a dismantling set of G. Observe that W must contain at least two vertices by Theorem 8.3.1, because G is not K-null. Let $\mathcal{M} = (C, \mathcal{A} \cup \mathcal{N})$ be an NHCA model of G where $\mathcal{A}$ is the set of arcs corresponding to W. Without loss of generality, we may assume that if an arc A contains another arc B, then there is some ending point between $s(A)$ and $s(B)$. We refer to this condition of $\mathcal{M}$ as the *s-ordering* condition. Also, by Proposition 8.3.3, we may assume that no vertex of W is properly dominated in G, hence if $A \in \mathcal{A}$ is dominated by $N \in \mathcal{N}$ then they must be twins. In the case that A and N are twins, assume that $s(N), s(A), t(N), t(A)$ appear in this order in a traversal of $C(\mathcal{M})$.

Claim 1: Every arc of $\mathcal{A}$ is strong. Let p be the first clique point of $\mathcal{M}$ that appears from $s(A)$, for $A \in \mathcal{A}$, and let B be the arc crossing p whose ending point is farthest from p. By Proposition 8.3.3, we have assumed that B does not properly dominate A in $\mathcal{M}$. Then, B crosses exactly the same clique points as A. If $A \neq B$, then B and A are twins. But we have also assumed that, in this case, $t(A)$ appears farther from p than $t(B)$, a contradiction. Then p is a strong clique point and A is its arc reduction. Therefore, every arc of $\mathcal{A}$ is strong and the claim is proved.

Claim 2: Every arc that belongs to $\mathcal{M}$ but not to its clique model $\mathcal{M}_Q$ is dominated by some strong arc in $\mathcal{M}$. If B belongs to $\mathcal{M}$ but not to $\mathcal{M}_Q$, it is because B is not an arc reduction. Then, either $s(B)$ is followed by some beginning point $s(B')$ in $\mathcal{M}$, or $(s(B), t)$ is an intersection segment which contains weak clique points, for some ending point t. In the former case, B does not contain B' by the s-ordering condition of $\mathcal{M}$, thus B is dominated by B'. In the latter case, there is some arc B' that crosses $s(B)$ and that reaches farther than $t(B)$, *i.e.*, B is contained B'. If B' is strong, then the proof of the claim is complete. Otherwise, by applying this reasoning iteratively and using the fact that domination is a transitive relation, we obtain that B is dominated by some strong arc of $\mathcal{M}$.

Claim 3: Every arc reduction of a weak clique point of $\mathcal{M}$ is dominated by a strong arc of $\mathcal{M}$. Suppose that B is the arc reduction of the weak clique point p, and let A be the

arc of $\mathcal{M}$ that crosses p and reaches farthest. By definition, $B = (s, t(A))$, where s is the first beginning point that appears from p in a counter-clockwise traversal of $C(\mathcal{M})$. Since p is weak, then A crosses s, and by the s-ordering condition of $\mathcal{M}$, there is some ending point between $s(A)$ and s. Hence, A crosses the clique point q that appears first from p in a counter-clockwise traversal of C. By definition, the arc reduction of q has $t(A)$ as its ending point, so the arc reduction of q dominates B. If q is a strong clique point, then the claim is proved. Otherwise, as in Claim 2, we can apply this reasoning and use the fact that domination is transitive to obtain that B is dominated by some strong arc of $\mathcal{M}$.

By Theorem 8.1.1, the clique model $\mathcal{M}_Q$ of $\mathcal{M}$ is an NPHCA model of $K(G)$. By definition, the arcs of $\mathcal{M}_Q$ are precisely the arc reductions of $\mathcal{M}$, thus $\mathcal{M}_Q$ contains all the strong arcs of $\mathcal{M}$. By Claim 3, we can compute the dismantling of $\mathcal{M}_Q$ by first removing all the arcs that are weak arc reductions of $\mathcal{M}$, and then computing the dismantling in the resulting model. In terms of vertices and graphs, the dismantling of $K(G)$ is isomorphic to the dismantling of $G[S]$, where $S \subseteq V(G)$ is the set of vertices that correspond to strong arcs of $\mathcal{M}$. Since $\mathcal{M}_Q$ contains all the strong arcs of $\mathcal{M}$, then, by Claim 1, $\mathcal{A}$ is a subset of arcs of $\mathcal{M}_Q$. That is, W is both a subset of vertices of $K(G)$ and a subset of S. Finally, by Claim 2, the vertices of $G \setminus G[S]$ are all dominated by some vertex of S. Thus, by Proposition 8.3.4, W is a dismantling set of $G[S]$ and therefore it is also a dismantling set of $K(G)$. Hence, by the induction hypothesis, $K(G)$ is K-convergent to $G[W]$ which concludes the proof. □

To end this section, we describe an implementation of the CA-dismantling algorithm which runs in $O(n)$ time. This implementation is rather similar to the one in [GH88] for computing the maximum independent set of a circular-arc graph. The main difference is that our algorithm eliminates all the dominated arcs, while the algorithm in [GH88] eliminates all the dominating arcs. The input of our algorithm is some circular-arc model $\mathcal{M}$ and the output is an induced submodel of it. Recall that the algorithm is divided into two steps. The first one is the removal of included arcs and the second step is the removal of arcs whose beginning points are followed by another beginning point. For the first step of the algorithm, traverse twice $C(\mathcal{M})$ from some beginning point $s(A)$, while maintaining the position of the farthest ending point t viewed so far. This farthest ending point is initialized to $t(A)$. When a beginning point $s(A_i)$ is reached, check if $t(A_i)$ reaches farthest than t. If so, then set $t := t(A_i)$; otherwise, A_i is contained in the arc whose ending point is t, so we can remove it. Since the circle is traversed twice, then every contained arc is eventually removed and we obtain a model $\mathcal{M}'$.

For the second step of the algorithm, first initialize a set $\mathcal{S}$ containing each non-singleton s-sequence. Each s-sequence can be represented as we usually represent ranges, by storing the first and last beginning points of the s-sequence. Now, choose some s-sequence $s_1, \ldots, s_k$ of $\mathcal{S}$. The beginning point s_1 is followed by the beginning point s_2,

thus we need to remove $A = (s_1, t(A))$ from $\mathcal{M}'$, and update (s_1, s_k) in $\mathcal{S}$ to (s_2, s_k). Let e_1 and e_2 be the previous and next extremes of $t(A)$ in $\mathcal{M}'$, respectively. If e_1 or e_2 is not a beginning point, then every non-singleton s-sequence is contained in $\mathcal{S}$. Otherwise, we may have to remove the s-sequences S_1 containing e_1 and S_2 containing e_2 from $\mathcal{S}$, and insert the non-singleton s-sequence $S_1 \cup S_2$. These operations can be done in $O(1)$ if two references are maintained in the first and last beginning points of each s-sequence. The first reference of each beginning point informs the position of its s-sequence in $\mathcal{S}$. The second reference links the first and last beginning points of the s-sequence. Hence, the whole algorithm can be implemented so as to run $O(n)$ time.

9 Open problems and further remarks

In this last chapter we describe many open problems related to the work we have done. We have briefly thought some of these problems, and tried to solve others with no success. For such problems, we describe some of our ideas which are vague, incomplete, and possibly wrong, with the hope that someone transforms our futile efforts into worthwhile contributions.

Characterization of circular-arc graphs and subclasses. In Chapter 3 we showed several characterizations that state which circular-arc graphs do not belong to a restricted class of circular-arc graphs. The only class treated in this thesis for which such characterization is still unknown is the class of NCA graphs. In [Mül97], Müller conjectured that a graph is both co-bipartite and NCA if and only if its complement contains no "insect" graph and no graph from a family $\mathcal{F}$. Hell and Huang [HH04] disproved this conjecture by showing a family of "bug" graphs that are co-bipartite but not NCA. They also claimed that there are others co-bipartite graphs that are neither NCA, bugs, nor belong to the family $\mathcal{F}$. One of the difficulties about the family of forbidden graphs is that it contains many graphs of large order that are hard to arrange into a manageable structure; insects and bug are examples of such forbidden graphs.

Problem 1. Show the family of circular-arc graphs that are not NCA.

Problem 2. Show the family of co-bipartite circular-arc graphs that are not NCA.

Besides knowing which circular-arc graphs belong to a restricted class, there is also a motivation for characterizing all those graphs that belong to a given class. As we already mentioned, such a characterization is unknown for circular-arc graphs. But, in this case, the characterization is also unknown for HCA and NCA graphs.

Problem 3. Show the family of minimally forbidden circular-arc graphs.

Problem 4. Show the family of minimally forbidden HCA graphs.

Problem 5. Show the family of minimally forbidden NCA graphs.

All the problems listed above have been open for a long time and they could be too hard problems to begin with. Perhaps, it would be better to begin with the class of NHCA graphs because they retain a lot of properties of interval graphs.

Problem 6. Show the family of minimally forbidden NHCA graphs.

Concave-round graphs. Recall that a PIG order of a graph G is a linear ordering $v_1, \ldots, v_n$ of $V(G)$ such that, for every $1 \leq i \leq n$, there exist two non-negative values l, r such that $N[v_i] = \text{LIN}[v_{i-l}, v_{i+r}]$, and both $\text{LIN}[v_{i-l}, v_i]$ and $\text{LIN}[v_i, v_{i+r}]$ are complete sets. Note that we can drop the conditions asking both of $\text{LIN}[v_{i-l}, v_i]$ and $\text{LIN}[v_i, v_{i+r}]$ to be complete sets. Indeed, if v_i is adjacent to v_j and $j > i+1$, then v_j has to be adjacent to all the vertices of $[v_i, v_j)$ because $N[v_j]$ must be a range.

Concave-round graphs are defined as the generalization of PIG orders, where the linear ordering $v_1, \ldots, v_n$ is replaced with a circular ordering. That is, a graph G is *concave-round* if there is a circular ordering $\phi = v_1, \ldots, v_n$ of $V(G)$ such that, for every $1 \leq i \leq n$, there exist two non-negative values l, r such that $N[v_i] = [v_{i-l}, v_{i+r}]$. The enumeration ϕ is said to be a *concave-round order* of G. Concave-round graphs were introduced by Bang-Jensen et al. [BJHY00] as a generalization of PCA graphs. Clearly, every concave-round order is a circular-arc order, thus every concave-round graph is a circular-arc graph as well.

Not every concave-round enumeration is an out-round enumeration. Consider, for instance, the path graph H with vertices v_1, v_2, v_3, v_4 where v_i is adjacent to v_{i+1} for $1 \leq i \leq 3$. Graph H admits the concave-round enumeration $\phi = v_3, v_1, v_2, v_4$, but ϕ is not an out-round enumeration of any orientation of H. The problem for orienting H, with ϕ as an enumeration, is that v_4 has to be oriented to v_3 and v_3 has to be oriented to v_4. Nevertheless, H admits an out-round orientation. Bang-Jensen et al. proved that if G is a concave-round graph whose complement is not bipartite, then G is a PCA graph.

Problem 7. Study the relationship between concave-round graphs and NCA graphs.

Distance and circulant graphs. Circulant graphs are a natural and important generalization of powers of cycles. A graph G is a *circulant graph* if there is a circular ordering $v_1, \ldots, v_n$ of $V(G)$ and a set $D \subseteq \mathbb{N}$ such that $N(v_i) = \{v_{i+d} \mid d \in D\} \cup \{v_{i-d} \mid d \in D\}$, for every $1 \leq i \leq n$. Circulant graphs are the Cayley graphs of cyclic groups and due to their symmetry and connectivity properties they have been proposed for various practical applications [BCH95].

A similarly defined class of graphs are distance graphs. A graph G is a *distance graph* if there is a linear ordering $v_1, \ldots, v_n$ of $V(G)$ and a set $D \subseteq \mathbb{N}$ such that $N(v_i) = \{v_{i+d} \mid d \in D \text{ and } i+d \leq n\} \cup \{v_{i-d} \mid d \in D \text{ and } i-d \geq 1\}$. Distance graphs are motivated in a research due to Eggleton et al. [EES85, EES90] who considered coloring problems for infinite distance graphs.

In the folklore of graph theory, it is known that every graph is an induced subgraph of a circulant graph. Distance graphs are a generalization of circulant graphs. In fact, if G is a circulant graph for some set D, then G is a distance graph for $D \cup \{n-d \mid d \in D\}$.

Therefore, every graph is an induced subgraph of some distance graph as well (a proof is given in [LRSS09]). The inconvenient aspect of the proof in [LRSS09] is that the generated distance graph has an exponential number of vertices with respect to the size of the graph.

Theorems 4.2.1 and 4.3.1 are interesting because they show how some well studied classes of graphs can be obtained as induced subgraphs of special distance graphs, by restricting the set D. Even more, both UCA and PIG graphs are induced subgraphs of a power of a path with polynomial number of vertices. This is because every UCA graph admits a UCA model whose extremes are integer and whose arcs are of length between 0 and $O(n^2)$ [LS08]. In view of these results, it would be interesting to find more classes which can be defined as induced subgraphs of special distance graphs, and if these classes are generated by polynomial size distance graphs.

Problem 8. Study special classes of circulant and distance graphs, and the classes defined by their induced subgraphs.

Polynomial-time recognition of NCA graphs. Recall that a circular-arc order of a graph G is a circular ordering $v_1, \ldots, v_n$ of $V(G)$ such that, for every pair v_i, v_j of adjacent vertices, either v_i is adjacent to all the vertices in $(v_i, v_j]$ or v_j is adjacent to all the vertices in $(v_j, v_i]$. Those graphs that admit a circular-arc order are precisely the circular-arc graphs. We proved in Chapter 3 that if we change the previous definition by adding an orientation, then the class of NCA graphs is obtained. That is, a graph G is an NCA if and only if it admits an out-round orientation.

Observe that it is easy to transform a circular-arc model of a graph G into a circular-arc order; just take the order induced by the beginning points of the model. Every circular-arc order $\phi = v_1, \ldots, v_n$ is associated with a digraph D such that $v_i \longrightarrow v_j$ if and only if v_i is adjacent to all the vertices in $(v_i, v_j]$. If the digraph D is an oriented graph, then D is an orientation of G and ϕ is an out-round enumeration of D, *i.e.*, G is an NCA graph. Even when G is an NCA graph and ϕ is an out-round enumeration of one of its orientations, the digraph D can fail to be an oriented graph. First, it could happen that some edges are oriented both ways; for instance, every universal vertex is oriented to all the other vertices. In this case, the duplicated edges have to be removed, if possible. This raises the second problem, it is impossible to remove some of the duplicated edges without changing the order of some of the vertices.

Problem 9. Find a polynomial-time algorithm to transform a circular-arc model into an NCA model.

Problem 10. Characterize all those (universal-free) circular-arc graphs for which any circular-arc order induces an out-round orientation as described above.

Linear-time transformation of circular-arc graphs into interval graphs. Recall that our algorithm to transform a circular-arc model $\mathcal{M}$ into an NHCA model has two steps. First, test whether $\mathcal{M}$ is an NHCA model. If so, there is nothing to be done. Otherwise, transform $\mathcal{M}$ into an interval model. We did not find an $O(n)$ time algorithm for this problem. Note that it is enough to consider only the case in which $\mathcal{M}$ is an HCA model.

Problem 11. Devise an $O(n)$ time algorithm to transform an HCA model into an interval model. If possible, output a negative certificate when such transformation is not possible.

Direct recognition of HCA and NHCA graphs. The recognition of HCA graphs is well solved from the viewpoint of time complexity, when a graph G is given as input. First build a circular-arc model $\mathcal{M}$ of G in $O(n+m)$ time and, if successful, transform this model into an HCA model. NHCA graphs can be recognized in $O(n+m)$ time with a similar procedure. The problem with these algorithms is that they require the construction of a circular-arc model, which is not a simple task.

By Theorem 3.0.1, a graph G is an interval graph if and only if there is a linear ordering of the cliques of G such that, for every vertex v, the cliques containing v appear consecutively in the ordering. That is, a graph is an interval graph if its clique matrix has the consecutive-ones property. This theorem justifies the use of the PQ-tree data structure for the recognition of interval graphs. First find the clique matrix of the graph, and then use the PQ-tree data structure to test whether the matrix has the consecutive-ones property. Note that the clique matrix is not really required, only the family of cliques is used, and this family is found in $O(n+m)$ time.

The generalization of Theorem 3.0.1 is given in Theorem 3.0.2. That is, a graph G is an HCA graph if and only if its clique matrix has the circular-ones property. So, we can extend the algorithm of Booth and Lueker, by replacing the PQ-tree with a PC-tree. The PC-tree data structure was introduced by Shih and Hsu [SH99] as a generalization of PQ-trees, with the purpose of recognizing planar graphs. The PC-tree data structure can be efficiently used to test if the clique matrix satisfies the circular-ones property. The major difficulty with this algorithm is that it is not easy to find the family of cliques of an HCA graph. The only algorithm that takes this approach is the one by Gavril [Gav74], and he finds all the cliques by running a general algorithm, at the cost of $O(n^3)$ time.

Problem 12. Find a linear-time algorithm for the recognition of HCA graphs, without computing a circular-arc model of the graph.

Problem 13. Find a linear-time algorithm for computing all the cliques of an HCA graph, without computing an HCA model of the graph.

A less ambitious project is to develop a direct algorithm for the recognition of NHCA graphs. The advantage of restricting the problem to NHCA graphs is that the scope of each vertex looks like an interval graph. The idea for a recognition algorithm could be as follows. First, test whether G is an interval graph. If so, output the interval model of G. Otherwise, try to split G into several interval graphs, resembling the process done by Deng et al. in [DHH96]. Then, recognize each part as an interval graph and "rebuild" the model. Other possibility is to find the cliques of each part so as to build a PC-tree latter. The next possibility is to extend the incremental data structure used by Korte and Möhring [KM89] in their interval graph recognition algorithm. Also, it could be interesting to generalize the incremental algorithm by Crespelle [Cre09], so as to insert each vertex in $O(n)$ time. Here, a dynamic PC-tree data structure should be designed.

Problem 14. Find a linear-time algorithm for the recognition of NHCA graphs, without computing a circular-arc model of the graph.

Problem 15. Find an algorithm for computing all the cliques of an NHCA graph, without computing an HCA model of the graph.

Problem 16. Extend the MPQ-tree data structure by Korte and Mohring, or the dynamic PQ-tree data structure by Crespelle, so as to recognize NHCA graphs.

Minimal UIG models. In Chapter 5 we developed a new algorithm for transforming a PIG model into a UIG model. The advantage of our algorithm over some of the previous algorithms is that it consumes $O(n)$ space. In the same chapter, we also showed that the algorithm by Corneil et al. [CKN+95] can be implemented so as to run in $O(n)$ time and space. We believe that our algorithm is simpler to implement, but its correctness is harder to prove. Also, the simplicity is not such an advantage, taking into account that the algorithm by Corneil et al. is also simple to implement.

There is another aspect in which the algorithms can be compared: the length of the generated model. Say that a UIG model $\mathcal{M}$ of a graph G is *shorter than* the UIG model $\mathcal{M}'$ of G if the length of the intervals in $\mathcal{M}$ is lower than or equal to the length of the intervals in $\mathcal{M}'$. When $\mathcal{M}$ is a shortest model of G, then $\mathcal{M}$ is a *minimal* UIG model. Of course, we assume that the intervals have integer extremes. Although both algorithms have guaranties with respect to the length of the intervals, the algorithms not always provide a minimal UIG model. When comparing both algorithms from the worst case viewpoint, the algorithm by Corneil et al. is better than ours; the length of an interval is always n for the algorithm by Corneil et al., while our algorithm sometimes requires intervals of length $2n$. However, our algorithm always finds a minimal UIG model when G is a power of a path, while the algorithm by Corneil et al. does not.

Problem 17. Find an $O(n)$ time and space algorithm to compute a minimal UIG model of a graph, when a PIG model is given as input.

In [Mit94], Mitas devised an algorithm to find a minimal UIG model of a graph G. The focus of his algorithm is on semiorders, instead of being on UIG graphs. The algorithm takes an acyclic directed graph G such that its transitive closure represents an ordering $<$, and it outputs a UIG model representing $<$. That is, $v < w$ if and only if the interval corresponding to $t(v) < s(w)$. The algorithm takes $O(n+m)$ time, so it is linear when G is given as input. (Note that the transitive closure G is in fact the complement of a UIG graph.) We believe that it should be possible to reduce its complexity to $O(n)$, when a PIG representation of $<$ is given. In such case, the algorithm could be used to transform any PIG model into a minimal UIG model. Nevertheless, the algorithm assumes that the intervals may share the values for their extremes, and so twin intervals are removed. Thus, this algorithm cannot be used to obtain a well formed minimal interval model. It would be interesting to adapt the algorithm by Mitas so as to solve the problem above, or at least the following problem.

Problem 18. Find an $O(n)$ time and space algorithm to compute a minimal UIG model of a twin-free graph, when a PIG model is given as input.

Lin and Szwarcfiter also ask for an algorithm that builds a minimal UCA model in linear time [LS09].

Fully dynamic algorithm for the PCA recognition problem. In Chapter 6 we described a vertex-only fully dynamic algorithm for the recognition of PCA graphs. In that chapter, we did not deal with the problem of inserting and removing edges from the graph. Certainly, we can insert an edge vw into a graph G by first removing the vertex v, and then inserting v with neighborhood $N_G(v) \cup \{w\}$. Similarly, we can remove an edge vw by first removing the vertex v, and then inserting v with neighborhood $N_G(v) \setminus \{w\}$. It is not hard to implement these algorithms so as to run in $O(n)$ time.

Problem 19. Implement a fully dynamic algorithm for the recognition of PCA graphs where the edge operations take $o(n)$ time.

One of the difficulties of the above problem lies in how to manage the co-components of the input graph. Recall that a round block enumeration Φ of a co-bipartite graph can be partitioned into co-bipartite ranges $\mathcal{X}_1, \ldots, \mathcal{X}_s$, where $\mathcal{X}_i$ and $\mathcal{X}_{i+1}$ form a range of Φ. When an edge vw is removed, for $v \in \mathcal{X}_i$ and $w \in \mathcal{X}_j$ with $i \neq j$, the co-bipartite range $\mathcal{X}_i$ has to be joined with $\overline{\mathcal{X}_j}$ (or with its reversal) so as to form a new co-bipartite range. Similarly, the co-bipartite range $\mathcal{X}_j$ has to be joined with $\overline{\mathcal{X}_i}$ (or its reversal). To implement such a join, we need to move some blocks of Φ so as to make $\mathcal{X}_i$ (resp. $\mathcal{X}_j$) consecutive with $\overline{\mathcal{X}_j}$ (resp. $\overline{\mathcal{X}_i}$). On the other hand, when a new edge vw is inserted, for $v \in \mathcal{X}_i$ and $w \in \overline{\mathcal{X}_i}$, we could need to split $\mathcal{X}_i$ and $\overline{\mathcal{X}_i}$ into two co-bipartite ranges each. So, a priori, we need to solve a Union-Find-Split problem to efficiently include the edge operations into our algorithms. Of course, the Union-Find-Split problem is also

present when we need to join or split two contigs of G. So, we could try to use some kind of balanced tree to solve this problem, as it is done for the contigs. The problem in this case is that we chose not to break the co-components into separate structures as it is done with the contigs, so as to simplify the data structures. There are at least two options left: solve the removal and insertion of a co-bipartite range into a ring; or modify the data structure of the algorithm to keep the co-components apart. Of course, a characterization of when is $G \cup \{vw\}$ (resp. $G \setminus \{vw\}$) a PCA graph should also be provided.

Computation of the minimally non-PCA induced subgraph. Kaplan and Nussbaum proposed a linear-time algorithm for computing a negative certificate of a non-PCA graph in [KN09]. The output of their algorithm is either a cycle of odd length in some incompatibility graph, or a minimally forbidden induced subgraph. There is, yet, no algorithm for computing a forbidden induced subgraph of a non-PCA graph.

Problem 20. Devise an $O(n+m)$ time algorithm to find a minimally non-PCA induced subgraph of a non-PCA graph.

In Chapter 6 we provided a characterization of those refinable PCA models. When the input graph G is co-connected, the Incremental Refinement Lemma provides a complete characterization of when is $G \cup \{v\}$ a PCA graph. We believe that the algorithms of that chapter could be adapted so as to find a forbidden induced subgraph of $G \cup \{v\}$. When G is not co-connected, the problem is harder because knowing that a PCA model of G is not refinable is not enough to conclude that $G \cup \{v\}$ is not a PCA graph. For simplicity, we solved the insertion problem by looking all the models formed by the co-components co-adjacent to v. However, we think that some models could be discarded, so as to analyze only one or two models. In that case, it could be possible to characterize when is $G \cup \{v\}$ a PCA graph. Furthermore, we believe that our algorithm could be modified so as to solve the following problem (perhaps with some tedious case by case analysis).

Problem 21. Find an algorithm for computing, in $O(d(v))$ time, a minimally non-PCA induced subgraph of a non-PCA graph $G \cup \{v\}$, when a round block enumeration Φ of G is given.

Characterization of clique graphs of circular-arc graphs. In Chapter 8 we described the structure of clique graphs of HCA graphs, by relating them to the class of PHCA graphs. Theorem 8.1.6 is like the end of the road for a work started by Durán and Lin in [DL01], not only because they set the direction towards this theorem, but also because all the results in their paper follow as corollaries of this theorem. At this point, it seems like a good idea to consider other classes of circular-arc graphs so as to characterize their clique graphs.

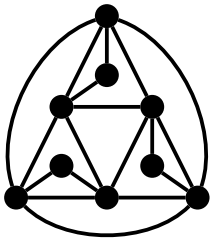


Figure 9.1: A circular-arc graph whose clique graph is not circular-arc.

Problem 22. Give a structural characterization for clique graphs of (normal, proper, unit) circular-arc graphs.

Problem 23. Prove or disprove: the recognition problem for clique graphs of (normal, proper, unit) circular-arc graphs can be solved in polynomial time.

As it is remarked in [Szw03], a large number of the classes whose clique graphs have been characterized so far are K-fixed or K-closed. This is the case for HCA graphs because they are K-closed. One of the difficulties for solving the above problems for circular-arc graphs is the fact that the class is not K-closed (see *e.g.* Figure 9.1). In [Szw03] it is also mentioned that many of the characterized classes $\mathcal{C}$ that are neither K-fixed nor K-closed are such that $K(K(\mathcal{C})) \subseteq \mathcal{C}$. For instance, clique graphs of chordal graphs are dually chordal, and clique graphs of dually chordal graphs are chordal.

Problem 24. Is there a circular-arc graph G such that $K^i(G)$ is not a circular-arc graph for every $i > 0$?

Problem 25. Describe the family of circular-arc graphs whose clique graph is also a circular-arc graph.

Problem 26. Characterize those circular-arc graphs that are also clique graphs.

Problem 27. Prove or disprove: the recognition of clique graphs restricted to the class of circular-arc graphs can be solved in polynomial time.

Bibliography

[AFdFG09] L. Alcón, L. Faria, C. M. de Figueiredo and M. Gutierrez – "The complexity of clique graph recognition", *Theoret. Comput. Sci.* **410** (2009), no. 21-23, p. 2072–2083.

[Asa91] T. Asano – "Dynamic programming on intervals", ISA '91. Algorithms (Taipei, 1991) (W. L. Hsu and R. C. T. Lee, eds.), Lecture Notes in Comput. Sci., vol. 557, Springer, Berlin, 1991, p. 199–207.

[BCH95] J.-C. Bermond, F. Comellas and D. F. Hsu – "Distributed loop computer networks: a survey", *J. Parallel Distrib. Comput.* **24** (1995), no. 1, p. 2–10.

[BDGS09] F. Bonomo, G. Durán, L. N. Grippo and M. D. Safe – "Partial characterizations of circular-arc graphs", *J. Graph Theory* **61** (2009), no. 4, p. 289–306.

[Ben59] S. Benzer – "On the topology of the genetic fine structure", *Proc. Natl. Acad. Sci. USA* **45** (1959), no. 11, p. 1607–1620.

[BES80] L. Babai, P. Erdős and S. M. Selkow – "Random graph isomorphism", *SIAM J. Comput.* **9** (1980), no. 3, p. 628–635.

[BHH96] B. Bhattacharya, P. Hell and J. Huang – "A linear algorithm for maximum weight cliques in proper circular arc graphs", *SIAM J. Discrete Math.* **9** (1996), no. 2, p. 274–289.

[BHZ87] R. B. Boppana, J. Håstad and S. Zachos – "Does co-NP have short interactive proofs?", *Inform. Process. Lett.* **25** (1987), no. 2, p. 127–132.

[BJG01] J. Bang-Jensen and G. Gutin – *Digraphs*, Springer Monographs in Mathematics, Springer-Verlag London Ltd., London, 2001, Theory, algorithms and applications.

[BJHY00] J. Bang-Jensen, J. Huang and A. Yeo – "Convex-round and concave-round graphs", *SIAM J. Discrete Math.* **13** (2000), no. 2, p. 179–193 (electronic).

[BK79] L. Babai and L. Kucera – "Canonical labelling of graphs in linear average time", *20th Annual Symposium on Foundations of Computer Science, 29-31 October 1979, San Juan, Puerto Rico* (Washington, DC, USA), IEEE, 1979, p. 39–46.

[BL76] K. S. Booth and G. S. Lueker – "Testing for the consecutive ones property, interval graphs, and graph planarity using *PQ*-tree algorithms", *J. Comput. System Sci.* **13** (1976), no. 3, p. 335–379, Working Papers presented at the

ACM-SIGACT Symposium on the Theory of Computing (Albuquerque, N. M., 1975).

[BL92] J. A. Bondy and S. C. Locke – "Triangle-free subgraphs of powers of cycles", *Graphs Combin.* **8** (1992), no. 2, p. 109–118.

[Bon06] F. Bonomo – "Self-clique Helly circular-arc graphs", *Discrete Math.* **306** (2006), no. 6, p. 595–597.

[Boo80] K. S. Booth – "Lexicographically least circular substrings", *Inform. Process. Lett.* **10** (1980), no. 4-5, p. 240–242.

[BP89] J.-C. Bermond and C. Peyrat – "Induced subgraphs of the power of a cycle", *SIAM J. Discrete Math.* **2** (1989), no. 4, p. 452–455.

[BW99] K. P. Bogart and D. B. West – "A short proof that "proper = unit"", *Discrete Math.* **201** (1999), no. 1-3, p. 21–23.

[CDG01] G. Confessore, P. Dell'Olmo and S. Giordani – "An approximation result for a periodic allocation problem", *Discrete Appl. Math.* **112** (2001), no. 1-3, p. 53–72, Combinatorial Optimization Symposium (Brussels, 1998).

[CdM07] C. N. Campos and C. P. de Mello – "A result on the total colouring of powers of cycles", *Discrete Appl. Math.* **155** (2007), no. 5, p. 585–597.

[CKN+95] D. G. Corneil, H. Kim, S. Natarajan, S. Olariu and A. P. Sprague – "Simple linear time recognition of unit interval graphs", *Inform. Process. Lett.* **55** (1995), no. 2, p. 99–104.

[Cor04] D. G. Corneil – "A simple 3-sweep LBFS algorithm for the recognition of unit interval graphs", *Discrete Appl. Math.* **138** (2004), no. 3, p. 371–379.

[COS98] D. G. Corneil, S. Olariu and L. Stewart – "The ultimate interval graph recognition algorithm? (extended abstract)", *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 1998)* (New York), ACM, 1998, p. 175–180.

[CP05] C. Crespelle and C. Paul – "Fully dynamic algorithm for recognition and modular decomposition of permutation graphs", Graph-theoretic concepts in computer science (D. Kratsch, ed.), Lecture Notes in Comput. Sci., vol. 3787, Springer, Berlin, 2005, p. 38–48.

[CP06] C. Crespelle and C. Paul – "Fully dynamic recognition algorithm and certificate for directed cographs", *Discrete Appl. Math.* **154** (2006), no. 12, p. 1722–1741.

[CP08] T. Czajka and G. Pandurangan – "Improved random graph isomorphism", *J. Discrete Algorithms* **6** (2008), no. 1, p. 85–92.

[Cre09] C. Crespelle – "Fully dynamic representations of interval graphs", Graph-Theoretic Concepts in Computer Science, Lecture Notes in Comput. Sci., Springer, 2009, to appear.

[Cur07] A. R. Curtis – *Linear-time graph algorithms for chordal comparability graphs and Helly circular arc graphs*, M.Sc. Thesis, Colorado State University, 2007.

[DG99] J. S. Deogun and K. Gopalakrishnan – "Consecutive retrieval property—revisited", *Inform. Process. Lett.* **69** (1999), no. 1, p. 15–20.

[DGM+06] G. Durán, A. Gravano, R. M. McConnell, J. Spinrad and A. Tucker – "Polynomial time recognition of unit circular-arc graphs", *J. Algorithms* **58** (2006), no. 1, p. 67–78.

[DHH96] X. Deng, P. Hell and J. Huang – "Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs", *SIAM J. Comput.* **25** (1996), no. 2, p. 390–403.

[DL01] G. Durán and M. C. Lin – "Clique graphs of Helly circular-arc graphs", *Ars Combin.* **60** (2001), p. 255–271.

[DLMS06] G. Durán, M. C. Lin, S. Mera and J. L. Szwarcfiter – "Algorithms for clique-independent sets on subclasses of circular-arc graphs", *Discrete Appl. Math.* **154** (2006), no. 13, p. 1783–1790.

[dMML06] C. P. de Mello, M. A. Morgana and M. Liverani – "The clique operator on graphs with few P_4's", *Discrete Appl. Math.* **154** (2006), no. 3, p. 485–492.

[Dra89] F. F. Dragan – "Center of graphs and the Helly property", Ph.D. Thesis, Moldova State University, Chişinău, Moldova, 1989.

[EES85] R. B. Eggleton, P. Erdős and D. K. Skilton – "Colouring the real line", *J. Combin. Theory Ser. B* **39** (1985), no. 1, p. 86–100.

[EES90] — , "Colouring prime distance graphs", *Graphs Combin.* **6** (1990), no. 1, p. 17–32.

[EGI99] D. Eppstein, Z. Galil and G. F. Italiano – "Dynamic graph algorithms", Algorithms and theory of computation handbook (M. J. Atallah, ed.), CRC, Boca Raton, FL, 1999, p. 8–1–8–25.

[EK03] B. Effantin and H. Kheddouci – "The b-chromatic number of some power graphs", *Discrete Math. Theor. Comput. Sci.* **6** (2003), no. 1, p. 45–54 (electronic).

[ES93] E. M. Eschen and J. P. Spinrad – "An $O(n^2)$ algorithm for circular-arc graph recognition", *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (Austin, TX, 1993)* (New York), ACM, 1993, p. 128–137.

[Esc73] F. Escalante – "Über iterierte Clique-Graphen", *Abh. Math. Sem. Univ. Hamburg* **39** (1973), p. 59–68.

[FANLP04] M. E. Frías-Armenta, V. Neumann-Lara and M. A. Pizaña – "Dismantlings and iterated clique graphs", *Discrete Math.* **282** (2004), no. 1-3, p. 263–265.

[FG65] D. R. Fulkerson and O. A. Gross – "Incidence matrices and interval graphs", *Pacific J. Math.* **15** (1965), p. 835–855.

[Fis85] P. C. Fishburn – *Interval orders and interval graphs*, Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons Ltd., Chichester, 1985, A study of partially ordered sets, A Wiley-Interscience Publication.

[Gar07] F. Gardi – "The Roberts characterization of proper and unit interval graphs", *Discrete Math.* **307** (2007), no. 22, p. 2906–2908.

[Gav74] F. Gavril – "Algorithms on circular-arc graphs", *Networks* **4** (1974), p. 357–369.

[GGKS95] P. W. Goldberg, M. C. Golumbic, H. Kaplan and R. Shamir – "Four strikes against physical mapping of DNA.", *J. Comput. Biol.* **2** (1995), no. 1, p. 139–152.

[GH64] P. C. Gilmore and A. J. Hoffman – "A characterization of comparability graphs and of interval graphs", *Canad. J. Math.* **16** (1964), p. 539–548.

[GH88] M. C. Golumbic and P. L. Hammer – "Stability in circular arc graphs", *J. Algorithms* **9** (1988), no. 3, p. 314–320.

[Gho72] S. P. Ghosh – "File organization: the consecutive retrieval property", *Commun. ACM* **15** (1972), no. 9, p. 802–808.

[GJ79] M. R. Garey and D. S. Johnson – *Computers and intractability*, W. H. Freeman and Co., San Francisco, Calif., 1979, A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[GJMP80] M. R. Garey, D. S. Johnson, G. L. Miller and C. H. Papadimitriou – "The complexity of coloring circular arcs and chords", *SIAM J. Algebraic Discrete Methods* **1** (1980), no. 2, p. 216–227.

[Gol04] M. C. Golumbic – *Algorithmic graph theory and perfect graphs*, second ed., Annals of Discrete Mathematics, vol. 57, Elsevier Science B.V., Amsterdam, 2004, With a foreword by Claude Berge.

[GS93] M. C. Golumbic and R. Shamir – "Complexity and algorithms for reasoning about time: a graph-theoretic approach", *J. Assoc. Comput. Mach.* **40** (1993), no. 5, p. 1108–1133.

[Haj57] G. Hajós – "Über eine Art von Graphen", *Intern. Math. Nachr* **11** (1957), Problem 65.

[Ham68] R. C. Hamelink – "A partial characterization of clique graphs", *J. Combinatorial Theory* **5** (1968), p. 192–197.

[HD64] H. Hadwiger and H. Debrunner – *Combinatorial geometry in the plane*, Translated by Victor Klee. With a new chapter and other additional material supplied by the translator, Holt, Rinehart and Winston, New York, 1964.

[HdFMPdM95] C. M. Herrera de Figueiredo, J. Meidanis and C. Picinin de Mello – "A linear-time algorithm for proper interval graph recognition", *Inform. Process. Lett.* **56** (1995), no. 3, p. 179–184.

[Hed84] B. Hedman – "Clique graphs of time graphs", *J. Combin. Theory Ser. B* **37** (1984), no. 3, p. 270–278.

[HH95] P. Hell and J. Huang – "Lexicographic orientation and representation algorithms for comparability graphs, proper circular arc graphs, and proper interval graphs", *J. Graph Theory* **20** (1995), no. 3, p. 361–374.

[HH04] — , "Interval bigraphs and circular arc graphs", *J. Graph Theory* **46** (2004), no. 4, p. 313–327.

[HH05] — , "Certifying LexBFS recognition algorithms for proper interval graphs and proper interval bigraphs", *SIAM J. Discrete Math.* **18** (2005), no. 3, p. 554–570 (electronic).

[HM91] W. L. Hsu and T. H. Ma – "Substitution decomposition on chordal graphs and applications", ISA '91. Algorithms (Taipei, 1991) (W. L. Hsu and R. C. T. Lee, eds.), Lecture Notes in Comput. Sci., vol. 557, Springer, Berlin, 1991, p. 52–60.

[HM99] W.-L. Hsu and T.-H. Ma – "Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs", *SIAM J. Comput.* **28** (1999), no. 3, p. 1004–1020 (electronic).

[HMPV00] M. Habib, R. McConnell, C. Paul and L. Viennot – "Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing", *Theoret. Comput. Sci.* **234** (2000), no. 1-2, p. 59–84.

[HSS01] P. Hell, R. Shamir and R. Sharan – "A fully dynamic algorithm for recognizing and representing proper interval graphs", *SIAM J. Comput.* **31** (2001), no. 1, p. 289–305 (electronic).

[Hsu93] W. L. Hsu – "A simple test for interval graphs", Graph-theoretic concepts in computer science (Wiesbaden-Naurod, 1992) (E. W. Mayr, ed.), Lecture Notes in Comput. Sci., vol. 657, Springer, Berlin, 1993, p. 11–16.

[Hsu95] — , "$O(M \cdot N)$ algorithms for the recognition and isomorphism problems on circular-arc graphs", *SIAM J. Comput.* **24** (1995), no. 3, p. 411–439.

[Hsu96] — , "On-line recognition of interval graphs in $O(m + n \log n)$ time", *Combinatorics and computer science (Brest, 1995)* (Berlin) (M. Deza, R. Euler and I. Manoussakis, eds.), Lecture Notes in Comput. Sci., vol. 1120, Springer, 1996, p. 27–38.

[HT91] W. L. Hsu and K.-H. Tsai – "Linear time algorithms on circular-arc graphs", *Inform. Process. Lett.* **40** (1991), no. 3, p. 123–129.

[Hua92] J. Huang – "Tournament-like oriented graphs", Ph.D. Thesis, Simon Fraser University, 1992.

[Hua95] — , "On the structure of local tournaments", *J. Combin. Theory Ser. B* **63** (1995), no. 2, p. 200–221.

[Iba08] L. Ibarra – "Fully dynamic algorithms for chordal graphs and split graphs", *ACM Trans. Algorithms* **4** (2008), no. 4, p. 1–20.

[Iba09a] L. Ibarra – "A fully dynamic graph algorithm for recognizing interval graphs", *Algorithmica* (2009), accepted.

[Iba09b] — , "A fully dynamic graph algorithm for recognizing proper interval graphs", *WALCOM: Algorithms and Computation* (S. Das and R. Uehara, eds.), Lecture Notes in Comput. Sci., vol. 5431, Springer, 2009, p. 190–201.

[JLM+09] B. Joeris, M. C. Lin, R. M. McConnell, J. Spinrad and J. L. Szwarcfiter – "Linear-time recognition of helly circular-arc models and graphs", *Algorithmica* (2009), available online.

[Ken69] D. G. Kendall – "Incidence matrices, interval graphs and seriation in archeology", *Pacific J. Math.* **28** (1969), p. 565–570.

[Kle69] V. Klee – "Research Problems: What Are the Intersection Graphs of Arcs in a Circle?", *Amer. Math. Monthly* **76** (1969), no. 7, p. 810–813.

[KM89] N. Korte and R. H. Möhring – "An incremental linear-time algorithm for recognizing interval graphs", *SIAM J. Comput.* **18** (1989), no. 1, p. 68–81.

[KMMS06] D. Kratsch, R. M. McConnell, K. Mehlhorn and J. P. Spinrad – "Certifying algorithms for recognizing interval graphs and permutation graphs", *SIAM J. Comput.* **36** (2006), no. 2, p. 326–353 (electronic).

[KN04] M. Krivelevich and A. Nachmias – "Colouring powers of cycles from random lists", *European J. Combin.* **25** (2004), no. 7, p. 961–968.

[KN06] H. Kaplan and Y. Nussbaum – "A simpler linear-time recognition of circular-arc graphs", Algorithm theory—SWAT 2006 (L. Arge and R. Freivalds, eds.), Lecture Notes in Comput. Sci., vol. 4059, Springer, Berlin, 2006, p. 41–52.

[KN09] — , "Certifying algorithms for recognizing proper circular-arc graphs and unit circular-arc graphs", *Discrete Appl. Math.* **157** (2009), no. 15, p. 3216–3230.

[KST93] J. Köbler, U. Schöning and J. Torán – *The graph isomorphism problem: its structural complexity*, Progress in Theoretical Computer Science, Birkhäuser Boston Inc., Boston, MA, 1993.

[kY06] T. kyzy Yrysgul – "A fully dynamic algorithm for recognizing and representing chordal graphs", Ershov Memorial Conference (I. Virbitskaite and A. Voronkov, eds.), Lecture Notes in Comput. Sci., vol. 4378, Springer, 2006, p. 481–486.

[LB63] C. G. Lekkerkerker and J. C. Boland – "Representation of a finite graph by a set of intervals on the real line", *Fund. Math.* **51** (1962/1963), p. 45–64.

[LB79] G. S. Lueker and K. S. Booth – "A linear time algorithm for deciding interval graph isomorphism", *J. Assoc. Comput. Mach.* **26** (1979), no. 2, p. 183–195.

[LdMM+04] F. Larrión, C. P. de Mello, A. Morgana, V. Neumann-Lara and M. A. Pizaña – "The clique operator on cographs and serial graphs", *Discrete Math.* **282** (2004), no. 1-3, p. 183–191.

[LL07] D. Li and M. Liu – "Hadwiger's conjecture for powers of cycles and their complements", *European J. Combin.* **28** (2007), no. 4, p. 1152–1155.

[LNLP09] F. Larrión, V. Neumann-Lara and M. A. Pizaña – "On expansive graphs", *European J. Combin.* **30** (2009), no. 2, p. 372–379.

[LO93] P. J. Looges and S. Olariu – "Optimal greedy algorithms for indifference graphs", *Comput. Math. Appl.* **25** (1993), no. 7, p. 15–25.

[LRSS09] M. C. Lin, D. Rautenbach, F. J. Soulignac and J. L. Szwarcfiter – "Powers of cycles, powers of paths, and distance graphs", preprint, 2009.

[LS06] M. C. Lin and J. L. Szwarcfiter – "Characterizations and linear time recognition of Helly circular-arc graphs", Computing and combinatorics (D. Z. Chen and D. T. Lee, eds.), Lecture Notes in Comput. Sci., vol. 4112, Springer, Berlin, 2006, p. 73–82.

[LS08] — , "Unit circular-arc graph representations and feasible circulations", *SIAM J. Discrete Math.* **22** (2008), no. 1, p. 409–423.

[LS09] — , "Characterizations and recognition of circular-arc graphs and subclasses: A survey", *Discrete Math.* **309** (2009), no. 18, p. 5618–5635, Combinatorics 2006, A Meeting in Celebration of Pavol Hell's 60th Birthday (May 1-5, 2006).

[LSS08] M. C. Lin, F. J. Soulignac and J. L. Szwarcfiter – "A simple linear time algorithm for the isomorphism problem on proper circular-arc graphs", Algorithm theory—SWAT 2008 (J. Gudmundsson, ed.), Lecture Notes in Comput. Sci., vol. 5124, Springer, Berlin, 2008, p. 355–366.

[Luc56] R. D. Luce – "Semiorders and a theory of utility discrimination", *Econometrica* **24** (1956), p. 178–191.

[McC03] R. M. McConnell – "Linear-time recognition of circular-arc graphs", *Algorithmica* **37** (2003), no. 2, p. 93–147.

[Mit94] J. Mitas – "Minimal representation of semiorders with intervals of same length", Orders, algorithms, and applications (Lyon, 1994) (V. Bouchitté and M. Morvan, eds.), Lecture Notes in Comput. Sci., vol. 831, Springer, Berlin, 1994, p. 162–175.

[Mül97] H. Müller – "Recognizing interval digraphs and interval bigraphs in polynomial time", *Discrete Appl. Math.* **78** (1997), no. 1-3, p. 189–205.

[NL78] V. Neumann-Lara – "On clique-divergent graphs", Problèmes Combinatoires et Théorie des Graphes, Colloques Internationaux C.N.R.S., vol. 260, C.N.R.S., 1978, p. 313–315.

[NPP06] S. D. Nikolopoulos, L. Palios and C. Papadopoulos – "A fully dynamic algorithm for the recognition of P_4-sparse graphs", Graph-theoretic concepts in computer science (F. V. Fomin, ed.), Lecture Notes in Comput. Sci., vol. 4271, Springer, Berlin, 2006, p. 256–268.

[Nus08] Y. Nussbaum – "From a circular-arc model to a proper circular-arc model", Graph-Theoretic Concepts in Computer Science (H. Broersma, T. Erlebach,

T. Friedetzky and D. Paulusma, eds.), Lecture Notes in Comput. Sci., vol. 5344, Springer, Berlin, 2008, p. 324–335.

[Ola91] S. Olariu – "An optimal greedy heuristic to color interval graphs", *Inform. Process. Lett.* **37** (1991), no. 1, p. 21–25.

[Pri92] E. Prisner – "Convergence of iterated clique graphs", *Discrete Math.* **103** (1992), no. 2, p. 199–207.

[PS97] I. Pe'er and R. Shamir – "Realizing interval graphs with size and distance constraints", *SIAM J. Discrete Math.* **10** (1997), no. 4, p. 662–687.

[Rob69] F. S. Roberts – "Indifference graphs", Proof Techniques in Graph Theory (Proc. Second Ann Arbor Graph Theory Conf., Ann Arbor, Mich., 1968), Academic Press, New York, 1969, p. 139–146.

[Rob78] — , *Graph theory and its applications to problems of society*, CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 29, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pa., 1978.

[RS71] F. S. Roberts and J. H. Spencer – "A characterization of clique graphs", *J. Combinatorial Theory Ser. B* **10** (1971), p. 102–108.

[RTL76] D. J. Rose, R. E. Tarjan and G. S. Lueker – "Algorithmic aspects of vertex elimination on graphs", *SIAM J. Comput.* **5** (1976), no. 2, p. 266–283.

[Sch88] U. Schöning – "Graph isomorphism is in the low hierarchy", *J. Comput. System Sci.* **37** (1988), no. 3, p. 312–323.

[SE04] S. Stefanakos and T. Erlebach – "Routing in all-optical ring networks revisited", *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, vol. 1, June-1 July 2004, p. 288–293 Vol.1.

[SH99] W.-K. Shih and W.-L. Hsu – "A new planarity test", *Theoret. Comput. Sci.* **223** (1999), no. 1-2, p. 179–191.

[Shi81] Y. Shiloach – "Fast canonization of circular strings", *J. Algorithms* **2** (1981), no. 2, p. 107–121.

[Skr82] D. J. Skrien – "A relationship between triangulated graphs, comparability graphs, proper interval graphs, proper circular-arc graphs, and nested interval graphs", *J. Graph Theory* **6** (1982), no. 3, p. 309–316.

[Spi03] J. P. Spinrad – *Efficient graph representations*, Fields Institute Monographs, vol. 19, American Mathematical Society, Providence, RI, 2003.

[SS58] D. Scott and P. Suppes – "Foundational aspects of theories of measurement", *J. Symb. Logic* **23** (1958), p. 113–128.

[SS04] R. Shamir and R. Sharan – "A fully dynamic algorithm for modular decomposition and recognition of cographs", *Discrete Appl. Math.* **136** (2004), no. 2-3, p. 329–340, The 1st Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW 2001).

[Sta67] F. W. Stahl – "Circular genetic maps", *J. Cell. Physiol.* **70** (1967), no. S1, p. 1–12.

[Sto68] K. E. Stoffers – "Scheduling of traffic lights—a new approach", *Trasport. Res.* **2** (1968), p. 199–234.

[Szw97] J. L. Szwarcfiter – "Recognizing clique-Helly graphs", *Ars Combin.* **45** (1997), p. 29–32.

[Szw03] J. L. Szwarcfiter – "A survey on clique graphs", Recent advances in algorithms and combinatorics (C. Linhares Sales and B. Reed, eds.), CMS Books Math./Ouvrages Math. SMC, vol. 11, Springer, New York, 2003, p. 109–136.

[TC07] M. Tedder and D. Corneil – "An optimal, edges-only fully dynamic algorithm for distance-hereditary graphs", STACS 2007 (W. Thomas and P. Weil, eds.), Lecture Notes in Comput. Sci., vol. 4393, Springer, Berlin, 2007, p. 344–355.

[Tho05] C. Thomassen – "Some remarks on Hajós' conjecture", *J. Combin. Theory Ser. B* **93** (2005), no. 1, p. 95–105.

[Tuc74] A. Tucker – "Structure theorems for some circular-arc graphs", *Discrete Math.* **7** (1974), p. 167–195.

[Tuc78] — , "Circular arc graphs: new uses and a new algorithm", Theory and applications of graphs (Proc. Internat. Conf., Western Mich. Univ., Kalamazoo, Mich., 1976) (Y. Alavi and D. R. Lick, eds.), Springer, Berlin, 1978, p. 580–589. Lecture Notes in Math., Vol. 642.

[Tuc80] — , "An efficient test for circular-arc graphs", *SIAM J. Comput.* **9** (1980), no. 1, p. 1–24.

[Weg67] G. Wegner – "Eigenschaften der Nerven homologische-einfactor Familien im R^n", Ph.D. Thesis, Universität Göttingen, Göttingen, Germany, 1967.