Structured Families of Graphs: Properties, Algorithms, and Representations

Marina Lipshteyn

A THESIS SUBMITTED FOR THE DEGREE
"DOCTOR OF PHILOSOPHY"

University of Haifa
Graduate Studies Authority
The Committee for Doctoral Studies Science

November, 2005

Structured Families of Graphs: Properties, Algorithms, and Representations

By: Marina Lipshteyn

Supervised by: Prof. Martin Charles Golumbic

A THESIS SUBMITTED FOR THE DEGREE
"DOCTOR OF PHILOSOPHY"

University of Haifa
Graduate Studies Authority
The Committee for Doctoral Studies Science

November, 2005

Recommended by: _____ Date: _____
(Advisor)

Approved by: _____ Date: _____
(Chairman of Ph.D Committee)

Acknowledgements

The work with this dissertation has been extensive and trying, but in the first place exciting, instructive, and fun. Without help, support, and encouragement from several persons, I would never have been able to finish this work.

I would like to express my sincere gratitude and appreciation to my advisor, Professor Martin Charles Golumbic, for providing me with the unique opportunity to work in the research area of graph theory, for his inspiring and encouraging way to guide me to a deeper understanding, for his invaluable comments during the whole work with this dissertation and for his encouragement and support at all levels.

A special thanks goes to my colleague Dr. Michal Stern, who is most responsible for helping me complete the writing of this dissertation as well as the challenging research that lies behind it. Michal has been a friend and colleague. We spent uncountable hours of work together. She was always there to meet and discuss my ideas. Without her encouragement and constant guidance, I could not have finished this dissertation.

I will also give a special thanks to Dr. Anne Berry for a fruitful collaboration and as co-author.

Thanks also to all my colleagues at the Computer Science Department and the Caesarea Edmond Benjamin de Rothschild Foundation Institute for Interdisciplinary Applications of Computer Science (C.R.I.) for providing a good working atmosphere.

Besides my advisors, I would like to thank the rest of my thesis committee: Prof. Andreas Brandstädt and Ross McConnell, who gave insightful comments and reviewed my work on a very short notice.

Marina Lipshteyn

Contents

Marina Lipshteyn

Abstract

The intersection graph of a collection of sets $\mathcal{F}$ is the graph obtained by assigning a distinct vertex to each set in $\mathcal{F}$ and joining two vertices by an edge precisely when their corresponding sets have a nonempty intersection. When $\mathcal{F}$ is allowed to be an arbitrary family of sets, the class of graphs obtained as intersection graphs is simply all undirected graphs. When the types of sets allowed in $\mathcal{F}$ is limited, interesting structured families of graphs result. The problem of characterizing the intersection graphs of families of sets having some specific topological or other pattern is often very interesting and frequently has applications to the real world.

One of the well-known structured families of intersection graphs is the family of interval graphs. The interval graphs arise when the sets in $\mathcal{F}$ are intervals in the real line, that is, a graph $G = (V, E)$ is an interval graph if each vertex $v \in V$ can be assigned a real interval I_v so that $xy \in E \Leftrightarrow I_x \cap I_y \neq \emptyset$. The set of intervals $\{I_v | v \in V\}$ is an interval representation of G.

The class of tolerance graphs is a generalization of interval graphs. Tolerance graphs are constructed from intersecting intervals in a manner similar to interval graphs, by putting an edge between two vertices depends on measuring the size of the intersection of their two intervals before declaring that an edge exists.

This work was mainly motivated by the study of tolerance graphs, where we have investigated families of graphs that are not necessarily tolerance, but generalize the tolerance model. Our earlier work [26], focused on the interval probe graphs, a subfamily of tolerance graphs, that are used in physical mapping of DNA. The new results reported in the thesis are organized in two major parts. In Part I we introduce the chordal probe graphs, which are a generalization of interval probe graphs, but are not a subfamily of tolerance graphs. In Part II we investigate an approach to generalizing tolerance graphs by replacing the real line by a tree and replacing the role of intervals by either paths or other types of subtree. Part II deals with tolerance models of paths and other types of subtrees. In particular, we study the edge intersection graphs of paths in a tree.

In Part I, we introduce the class of chordal probe graphs which are a generalization of both interval probe graphs and chordal graphs. A graph $G = (V, E)$ is chordal probe if its vertices can be partitioned into two sets P (probes) and N (non-probes) where N is a stable set and such that G can be extended to a chordal graph by adding edges between non-probes.

We show that chordal probe graphs may contain neither an odd-length chordless cycle nor the complement of a chordless cycle, hence they are perfect graphs. We

present a complete hierarchy with separating examples for chordal probe and related classes of graphs.

We give several characterizations of chordal probe graphs, first in the case of a fixed given partition of the vertices into probes and non-probes, and second in the more general case where no partition is given. In both of these cases, our results are obtained by introducing new classes, namely, N-triangulatable graphs and cycle-bicolorable graphs. We give polynomial time recognition algorithms for each class. We also give polynomial time recognition algorithms for the subfamily of chordal probe graphs which are also weakly chordal, both in the case of a fixed given partition of the vertices into probes and non-probes, and in the more general case where no partition is given.

N-triangulatable graphs have properties similar to chordal graphs, and we characterize them using graph separators and using a vertex elimination ordering. Cycle-bicolorable graphs are shown to be perfect, and any cycle-bicoloring of a graph renders it N-triangulatable.

The corresponding recognition complexity for chordal probe graphs, given a partition of the vertices into probes and non-probes, is $O(|P||E|)$. If no partition is given in advance, the complexity of our recognition algorithm is $O(|V|^2|E|)$.

Part II deals with tolerance models of paths and other types of subtrees. Intersection graphs of intervals on the line (interval graphs) are generalized to tolerance graphs, by adding an edge between two vertices in the tolerance graph when the size of the intersection of their intervals exceeds at least one of the tolerances. In the same fashion, one can generalize intersection graphs of paths in a tree (EPT) to tolerance intersection graphs of paths in a tree, by adding an edge between two vertices in the model, when the size of the intersection of their paths exceeds at least one of the tolerances. Thus, k-EPT graphs are tolerance intersection graphs of paths in a tree with constant tolerance corresponding to k edges in the tree. The edge intersection graphs are used in network applications. The EPT graphs are useful in network applications. Scheduling undirected calls in a tree or assigning wavelengths to virtual connections in an optical tree network are equivalent to coloring its EPT graph.

Let $\mathcal{P}$ be a collection of nontrivial simple paths in a tree T. We define the k-edge ($k \geq 1$) intersection graph $k\text{-}EPT(\mathcal{P})$, whose vertices correspond to the members of $\mathcal{P}$, and two vertices are joined by an edge if the corresponding members of $\mathcal{P}$ share k edges in T. An undirected graph G is called a k-edge intersection graph of paths in a tree, and denoted by k-EPT, if $G = k\text{-}EPT(\mathcal{P})$ for some $\mathcal{P}$ and T, where $\langle \mathcal{P}, T \rangle$ is called a k-EPT representation of G.

It is known that the recognition and the coloring of the 1-EPT graphs are NP-complete. We extend this result and prove that the recognition and the coloring of the k-EPT graphs are also NP-complete for any fixed $k > 1$. We show that the problem of finding the largest clique on k-EPT graphs is polynomial, as was the case for 1-EPT graphs. We prove that the family of 1-EPT graphs is contained in, but not equal to, the family of k-EPT graphs for any fixed $k \geq 2$.

It is well-known that chordal graphs correspond to vertex intersection graphs of subtrees on a tree. An undirected graph G is weakly chordal if every cycle of length greater than 4 in G and in its complement $\overline{G}$ possesses a chord. It is known that recognition and coloring of EPT graphs are NP-complete problems. However, the

EPT graphs restricted to host trees of vertex degree 3 are precisely the chordal EPT graphs, and therefore can be colored in polynomial time complexity. We prove a new analogous result that weakly chordal EPT graphs are precisely the EPT graphs with host tree restricted to degree 4. This also implies that the coloring of the edge intersection graph of paths in a degree 4 tree is polynomial. Moreover, we introduce an algorithm to reduce a given EPT representation of a weakly chordal EPT graph to an EPT representation on a degree 4 tree.

Moreover, it is known that chordal graphs correspond to intersection graphs of binary subtrees on a tree that have an orthodox representation. Motivated by known results on chordal graphs and subtrees of a tree, we consider variations of k-EPT graphs by three main parameters: maximal degree of a host tree, the number of intersecting edges of paths and whether the graph has an orthodox representation. We give the complete hierarchy of relationships between classes of weakly chordal, chordal, K-EPT and orthodox k-EPT graphs.

The results in this work were published in [8, 27, 28, 29], accepted for publication in [30] and submitted for publication in [9, 31, 32].

List of Tables

List of Figures

1 Introduction

Graph theory has developed into a very broad field since its introduction by Euler in 18th century. Graphs are an intuitively pleasing and flexible method for dealing with relationships between objects, and are used in so many areas of mathematical research that they need no justification for study in their own right. In the mathematical tradition, graphs and classes of graphs are studied with no specific applications in mind, although many of the classes discussed arise from important applications.

The mathematical problems in graph theory typically relate to a variety of other combinatorial structures and therefore are connected with many difficult practical problems. Elaboration of new theoretical structures has motivated a search for new algorithms compatible with those structures. A set of graphs defined by a structure is called a structured family of graphs. In this work we define new combinatorial structures, and therefore new structured families of graphs. We investigate properties, algorithms and representations of these graphs.

A graph $G = (V, E)$ consists of a finite set V and a set $E = \{(x, y)|x, y \in V\}$. The set V is called the vertex set of the graph and E is the edge set of the graph. A representation of a graph G is any formal description of the binary relation on the set V. The graphs in this work are undirected, finite and simple (i.e., without self-loops and parallel edges).

The intersection graph of a collection of sets $\mathcal{F}$ is the graph obtained by assigning a distinct vertex to each set in $\mathcal{F}$ and joining two vertices by an edge precisely when their corresponding sets have a nonempty intersection. When $\mathcal{F}$ is allowed to be an arbitrary family of sets, the class of graphs obtained as intersection graphs is simply all undirected graphs. When the types of sets allowed in $\mathcal{F}$ is limited, interesting structured families of graphs result. The problem of characterizing the intersection graphs of families of sets having some specific topological or other pattern is often very interesting and frequently has applications to the real world.

One of the well-known structured families of intersection graphs is the family of

interval graphs. The interval graphs arise when the sets in $\mathcal{F}$ are intervals in the real line, that is, a graph $G = (V, E)$ is an interval graph if each vertex $v \in V$ can be assigned a real interval I_v so that $xy \in E \Leftrightarrow I_x \cap I_y \neq \emptyset$. The set of intervals $\{I_v | v \in V\}$ is an interval representation of G.

The class of tolerance graphs is a generalization of interval graphs. Tolerance graphs are constructed from intersecting intervals in a manner similar to interval graphs, by putting an edge between two vertices depends on measuring the size of the intersection of their two intervals before declaring that an edge exists. Tolerance graphs were introduced by Golumbic and Monma in 1982 [33] to generalize some of the well-known applications associated with interval graphs. In 2004, a survey book on tolerance graphs was published by Golumbic and Trenk [34]. The family of tolerance graphs contains a lot of well-studied structured families of graphs. For example: parallelogram, permutation, interval, threshold and interval probe graphs.

A study of a structural family of graphs always includes the recognition problem. The recognition problem for a class of graphs is the problem of determining whether an input graph is in the class. Sometimes, we can use algorithms for recognizing whether a graph is in the class directly to efficient algorithms for constructing a representation for the class. In other cases, we may suspect that an input graph is from a class which allows a good representation or can be optimized efficiently. In such cases, we often first perform a recognition algorithm to see if the graph is in the class, and if it is in the class we use a different algorithm designed to construct a representation or solve the optimization problem correctly given the assumption that the input is in the specified class of graphs. In the characterization problem, we are looking for a representation which is possible if and only if the graph is in the class. In this work, we will solve recognition and characterization problems for several families of graphs.

This work was mainly motivated by the study of tolerance graphs, where we have investigated families of graphs that are not necessarily tolerance, but generalize the tolerance model. Our earlier work [26], focused on the interval probe graphs,

a subfamily of tolerance graphs, that are used in physical mapping of DNA. The new results reported in the thesis are organized in two major parts. In Part I we introduce the chordal probe graphs, which are a generalization of interval probe graphs, but are not a subfamily of tolerance graphs. In Part II we investigate an approach to generalizing tolerance graphs by replacing the real line by a tree and replacing the role of intervals by either paths or other types of subtree. Part II deals with tolerance models of paths and other types of subtrees. In particular, we study the edge intersection graphs of paths in a tree.

Basic graph theory definitions

All standard definitions of terms we use can be found in [22, 51]. The following are some basic definitions:

The complement $\overline{G} = (V, \overline{E})$ of G has the same set of vertices and edge set defined by $\overline{E} = \{(x, y) \mid x, y \in V \text{ and } x \neq y \text{ and } (x, y) \notin E\}$. Given a subset $A \subset V$, the subgraph induced by a vertex set A is denoted by $G_A = (A, E(G_A))$, where $E(G_A) = \{(v, w) \in E \mid v \in A \text{ and } w \in A\}$. Similarly, $\overline{G}_A$ denotes the subgraph induced by A in the complement $\overline{G}$ of G. A set $A \subseteq V$ is a **stable set** in G if for all $u, v \in A$, $(u, v) \notin E$. A set $A \subseteq V$ is a **clique** in G if for all $u, v \in A$, $u \neq v$, $(u, v) \in E$. If $V(G)$ is a clique, then G is a **complete** graph. A sequence $[v_1, \ldots, v_k]$ of distinct vertices is a **path** in G if (v_1, v_2), $\ldots$, $(v_{k-1}, v_k) \in E$. These edges are called the edges of the path. The length of the path is the number $k - 1$ of its edges. A closed path $[v_1, \ldots, v_k, v_1]$ is called a **cycle** if in addition $(v_k, v_1) \in E$. A chord of a cycle $[v_1, \ldots, v_k, v_1]$ is an edge between two vertices of the cycle that is not an edge of the cycle. A cycle is **chordless** if it contains no chords. Trivially, a triangle has no chord, so we refer to a chordless cycle in this work as having length strictly greater than 3. We denote by C_k the chordless cycle on k vertices, and we always assume $k > 3$. An undirected graph G is a **chordal (triangulated) graph**, if every cycle in G of length strictly greater than 3 possesses a chord, i.e., there is no chordless cycle C_k in G. A graph $G = (V, E)$ is a **weakly chordal graph** if neither

G nor its complement $\overline{G}$ have an induced subgraph C_k, $k \geq 5$.

Part I

Chordal probe graphs

2 Introduction to chordal probe graphs

We introduce the class of chordal probe graphs which are a generalization of both interval probe graphs and chordal graphs. Our study of chordal probe graphs was originally motivated as a generalization of the interval probe graphs . However, chordal probe graphs also have their own computational biology application as a special case of constructing phylogenies, tree structures which model genetic mutations.

2.1 Motivation

The use of interval models in molecular biology dates back to the original studies on the linearity of genes by the well-known biologist Seymore Benzer. In [3], interval graphs were defined in order to study overlap data on subelements inside the gene. The question at that time was whether the overlap data was consistent with the hypothesis that genes are linear structures with the sub-elements being intervals on that line. If perfect and complete data were to be available for of all pairs of these subelements, then the problem would be that of recognizing an interval graph. However, with only partial data, as is always the case in experimental genetics, the problems involve embedding the data in a larger consistent set (e.g., interval graph completion). This will generally require inferring additional intersections (e.g., selectively adding chords to a cycle), and using additional biological information or assumptions to test consistency and propose the possible linear orderings.

Interval probe graphs, a generalization of interval graphs, were introduced by

Zhang [57] and used in [58] and [59] to model certain problems in physical mapping of DNA when only partial data is available on the overlap of clones (i.e., the intervals). Specifically, the clones are distinguished by the experimental scientist as being either probes or non-probes, where no intersection information is known between pairs of non-probes, but complete intersection information is known between all probe/probe and all probe/non-probe pairs. Zhang defined a graph to be interval probe if its vertex set can be partitioned into probes and non-probes in such a fashion that it can be completed into an interval graph by adding only edges between non-probes. This shows two different facets of the problem: either the partition is given in advance, or a partition has to be proposed as part of the solution.

The definitions of chordal probe and interval probe graph do not specify a particular partition of the vertices in advance. However, in biology applications, the partition into probes and nonprobes is usually part of the input. Therefore, we may distinguish between the general case of **non-partitioned probe** graphs, where we must find both a partition and an interval completion for it, and the special case of **partitioned probe** graphs, where we are given a fixed partition and must only find a completion for it.

Recently, for partitioned interval probe graphs, an $O(n^2)$ time recognition algorithm was first reported in [42] which uses PQtrees. Another method given in [45], uses modular decomposition and has complexity $O(n + m \log n)$ for a graph with n vertices and m edges. In contrast to this, however, recognizing interval probe graphs (when no partition is given) is an open problem. However, in the case of trees, Sheng [50] gives characterizations by a family of forbidden subgraphs for both the partitioned and the non-partitioned case, thus ensuring polynomial time recognition of trees which are interval probe graphs (see also [34]).

A graph G is chordal probe if its vertices can be partitioned into two sets P (probes) and N (non-probes) where N is a stable set and such that G can be extended to a chordal graph by adding edges between non-probes.

2.2 Outline of our results on chordal probe graphs

The initial results on chordal probe graphs are given in Sections 3 and 3.1. In Section 3.2, we give the complete hierarchy with separating examples for the classes of chordal probe , weakly chordal, interval probe and related families of graphs. In the remainder of the paper, we discuss the algorithmic aspects of recognizing those graphs which are chordal probe graphs and are also weakly chordal. As shown in Section 3.2, this class properly contains the class of interval probe graphs. In Section 4, we consider the case with respect to a given partition. The more challenging case without being given the partition is presented in Section 5.

We prove a number of basic properties of chordal probe graphs, and give a result on the enhancement of a graph which generalizes a previous result of Zhang [57] on interval probe graphs. We then study the chordal probe graphs which are also weakly chordal, a subfamily which includes both the interval probe graphs and the chordal graphs.

We present a complete hierarchy with separating examples for chordal probe and related classes of graphs. We give polynomial time recognition algorithms for the subfamily of chordal probe graphs which are also weakly chordal, first in the case of a fixed given partition of the vertices into probes and non-probes, and second in the more general case where no partition is given. This algorithm has time complexity $O(m^2)$ and recognizes even-chordal, chordal probe graphs, the subfamily of chordal probe graphs which have no even hole; this class includes the interval probe graphs and is also weakly chordal ([35]).

We then solve the general problem for chordal probe graphs, by giving polynomial time recognition algorithms for the partitioned as well as the non-partitioned case. In doing so, we introduce two new graph superclasses, the N-triangulatable graphs and cycle-bicolorable graphs, proving interesting properties for both of them.

In the partitioned case, we solve a broader problem in which the set N is not assumed to be a stable set, which defines the class of N-triangulatable graphs. We

investigate the structural properties of this class, and show that several properties of chordal graphs can be extended to this class; these results enable us to propose a recognition algorithm with a complexity of $O(|P|m)$.

In the non-partitioned case, we again solve a broader problem by introducing the class of cycle-bicolorable graphs, a superclass of chordal probe graphs. We characterize chordal probe graphs as cycle-bicolorable graphs in which one color defines a stable set, and give a corresponding $O(n^2m)$ time recognition algorithm. These results are based on a new graph decomposition, which groups together the cycles of the graph into so-called C-components. The polynomial time complexity relies on the theory of graph separators.

N-triangulatable graphs have properties similar to chordal graphs, and we characterize them using graph separators and using a vertex elimination ordering. Cycle-bicolorable graphs are shown to be perfect, and any cycle-bicoloring of a graph renders it N-triangulatable.

The corresponding recognition complexity for chordal probe graphs, given a partition of the vertices into probes and non-probes, is $O(|P||E|)$, thus also providing a interesting tractable subcase of the chordal graph sandwich problem. If no partition is given in advance, the complexity of our recognition algorithm is $O(|V|^2|E|)$.

2.3 Preliminaries

Definition 2.1 *An undirected graph $G=(V,E)$ is a **chordal probe graph** if its vertex set can be partitioned into two subsets, P (probes) and N (non-probes), where N is a stable set and there exists a completion $E' \subseteq \{(u,v) \mid u,v \in N, u \neq v\}$ such that $G' = (V, E \cup E')$ is a chordal graph.*

Example 2.2 *The graph C_4 is not chordal by definition. However, it is chordal probe as shown in Figure 1, since there exists a partition $P = \{c,d\}$, $N = \{a,b\}$ that can be completed into a chordal graph by adding the edge (a,b).*

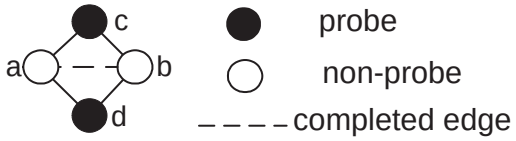

Figure 1: Chordal completion of the graph C_4

Definition 2.3 *The graph $G = (V, E)$ is a **bipartite graph** if there exists a partition of its vertex set into two stable sets, which is called a bipartition of the graph. The graph $G = (V, E)$ is a **split graph** if there exists a partition of its vertex set into a clique and a stable set. A split graph is characterized as being a graph which is chordal and whose complement is also chordal.*

Remark 2.4 *Bipartite graphs are chordal probe graphs.*

Proof: A chordal completion can be obtained by taking the same bipartite partition (X,Y) of the vertices into two stable sets, and making X probes and Y non-probes and filling in edges to make Y into a clique. This completion is a split graph and thus a chordal graph. □

Definition 2.5 *An undirected graph $G = (V, E)$ is an **interval graph** if its vertices can be put into one-to-one correspondence with a set of intervals $\Im = \{I_v\}_{v \in V}$ of a linearly ordered set (like the real line) such that two vertices are adjacent in G if and only if the corresponding intervals have a non-empty intersection that is, $(u, v) \in E \Leftrightarrow I_u \cap I_v \neq \emptyset$.*

Definition 2.6 *An undirected graph $G = (V, E)$ is a **interval probe graph** if its vertex set can be partitioned into two subsets, P (probes) and N (non-probes), where N is a stable set and there exists a completion $E' \subseteq \{(u, v) \mid u, v \in N,\ u \neq v\}$ such that $G' = (V, E \cup E')$ is an interval graph.*

Remark 2.7 *Interval probe graphs are chordal probe graphs.*

Proof: Any interval completion is also a chordal completion. The converse is not true, as follows. The even chordless cycles of length greater than 4 are chordal probe graphs, because they are bipartite. Since interval probe graphs are weakly chordal (see [34, 48, 47]), *the even length chordless cycles greater than 4 are chordal probe but not interval probe graphs.* □

The odd chordless cycles of length greater than 4, however, are not chordal probe graphs as we will show in Section 3.

Definition 2.8 *A graph $G = (V, E)$ is a **tolerance graph** if each vertex $v \in V$ can be assigned a closed interval I_v and a tolerance $t_v \in R^+$ so that $(x, y) \in E$ if and only if $|I_x \cap I_y| \geqslant min(t_x, t_y)$, see [33, 34].*

Remark 2.9 *Every interval probe graph is a tolerance graph, by assigning infinite tolerances to non-probes and very small tolerances to probes.*

Theorem 2.10 *([33]) Tolerance graphs are weakly chordal.*

The complexity of recognizing tolerance graphs is an open problem. In [26], we presented the hierarchy of tolerance, interval probe and interval graphs, and the restricted cases of having an interval representation where (i) intervals have unit length or (ii) no interval properly contains another interval, see also [34].

We often use the notation $P_{H[v,w]}$ to denote a chordless path $[v = v_1, \ldots, v_k = w]$ in the induced subgraph H, that is, $v_1, \ldots, v_k \in V(H)$ and $\{(v_1, v_2), \ldots, (v_{k-1}, v_k)\}$ $= E(H_{\{v_1,\ldots,v_k\}})$. If $C = [v_1, \ldots, v_k, v_1]$ is a chordless cycle in G, then we denote by $P_{C[v,w]}$ the path $[v = v_i, v_{i+1}, \ldots, w]$ in the cycle C starting with vertex $v \in C$ and ending with the vertex $w \in C$, following the same cyclic order.

Definition 2.11 *Three independent vertices of G form an **asteroidal triple** of G if for every pair of them there is a path connecting these two vertices that avoids the neighborhood of the remaining vertex.*

Theorem 2.12 *(Lekkerkerker and Boland [44]) A graph G is an interval graph if and only if it is chordal and does not contain an asteroidal triple.*

We say that we *saturate* a set of vertices if we add all the edges necessary to make it a clique. In this paper, a *connected component* is a vertex set which induces a maximal connected subgraph.

The (open) *neighborhood* of a vertex x in graph G is the set $\mathcal{N}_G(x) = \{y \neq x \mid xy \in E\}$; we will say that a vertex x *sees* another vertex y if $xy \in E$. The *closed neighborhood* of x is $\mathcal{N}_G[x] = \mathcal{N}_G(x) \cup \{x\}$. We extend the notion of neighborhood to a set of vertices A by defining $\mathcal{N}_G(A) = \cup_{x \in A} \mathcal{N}_G(x) - A$ and $\mathcal{N}_G[A] = \mathcal{N}_G(A) \cup \{A\}$. When there is no ambiguity as to which graph is referred to, the subscript will be omitted, i.e. $\mathcal{N}_G(x)$ will be written simply $\mathcal{N}(x)$. The degree of vertex x will be denoted by $deg(x) = |\mathcal{N}(x)|$.

A chordless cycle of length k is denoted by C_k, and we always assume $k \geq 4$. A *hole* is a chordless cycle of length at least five; a hole is called *odd* or *even* depending on the parity of its length. An *antihole* is the complement of a hole. A graph G is called *perfect* if every induced subgraph G_A satisfies the equality $\omega(G_A) = \chi(G_A)$, where ω denotes the size of the largest clique and χ is the chromatic number. The *Strong Perfect Graph Theorem* [15], originally conjectured by Berge, states that *a graph is perfect if and only if it contains neither an odd hole nor an odd antihole.*

Chordal graphs are a well known family of perfect graphs. A graph is defined to be *chordal* if every cycle of length 4 or greater has a chord, that is, an edge joining two nonconsecutive vertices of the cycle. Chordal graphs have important application areas including acyclic relational database schemes, facility location problems, statistical analysis, and the problem of tracing genetic mutations over an evolutionary period by constructing phylogenetic trees (see [22], [48]).

It is often the case in such applications, however, that an input graph G has edges missing due to incomplete data. This gives rise to the problem of adding additional edges F in order to complete it into a chordal supergraph $G' = (V, E + F)$ of G. The edge set F is said to be *minimal* if no proper subset defines a chordal graph when

added; the resulting chordal graph is then called a *minimal triangulation*. When $|F|$ is required to be smallest possible, it is called a *minimum triangulation*.

The *chordal graph sandwich problem* (see [11], [25], [52]) is another variation where a specified set of optional edges E_0 is given with the input, and the triangulation F (not necessarily minimum) must satisfy $F \subset E_0$. Both the minimum triangulation problem and the chordal graph sandwich problem are NP-complete.

The *chordal probe graphs* may also be defined as: An undirected graph $G = (V, E)$ is a *chordal probe graph* if its vertex set can be partitioned into two subsets, P (probes) and N (non-probes), where N is a stable set and there exists a completion $F \subseteq N^2$ such that $H = (V, E + F)$ is a chordal graph. The class of chordal probe graphs was introduced in [28] as a generalization of interval probe graphs. Interval probe graphs are defined similarly, where the completed graph H must be an interval graph.
A vertex is *simplicial* if its neighborhood is a clique. The notion of *simplicial vertex* was introduced independently by Dirac in 1961 [16] and by Lekkerkerker and Boland in 1962 [44] as an extension of the notion of a leaf in a tree, and is the basis for the following theorem by Dirac:

Theorem 2.13 *([16]) A non-clique chordal graph has at least two non-adjacent simplicial vertices.*

This led Fulkerson and Gross [19] to characterize chordal graphs in an algorithmic manner as follows:

Characterization 2.14 *([19]) A graph is chordal if and only if one can repeatedly find a simplicial vertex and delete it from the graph, until no vertex is left.*

This defines an ordering on the vertices called a *perfect elimination ordering (peo)*.

One of the earliest ways which was used to compute a triangulation was to force the graph into respecting this characterization, by using an ordering α on the vertices and repeatedly choosing the next vertex in this ordering, forcing its neighborhood into a

clique by the addition of any missing edges, and removing the vertex; we refer to this process as the *elimination game on* (G, α). Each of the successive graphs obtained is called a *transitory elimination graph* and is denoted by G_i. At the end of the process, the set F of added edges define a triangulation $G^+_\alpha = (V, E + F)$ of the input graph (V, E) (see [10]).

The following property is well-known:

Property 2.15 *If G' is a triangulation of G and α is a peo of G', then $G' = G^+_\alpha$.*

2.4 Minimal separators and minimal triangulations

Minimal separators were introduced by Dirac [16]. A subset S of vertices of a connected graph G is called a *separator* if G_{V-S} is not connected. A separator S is called an *ab-separator* if a and b are in different connected components of G_{V-S}, a *minimal ab-separator* if S is an *ab*-separator and no proper subset of S is an *ab*-separator; finally, a separator S is a *minimal separator* if there is some pair $\{a, b\}$ such that S is a minimal *ab*-separator. Equivalently, a separator S is minimal if there exist two distinct components C_1 and C_2 in G_{V-S} such that $\mathcal{N}(C_1) = \mathcal{N}(C_2) = S$; such components are called *full components of S*.
Minimal separators turn out to be a very useful tool for computing a minimal triangulation.

Definition 2.16 (Kloks, Kratsch, and Spinrad [43]) *Let S and T be two minimal separators of G. Then S* crosses *T if there exist two components X_1, X_2 of $G_{(V-T)}$, $X_1 \neq X_2$, such that $S \cap X_1 \neq \emptyset$ and $S \cap X_2 \neq \emptyset$.*

It is shown in [49] that the crossing relation is symmetric, i.e. *S crosses T* if and only if *T crosses S*.

Property 2.17 *Let S and T be two minimal separators of G. Then S* crosses *T if and only if T has a vertex in each full component of S.*

Theorem 2.18 *([49]) When a minimal separator S is saturated, creating graph G':*

1. *All the minimal separators which cross S disappear.*
2. *All the minimal separators which do not cross S remain.*
3. *No new minimal separator appears.*
4. *Any minimal triangulation of G' is a minimal triangulation of G.*

Thus, computing a minimal triangulation of a graph G is equivalent to saturating a maximal set of pairwise non-crossing minimal separators of G (see [49]).

The following is a consequence of Theorem 2.18:

Property 2.19 *S and T are two crossing minimal separators of a graph G if and only if S contains two non-adjacent vertices x and y such that T is a minimal xy-separator of G.*

Lekkerkerker and Boland in [44] introduced the following notion which will be fundamental to this paper.

Definition 2.20 *A* substar *S of x is a subset of $\mathcal{N}(x)$ such that for some connected component U of $G_{V-\mathcal{N}[x]}$, $S = \mathcal{N}(U)$,* i.e., all the vertices of a substar see some common connected component of $G_{V-\mathcal{N}[x]}$.

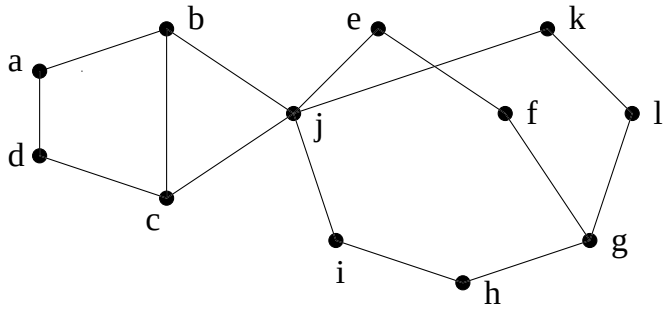

Figure 2: Illustration of Example 2.21

Note that the substars of x are exactly the minimal separators included in the neighborhood of x.

Example 2.21 *In Figure 2, the substars of j are $\{b, c\}$ and $\{e, i, k\}$; j is on a chordless cycle with e, i and k, because substar $\{e, i, k\}$ is an independent set, but j is* not *on a chordless cycle with b or c, because substar $\{b, c\}$ is a clique.*

Additional properties which are useful are the following.

Property 2.22 *([6]) For a vertex x, the substars of x are pairwise non-crossing.*

Property 2.23 *([6]) Let x, y be two non-adjacent vertices of a graph G; then no substar of x can cross a substar of y.*

Property 2.24 *([44]) A vertex x is on a chordless cycle if and only if at least one of its substars is not a clique. More precisely, if X is a connected component of $G_{V-N[x]}$ such that $S = N(X)$ contains the non-edge yz, then x is on a chordless cycle C on which it sees y and z, and all the other vertices of C are in X.*

For example, in Figure 2, vertex j is on a chordless cycle with e, i and k, because substar $\{e, i, k\}$ is an independent set, but j is *not* on a chordless cycle with b or c, because substar $\{b, c\}$ is a clique.

This leads us to the following definition.

Definition 2.25 *We say that a vertex x is* LB-simplicial *if all the substars of x are cliques.*

Finally, we recall the following characterization of chordal graphs which appears not to be well known.

Characterization 2.26 *([44]) A graph is chordal if and only if every vertex is LB-simplicial.*

3 First Results

Lemma 3.1 *If $G = (V, E)$ is a chordal probe graph with respect to a given partition of the vertex set of G into P (probes) and N (non-probes), where N is a stable set, then probes and non-probes alternate in every chordless cycle in G.*

Proof: Suppose there is a chordless cycle $C = [x_1, \ldots, x_k, x_1]$ in G ($k \geq 4$), such that probes and non-probes don't alternate in C. There is no pair of consecutive non-probes in C, because N is a stable set. Therefore, (by renumbering if necessary) let $\{x_1, x_2\}$ be a pair of adjacent probes in C. Let G' be any chordal completion graph of G. The probes x_1 and x_2 remain with the same neighborhood in G', because edges are added only between non-probes. Let (x_i, x_j) be a completed edge in G', such that $i > 2$ is smallest possible and j is the largest possible among the neighbors x_j of x_i in the completion. Assign $C' = [x_1, x_2, V(P_{C(x_2,x_i)}), x_i, x_j, V(P_{C(x_j,x_1)}), x_1]$. Now C' is a chordless cycle of length greater than 3 in G', because the cycle contains at least x_1, x_2, x_i and x_j. This contradicts the chordality of G'. Therefore, the given partition (P, N) of vertex set of G does not have a chordal completion, contradicting the assumption. Hence, probes and non-probes alternate along the cycle C. □

Theorem 3.2 *If $G = (V, E)$ is a chordal probe graph, then G has no induced subgraph C_{2k+1}, for $k \geq 2$, i.e., G has no odd chordless cycle of length greater than 4.*

Proof: Suppose there exists an odd chordless cycle C of length greater than 4 in G. For any partition of the vertex set of G into P and N, probes and non-probes can not alternate in C. This is a contradiction to Lemma 3.1. □

Theorem 3.3 *Let G be a chordal probe graph, then G has no induced subgraph $\overline{C_k}$, for $k \geq 5$, i.e., G has no complement of a chordless cycle of length greater than 4.*

Proof: Let $G = (V, E)$ be a chordal probe graph. By Theorem 3.2, G has no induced $\overline{C_5}$, since $\overline{C_5}$ is isomorphic to C_5. Suppose there exists $k \geq 6$, such that $\overline{C_k}$

is an induced subgraph of G. Let $x_1, \ldots, x_k$ be the vertices of $\overline{C_k}$ ordered according to the ordering of vertices in C_k. Observe that $C' = [x_2, x_4, x_1, x_5, x_2]$ is a chordless cycle of length 4 in G. In any partition of V into probes (P) and non-probes (N), which has a chordal completion, either x_1 and x_2 are non-probes or x_4 and x_5 are non-probes. Without loss of generality, assume that x_1 and x_2 are non-probes. The vertex x_1 is adjacent to all the vertices in $\overline{C_k}$, except for x_2 and x_k. Therefore, all the vertices in $\overline{C_k}$, except possibly x_2 and x_k are probes. But $C'' = [x_3, x_5, x_2, x_6]$ is also a chordless cycle of length 4 in G. In any partition (P, N) of V, which has a chordal completion, either x_2 and x_3 are non-probes or x_5 and x_6 are non-probes. Contradiction. □

Definition 3.4 *A graph G is called **perfect** if every induced subgraph G_X satisfies the equality $\omega(G_X) = \chi(G_X)$, where ω denotes the size of the largest clique and χ is the chromatic number.*

Theorem 3.5 ***(Strong Perfect Graph Theorem)*** *[15] A graph G is perfect if and only if it contains neither a chordless odd cycle of length greater than 4 nor its complement.*

Therefore, Theorems 3.2 and 3.3 prove the following.

Corollary 3.6 *Chordal probe graphs are perfect.*

3.1 Even-Chordal Graphs

Definition 3.7 *A graph $G = (V, E)$ is an **even-chordal graph** if it has no induced chordless cycle of even length strictly greater than 4. (This is what would be called (1,6)-even-chordal in [13]).*

Corollary 3.8 *If G is a chordal probe graph, then G is even-chordal if and only if G is weakly chordal.*

Proof: ($\Leftarrow$) This direction is immediate, since weakly chordal graphs are even-chordal.

($\Rightarrow$) Suppose G is an even-chordal graph. By Theorem 3.2, G contains no odd chordless cycle of length ≥ 5, so G contains no C_k, $k \geq 5$. In addition, by Theorem 3.3, the complement of G contains no chordless cycle of length ≥ 5. Therefore, G is a weakly chordal graph. $\Box$

By Remark 2.9, the interval probe graphs are contained in the class of tolerance graphs and by Remark 2.10, tolerance graphs are weakly chordal. Therefore, we remark the following:

Remark 3.9 *If a chordal probe graph G is not weakly chordal, then G is not an interval probe graph.*

Definition 3.10 *Let G be a graph, whose vertices are partitioned into P (probes) and N (non-probes). Let C=$[a, c, b, d, a]$ be a chordless cycle in G, such that probes and non-probes alternate in C, as shown in Figure 1. Any chordal completion of G=$(P \cup N, E)$ must have the edge (a, b), which we call an **enhanced** edge and the probes $\{c, d\}$ are called the **creator pair** of the enhanced edge (a, b). The **enhanced graph** G^*=$(P \cup N, E^*)$ is the graph obtained from G by adding all enhanced edges.*

The notion of the enhanced graph was first introduced for interval probe graphs in [57], (see [47]). We apply it more generally.

Definition 3.11 *Let $G = (V, E)$ be a graph. We call a partition of its vertex set into two subsets P (probes) and N (non-probes) **valid** if N is a stable set and probes and non-probes alternate in every chordless cycle in G.*

We now present the main result of this section.

Theorem 3.12 *Let $G = (V, E)$ be an even-chordal graph. If there exists a valid partition of V, then G is a chordal probe graph and the enhanced graph G^* is a chordal completion.*

Proof: Let $G = (P, N, E)$ be an even-chordal graph with a given valid partition of V into P and N. By the definition of valid partition, there are no chordless cycles of odd length in G, hence all chordless cycles in G are 4-cycles.

We will show by contradiction that G^* has no induced chordless cycle. Suppose C is a chordless cycle in G^*, with the smallest possible number t of enhanced edges.

If C has no enhanced edge ($t = 0$), then C is a 4-cycle in G. Since probes and non-probes alternate in C, it must have enhanced edge. Contradiction.

Thus, $t \geq 1$ and there is no chordless cycle in G^* with less than t enhanced edges. Let $\{(a_1, b_1), \ldots, (a_t, b_t)\}$ be the enhanced edges in C.

We prove the two following claims needed to complete the proof.

Claim 3.13 *Let $\{c, d\}$ be a creator pair of an enhanced edge (a, b) in C. At most one of c or d may have a neighbor $x \in C$, such that $x \neq a$, $x \neq b$.*

Proof of Claim 3.13: Suppose both c and d have a common neighbor $x \in C$, such that $x \neq a$ and $x \neq b$, as shown in Figure 3. The vertex x cannot be adjacent to both a and b, because C is a chordless cycle with at least 4 vertices. If $(a, x) \in E^*$, then let $C' = [c, b, d, x, c]$, else let $C' = [c, a, d, x, c]$. In either case, C' is a chordless cycle of length 4 in G^* and also in G. Therefore, probes and non-probes alternate in C', and consequently it must have an enhanced edge. Contradiction!

Suppose c and d have no common neighbor, then let $Q = [a = x_1, \ldots, x_t = b]$ be the longer path from a to b on the cycle C. Let x_i be a first vertex on Q, which is a neighbor of either c or d, and let x_j be the next on Q which is a neighbor of the other of d or c, as shown in the Figure 4. Then $C' = [a, \ldots, x_i, \ldots, x_j, d, a]$ is a chordless cycle in G^*, with less than t enhanced edges. Contradiction. This proves Claim 3.13.

□

Claim 3.14 *No creator pair creates more than one enhanced edge in C. In other words there are no two equal creator pairs $\{c_i, d_i\} = \{c_j, d_j\}$, $i \neq j$, which create (a_i, b_i) and (a_j, b_j) respectively.*

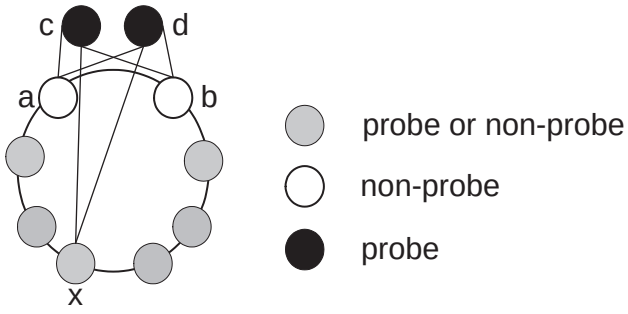

Figure 3: Illustration of the proof of Claim 3.13

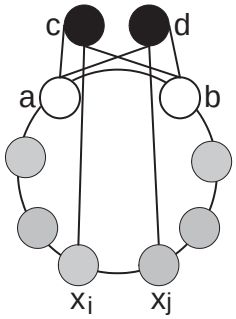

Figure 4: Illustration of the proof of Claim 3.13

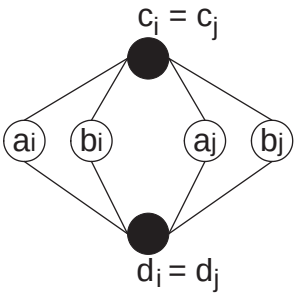

Figure 5: Illustration of the proof of Claim 3.14

Proof of Claim 3.14: Suppose that there exist creators $\{c_i, d_i\} = \{c_j, d_j\}$, $i \neq j$, which create (a_i, b_i) and (a_j, b_j) respectively, as shown in Figure 5, while possibly $a_i = a_j$ or $b_i = b_j$. Then the pair $\{c_i, d_i\}$ also creates the enhanced edges (a_i, b_j), (b_i, a_j), and (b_i, b_j) if $b_i \neq b_j$ and (a_i, a_j) if $a_i \neq a_j$. This means that the vertices a_i, a_j, b_i, b_j form a clique in G^*, which contradicts the assumption that C is a chordless cycle in G^*. This proves Claim 3.14. □

We now complete the proof of Theorem 3.12. By Claim 3.13, there exists a creator d_i for every enhanced edge (a_i, b_i), such that d_i is adjacent only to vertices a_i and b_i in C. All the elements of $X = \{d_1, \ldots, d_t\}$ are different because their neighborhoods are different.

If X is a stable set of G, then by replacing each enhanced edge (a_i, b_i) by the two edges $(a_i, d_i), (d_i, b_i) \in E(G)$, we would obtain a chordless cycle C' of G of length greater than 4. Contradiction!

Otherwise, (by renumbering if necessary) there exist probes d_1 and $d_i, i > 1$, such that $(d_1, d_i) \in E(G)$. If $a_1 = b_i$, then $b_1 \neq a_i$ due to Claim 3.14. If $a_1 = b_i$, then assign $C' = [b_1, d_1, d_i, a_i, V(P_{C[a_i,b_1]}), b_1]$, else assign $C' = [a_1, d_1, d_i, b_i, V(P_{C[b_i,a_1]}), a_1]$. The cycle C' is a chordless cycle in G^* with fewer than t enhanced edges. Contradiction! Thus, we have shown that the enhanced graph G^* of G, with respect to the given valid partition, is a chordal graph. Therefore, it is a chordal completion and G is a chordal probe graph. □

Corollary 3.15 *Let $G = (P, N, E)$ be a chordal probe graph with respect to a given partition of $V(G)$ into P and N, where N is a stable set. If G is an even-chordal graph, then the enhanced graph G^* is chordal.*

Proof: According to Lemma 3.1, probes and non-probes alternate in every cycle greater than 3 in G. Since G is an even-chordal graph, then according to Theorem 3.12 the enhanced graph G^* is chordal. □

By Remark 2.7, every interval probe graph is a chordal probe graph with respect to the same partition and is also an even-chordal graph. Therefore, we obtain an alternate proof of the following:

Corollary 3.16 (Zhang [57]) *Let $G = (V, E)$ be an interval probe graph with respect to the partition of its vertex set into P and N, where N is a stable set. The enhanced graph G^* is chordal.*

3.2 Hierarchy of chordal probe graphs

In this section, we present the hierarchical relationships of chordal probe graphs and other well-studied families of graphs, as summarized in Figure 6.

We say that a hierarchy is **complete**, when all containment relationships are given. That is, (1) a downward edge from class A to class B indicates that class A contains class B, (2) the lack of a hierarchical (containment) relation indicates that the classes are incomparable, (3) classes that appear in the same box are equivalent, (4) an example appearing along the edge between two classes is a separating example for those classes.

Theorem 3.17 *The hierarchy and separating examples in Figure 6 are correct. Moreover, the hierarchy is complete.*

Proof: The hierarchical containment of interval $\subset$ interval probe $\subset$ tolerance $\subset$ weakly chordal is presented in [26] (see also [34]). The containment of weakly chordal graphs in the family of even-chordal graphs and the containment of chordal $\subset$ (chordal probe $\cap$ even-chordal) $\subset$ chordal probe are immediate from the definitions. Interval graphs are contained in the family of chordal graphs as stated in [44], and according to Remark 2.4, bipartite graphs are contained in the family of chordal probe graphs. The family of weakly chordal graphs contains those chordal probe graphs which are also even-chordal according to Corollary 3.8. Interval probe graphs are even-chordal and are contained in the family of chordal probe graphs due to Remark 2.7.

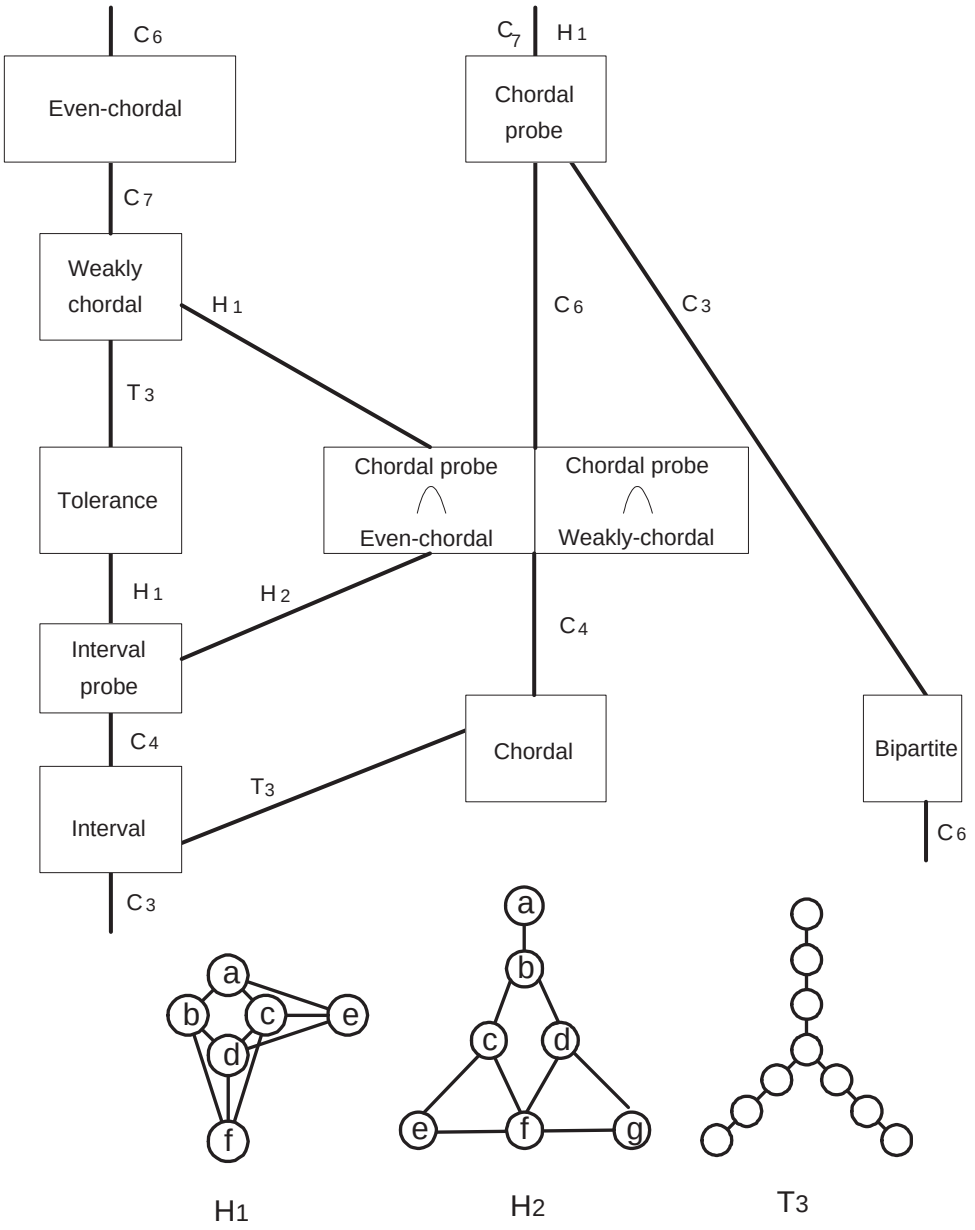

Figure 6: The complete hierarchy between chordal probe graphs and other well-studied families of graphs.

We now prove that the examples shown along edges of the diagram are the separating examples.

The graph H_1: In order to complete the graph H_1 to chordal, one must add a chord to the chordless cycle $[a, b, d, c, a]$. Suppose the chord is (b, c), then $\{a, d, e\}$ are probes, in which case a chordless cycle $[a, b, d, e, a]$ cannot be completed. Suppose the chord is (a, d) and so $\{b, c, f\}$ are probes, then a chordless cycle $[a, c, f, b, a]$ cannot be completed. Thus, any partition of H_1 has no chordal completion and therefore H_1 is not a chordal probe graph. However, the graph H_1 is a tolerance graph, since it has a tolerance representation as shown in Figure 8.

The graph H_2: To complete this graph into a chordal graph, one must add a chord to the chordless cycle $[b, c, f, d, b]$. Suppose the chord is (b, f), then $\{a,c,d,e,g\}$ are probes, in which case no other completed edges can be added and therefore $\{a,e,g\}$ compose an asteroidal triple. Suppose the chord is (c, d) and so $\{b,e,f,g\}$ are probes and $\{a\}$ is either a probe or a non-probe, in which case only completed edges (a, c), (a, d) can be added and therefore again $\{a,e,g\}$ is an asteroidal triple. Hence, any chordal completion of the graph H_2 has an asteroidal triple and therefore H_2 is not an interval probe graph. However, the graph H_2 is even-chordal and also is a chordal probe graph, since it has a valid partition and a completion of that partition as shown in Figure 7.

The graph T_3: This tree is not a tolerance graph, as proved in [33]. However, it is a chordal graph and therefore is a chordal probe graph and also an even-chordal graph.

The graph C_3: The triangle is obviously an interval graph, thus it is chordal probe , but it is not bipartite.

The graph C_4: The 4-cycle is not chordal by definition and therefore is not an interval graph. It is interval probe, since it can be completed into an interval graph as in Figure 1.

The graph C_6: The 6-cycle is not an even-chordal graph by definition. It is chordal probe , since by partitioning the vertices such that probes and non-probes alternate in the cycle, it can be completed into a chordal graph by connecting the non-probes

Table 1: Separating examples between chordal probe graphs and related incomparable classes.

A	B	$G_1 \in A - B$	$G_2 \in B - A$
chordal probe	even-chordal/weakly chordal /tolerance	C_6	H_1
bipartite	even-chordal/weakly chordal /tolerance/interval probe/interval	C_6	C_3
tolerance	chordal probe $\cap$ even-chordal /chordal	H_1	T_3

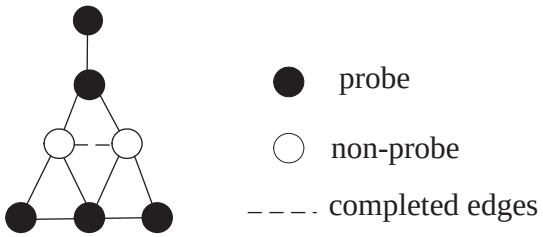

Figure 7: Partition of the graph H_2 and its chordal completion.

into a clique. Finally, every chordless cycle of even length is bipartite.

The graph C_7: The 7-cycle is even-chordal, but it is not a weakly chordal graph by definition. It is not chordal probe by Theorem 3.2.

Finally, we verify the incomparabilities between pairs of classes. We show that classes A and B are incomparable by exhibiting graphs $G_1 \in A - B$ and $G_2 \in B - A$ in Table 1. □

---

4 Partitioned Chordal Probe Graphs

In Section 4.1, we study the chordal probe graphs which are also even-chordal, a subfamily which includes both the interval probe graphs and the chordal graphs. We give a recognition algorithm of even-chordal partitioned chordal probe graphs

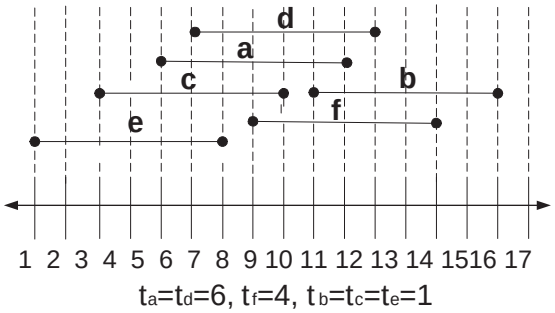

Figure 8: Tolerance representation of the graph H_1.

with time complexity $O(m^2)$. We investigate N-triangulatable graphs, which are a superclass of chordal probe graphs in Section 4.2. The properties of N-triangulatable graphs have been used in order to recognize the partitioned chordal probe graphs in Section 4.6 and give recognition algorithm with time complexity $O(nm)$.

4.1 Recognizing even-chordal partitioned chordal probe graphs

According to Corollary 3.8, a chordal probe graph is weakly chordal if and only if it is even-chordal. Therefore, by Remark 3.9, if a chordal probe graph G is not even-chordal, then G is not an interval probe graph. The recognition of weakly chordal graphs can be done in $O(m^2)$, see [5] or [37]. We, therefore, use this method in our algorithm. The algorithm may fail at the first stage, meaning that the graph is not weakly chordal, or it may fail at the second stage, in which case the graph is weakly chordal, but is not chordal probe. The overall time complexity of the algorithm is $O(m^2)$ by the following remark.

Remark 4.1 *The number of 4-cycles in a graph having m edges is at most $O(m^2)$ and generating them can be done in $O(m^2)$ time.*

Algorithm I: Recognition of even-chordal partitioned chordal probe graphs.

input : A graph $G = (V, E)$ with V partitioned into P and N, where N is a stable set.

output:

1 Verify that the graph G is weakly chordal using the algorithm in [5] or [37];

2 **foreach** *4-cycle* $C \in G$ **do**

  Verify that probes and non-probes alternate in C;

Theorem 4.2 *Let $G = (V, E)$ be a graph with vertex set partitioned into P and N, where N is a stable set. The graph G is a chordal probe graph with respect to (P, N) and an even-chordal graph iff Algorithm I recognizes it.*

Proof: ($\Leftarrow$) Suppose the graph G is recognized by the algorithm. Then the graph is weakly chordal (Step 1) and hence even-chordal by definition. Since probes and non-probes alternate in every chordless cycle of G (Step 2), (P, N) is a valid partition and the graph is chordal probe due to Theorem 3.12.

($\Rightarrow$) Let G be a chordal probe graph, which is also even-chordal. According to Corollary 3.8, G is a weakly chordal graph. In addition, probes and non-probes alternate in every chordless cycle in G due to Lemma 3.1. $\square$

4.2 N-triangulatable partitioned graphs

In order to solve recognition problem of partitioned chordal probe graphs, we first solve a more general new problem.

We introduce a new problem, namely, triangulating a graph whose vertex set is bipartitioned into 'probes' and 'non-probes' by adding only edges between non-probes. The corresponding class, which we call N-triangulatable graphs, is studied in this section.

One of the interesting developments is that N-triangulatable graphs turn out to be very similar to chordal graphs: we will show that several properties and characterizations of chordal graphs can be very profitably extended to N-triangulatable graphs, and that they yield the tools we need to handle this class efficiently.

Definition 4.3 *We will say that a graph $G = (P + N, E)$ is* N-triangulatable (N-T) *if a triangulation of G can be obtained by adding only edges whose endpoints are non-probes. We will call such a triangulation an* N-triangulation *of G.*

Remarks 4.4

1. *If G is N-T, then G_P is a chordal graph.*

2. *An induced subgraph of an N-T graph is an N-T graph.*

3. *In the case where $P = \emptyset$, the graph becomes an arbitrary graph, and it is always N-T.*

4. *In the case where N is a stable set, G is N-T if and only if G is chordal probe with respect to this partition.*

5. *Recognizing N-T graphs is a special case of the chordal graph sandwich problem, where the optional edges E_0 consist of all non-edges between non-probes.*

We will now see that both Lekkerkerker and Boland's Characterization 2.26 and Fulkerson and Gross' Characterization 2.14 can be extended to recognize this class. These will be studied respectively in Section 4.3 and Section 4.5.

4.3 Quasi LB-simpliciality of N-T graphs

In this section, we extend Characterization 2.26 of chordal graphs due to Lekkerkerker and Boland to N-T graphs. This will also enable us to give a recognition algorithm for N-T graphs using separators.

Definition 4.5 *We will say that a vertex x is* quasi-LB-simplicial *if all the non-edges of all the substars of x have both endpoints that are non-probes.*

Example 4.6 *In Figure 9, if black vertices are probes and white are non-probes, c is quasi-LB-simplicial, as its substars are $\{b, d\}$ and $\{j\}$.*

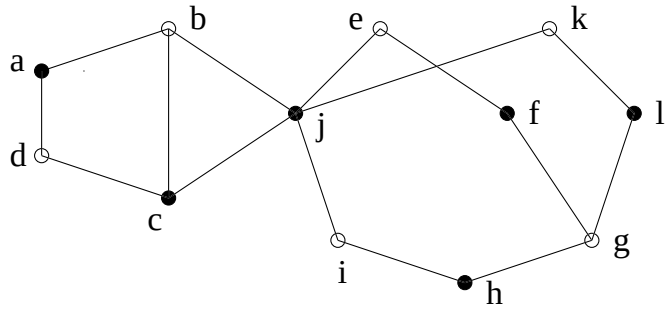

Figure 9: Illustration of example 4.6

We will see that examining the substars of the probes of the graph is sufficient to characterize N-T graphs.

Definition 4.7 *We will say that the substars of a probe are* P-substars.

Theorem 4.8 *The following conditions are equivalent for a graph $G = (P + N, E)$:*

1. *G is an N-T graph.*
2. *All probes of G are quasi-LB-simplicial.*
3. *G contains no chordless cycle with two adjacent probes.*

Proof:

(1) $\Rightarrow$ (3): Let $G = (P + N, E)$ be an N-T graph, let $V = P + N$, and let G' be an N-triangulation of G. Suppose by contradiction that in G there is a chordless cycle $(p_1, p_2, v_3, \ldots, v_k, p_1)$, where p_1 and p_2 are probes. In G', p_1 sees v_k and p_2; $v_3, v_4, \ldots, v_{k-1}$ belong to the same connected component X of $G'_{V-\mathcal{N}[p_1]}$. $\mathcal{N}(X)$ is a

substar of G', but it fails to be a clique, as it contains v_k and p_2, which are non-adjacent. This contradicts Characterization 2.26 for chordal graphs.

(3) $\Rightarrow$ (2): Assume in $G = (P + N, E)$ there is no chordless cycle with two adjacent probes. Suppose by contradiction that there exists a probe x which fails to be quasi-LB-simplicial: x has two non-adjacent neighbors, y and z, one of which is a probe; w.l.o.g. y is a probe. According to Property 2.24, there exists a chordless cycle which contains y, x and z consecutively, a contradiction.

(2) $\Rightarrow$ (1): Let $G = (P+N, E)$ be a graph such that all probes are quasi-LB-simplicial; we will prove that G is an N-T graph.

Let us use the minimal triangulation algorithm LB-TRIANG described in [6] which repeatedly chooses a vertex x, saturates its substars, and removes x. Regardless of the order in which the vertices are processed, LB-TRIANG computes a minimal triangulation of the input graph; we will run it by first choosing all the probes.

We claim that no new P-substar can appear. Because of Theorem 2.18, the only way a P-substar can be created is by adding edges which will cause a previous minimal separator S, which was not a P-substar, to be in the neighborhood of a probe. However, no new edge can be added incident to a probe, so this cannot happen.

After all the P-substars have been chosen and eliminated, only vertices from N are left in the graph. When we finish the execution, we will have computed a minimal triangulation of G which has added only edges between two non-probes, which is thus an N-triangulation of G. By definition, G is an N-T graph. $\square$

Complexity: The recognition algorithm based on Theorem 4.8 runs in $O(|P|m)$ time: the implementation of Algorithm LB-TRIANG proposed in [6] as in the proof of Theorem 4.8 uses a data structure, inspired from clique trees, and requires only $O(m)$ time per processed vertex; a global $O(m)$ time is then used to check that only edges between pairs of non-probes have been added.

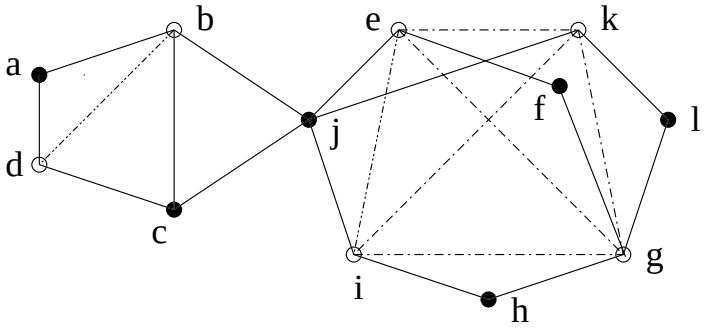

Figure 10: Illustration of example 4.12

To recognize N-T graphs, it is not necessary to actually compute an N-triangulation. Unless P is of order n, it is cheaper to recognize the class than to exhibit an N-triangulation for it, since computing a minimal triangulation requires $O(nm)$ time.

4.4 Properties of N-T graphs

Theorem 4.9 *Let $G = (P + N, E)$ be an N-T graph. The P-substars of G are pairwise non-crossing.*

Proof: Let $G = (P + N, E)$ be an N-T graph, and let $V = P + N$. Let us assume by contradiction that there are two crossing P-substars S_1 and S_2. By Property 2.23, S_1 and S_2 must be substars of two adjacent vertices p_1 and p_2. By Property 2.19, there must be two non-adjacent vertices x and y in S_1, such that S_2 is a minimal xy-separator.

Let us first suppose that p_2 belongs to S_1. Since p_1 is quasi-LB-simplicial, p_2 must see x and y. Therefore, every minimal xy-separator must contain p_2, which contradicts the fact that S_2 is a minimal xy-separator.

Let us now examine the case where p_2 is not in S_1. Let X be the connected component of $G_{V-\mathcal{N}[p_1]}$ such that $S_1 = \mathcal{N}(X)$. According to Property 2.24, x and y belong to some chordless cycle C on which x, p_1 and y are consecutive, with all other vertices in X. Suppose p_2 sees some of these intermediate vertices $C \cap (V - \mathcal{N}[p_1])$.

Then p_2 would belong to $\mathcal{N}(X)$, and thus to S_1, which is impossible. Therefore, S_2 has no vertex in X, which is a full component of S_1; by Property 2.17, S_1 and S_2 are non-crossing. □

Corollary 4.10 *The number of P-substars in an N-T graph is less than n.*

Proof: This follows from the simple observation that, since the P-substars are non-crossing minimal separators, they can all be chosen to be saturated and preserved in some minimal triangulation of G, which, as all chordal graphs, has less than n minimal separators. □

Recall that in the proof of Theorem 4.8, we ran LB-TRIANG by using all the probes in a first phase, and then the non-probes in a second phase. Since the minimal separators which are chosen as substars in the first phase are pairwise non-crossing, the resulting set F_P of edges which is added is the same, regardless of the order in which the probes are processed; the edges of F_P are mandatory, and we will use them to define G^* below. The set of edges computed by the second phase, however, depends on the order in which the non-probes are processed.

Definition 4.11 *We define the* enhanced graph G^* *of G to be the graph obtained from G by saturating all the P-substars of G.*

Example 4.12 *Figure 10 gives the enhanced graph of the graph of Figure 9.*

Theorem 4.13 *Any minimal triangulation of G^* is a minimal triangulation of G and an N-triangulation of G.*

Proof: By Theorem 4.9, the P-substars of G are pairwise non-crossing. Therefore, G^* is obtained by saturating a set of pairwise non-crossing minimal separators of G. By Theorem 2.18, any minimal triangulation of G^* is a minimal triangulation of G. If we run Algorithm LB-TRIANG as in the proof of Theorem 4.8 by first choosing the

probes, making these simplicial will only add edges between two non-probes; after the probes are processed and eliminated, only non-probes are left in the graph, and the chosen subsequent triangulation will also add only edges between two non-probes; the minimal triangulation thus obtained is an N-triangulation. $\square$

4.5 Quasi-perfect elimination in N-T graphs

We will now go on to show that Fulkerson and Gross' Characterization 2.14 can also be extended to an N-T graph and that, as is the case with chordal graphs with respect to perfect elimination orderings (peos), a greedy approach to playing the quasi-peo elimination game will successfully recognize N-T graphs.

Definition 4.14 *Let $G = (P + N, E)$ be an N-T graph. We will say that a vertex v of G is* quasi-simplicial *if every non-edge of $\mathcal{N}(v)$ has both endpoints which are non-probes.*

Definition 4.15 *We will say that an ordering α on the vertices of G is a* quasi-perfect elimination ordering (qpeo) *if at each step i of the elimination game on (G, α), vertex $\alpha(i)$ is quasi-simplicial in the transitory elimination graph G_i.*

Example 4.16 *In Figure 9, if black vertices are probes and white are non-probes, a is quasi-simplicial and d is not; however, if a is chosen first in a qpeo, saturating $\mathcal{N}(a)$ and removing a will make d quasi-simplicial in the transitory graph; $\alpha = (a, d, c, b, j, h, l, f, e, k, g, i)$ is a qpeo.*

Lemma 4.17 *Let $G = (P + N, E)$ be an N-T graph, and let v be a quasi-simplicial vertex of G. If G' is the graph obtained by making v simplicial and removing it, then G' is also N-T.*

Proof: Suppose by contradiction that G' fails to be an N-T graph, we will prove that G is not N-T. According to Theorem 4.8, there must be a chordless cycle C' in G' containing two consecutive probes. Let X' be the vertex set corresponding to C.

If $G_{X'}$ is also a chordless cycle (in G), then G fails to be N-T, by Theorem 4.8.

Otherwise, in cycle $G'_{X'}$ there exists a unique edge xy which was added while making v simplicial. (If several edges were added, C' would not be chordless). Let $X = X' \cup \{v\}$. Clearly, G_X is a cycle, call it C; suppose it fails to be chordless: v has a neighbor w on C, $w \neq x, y$; but in that case, edges xw and yw would have been added to G', which contradicts the fact that C' is chordless in G. Thus C is a chordless cycle with two consecutive probes, so by Theorem 4.8, G is not N-T. □

Theorem 4.18 *Let $G = (P + N, E)$ be a graph. The following are equivalent:*

1. *G is an N-T graph.*
2. *G has a quasi-perfect elimination ordering.*
3. *A greedy elimination game on quasi-simplicial vertices succeeds.*

Proof:

(2) ⇒ (1): Let $G = (P + N, E)$ be a graph with a qpeo α. Running the elimination game on (G, α) will add only edges between two non-probes, so it will produce an N-triangulation of G.

(1) ⇒ (2) Let $G = (P + N, E)$ be an N-T graph, let G' be an N-triangulation of G, and let $\beta = (v_1, \cdots, v_n)$ be a peo of G'. We claim that β is a qpeo of G. By Property 2.15, $G' = G^+_\beta$. Since v_i is simplicial in $G'_{\{v_i, \cdots, v_n\}}$, it is quasi-simplicial in $G_{\{v_i, \cdots, v_n\}}$, because the elimination game only adds edges whose endpoints are non-probes at each step. Therefore, β is a qpeo of G.

(1) ⇒ (3): If the elimination game fails at some step, then the corresponding transitory graph has no quasi-simplicial vertex, so by the equivalence (1) $\iff$ (2), it fails to be N-T. Therefore, by Lemma 4.17, G is not N-T.

(3) ⇒ (2): Trivial. □

Complexity: A recognition algorithm can be given based on condition (3) of Theorem 4.18. This runs in $O(n^2 m')$ time, where m' is the number of edges of the

N-triangulation computed by the elimination game run on a qpeo, since a brute force approach will require $O(nm')$ time to find a quasi-simplicial vertex and process it.

This complexity is not as good as the $O(|P|m)$ time we found in Section 4.3. However, there may be, as is the case for chordal graphs, a LEX M-type algorithm which could compute a qpeo in $O(|N|m)$-time - a question we leave open.
We now will use our results to extend Dirac's Theorem 2.13 to N-T graphs:

Theorem 4.19 *Let $G = (P + N, E)$ be an N-T graph which is not a clique; then in G there are at least two non-adjacent quasi-simplicial vertices.*

Proof:
By induction on the number of vertices.

Clearly, any N-T graph on 4 vertices which is not a clique has two non-adjacent vertices which are quasi-simplicial.

Let us consider an N-T graph G with n vertices. By Theorem 4.18, G has a quasi-simplicial vertex x. If x sees all the other vertices in G, let G' be obtained by simply removing x from G. By the induction hypothesis, G' has two non-adjacent quasi-simplicial vertices, which are trivially also quasi-simplicial and non-adjacent in G.

Otherwise, let G' be obtained by saturating $\mathcal{N}_G(x)$ and removing x. According to Lemma 4.17, G' is an N-T graph. By the induction hypothesis, G' must have two non-adjacent quasi-simplicial vertices, at least one of which, call it z, is not in $\mathcal{N}_G(x)$, since in G', $\mathcal{N}_G(x)$ is a clique. We claim that z is quasi-simplicial in G. Suppose this is not the case: in G, z must see a non-edge $\{v, w\}$, with v a probe, which is not a non-edge of G', so edge vw must have been added to G' when making x simplicial. But since x is quasi-simplicial in G, there can be no such non-edge $\{v, w\}$ in $G_{\mathcal{N}(x)}$. Thus, in G, x and z are two non-adjacent quasi-simplicial vertices. □

4.6 Recognizing partitioned chordal probe graphs

In this section, we apply our results from Section 4.2 on N-triangulatable graphs to characterizing and recognizing partitioned chordal probe graphs.

Theorem 4.20 *Let $G = (P + N, E)$, with N a stable set. The following conditions are equivalent:*

1. *G is chordal probe .*
2. *All probes of G are quasi-LB-simplicial.*
3. *G contains no chordless cycle with two adjacent probes.*

Proof: This follows directly from Theorem 4.8 and Remark 4.4(4). □

Complexity: Provided we test that N is a stable set, the recognition algorithm for N-T graphs given in Section 4.3 also recognizes chordal probe graphs, with the same $O(|P|m)$ complexity.

Theorem 4.21 *Let $G = (P + N, E)$ be a graph, with N a stable set. The three following are equivalent:*

1. *G is a chordal probe graph.*
2. *G has a quasi-perfect elimination ordering.*
3. *A greedy elimination game on quasi-simplicial vertices succeeds.*

Proof:

This follows immediately from Theorem 4.18. □

Remark 4.22 *Theorem 4.21 defines an elimination process on neighborhoods which are split graphs, as the vertices in each neighborhood are partitioned into a clique of probes and a stable set of non-probes; moreover, they form a special kind of split graph,*

which can be qualified as 'complete split graph', meaning that all possible edges between a probe and a non-probe belong to the graph. This extends the simplicial elimination process on chordal graphs, where the elimination is on complete neighborhoods.

Theorem 4.19 also trivially extends to chordal probe graphs:

Corollary 4.23 *Let $G = (P + N, E)$ be a chordal probe graph which is not a clique; then in G there are at least two non-adjacent quasi-simplicial vertices.*

5 Non-partitioned Chordal Probe Graphs

Having solved the partitioned case for recognizing chordal probe graphs in Section 4, we will now go on to the non-partitioned case. Again we do this by first solving a more general problem.

5.1 Decomposing an arbitrary graph into C-components

Our approach to recognizing chordal probe graphs uses Lemma 3.1, which remarks that in any valid partition of a chordal probe graph, probes and non-probes must alternate on every chordless cycle. In order to study this phenomenon, we first propose a partition of the vertices of a graph into components which group together cycles of the graph, thus introducing a new graph decomposition.

Definition 5.1 *Let G be an arbitrary graph.*

1. *A* C-edge *is an edge which belongs to some chordless cycle.*
2. *A* C-path *is a path made out of C-edges.*
3. *A* C-component *is a maximal set of vertices in which there is a C-path connecting each pair of vertices in the set.*
4. *An* external edge *is an edge which has its endpoints in two different C-components.*

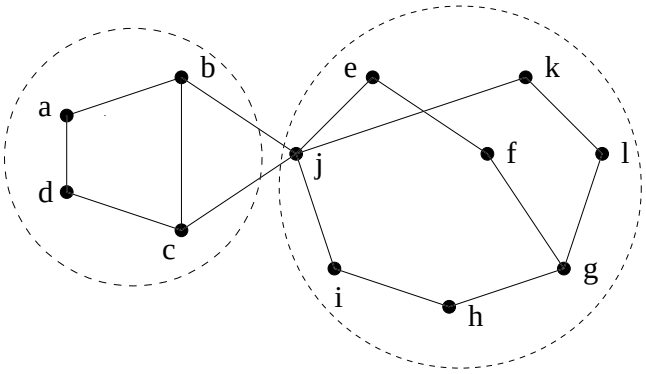

Figure 11: Illustration of example 5.2

Example 5.2 *Figure 11 shows the partition into C-components of a graph.*

Property 5.3 *Let G be an arbitrary graph; being connected by a C-path is an equivalence relation on the vertices of G; we will denote this relation by $\sim$.*

Proof: Trivially, if x and y are connected by a C-path, then y and x are also connected by the same path, thus ensuring symmetry. Transitivity: let μ_1 be a C-path from x to y, and μ_2 be a C-path from y to z; the concatenation of μ_1 with μ_2 is a C-path from x to z. □

Property 5.4 *Every chordless cycle is entirely contained in some C-component.*

Proof: Two vertices belonging to some chordless cycle are connected by a C-path, so by definition they must belong to some common C-component. □

Property 5.5 *Every antihole is entirely contained in some C-component.*

Proof: The antihole on five vertices is C_5, and therefore is entirely contained in some C-component by Property 5.4. Every vertex of an antihole on more than five vertices belongs to a C_4 along with every other vertex of the antihole. □

As a result of Properties 5.4 and 5.5, we have the following:

Theorem 5.6 *The decomposition into C-components is hole and antihole preserving.*

The partition induced by $\sim$ on the vertices of G very naturally defines a quotient graph, which we will denote by G^0.

Definition 5.7 *Let $G = (V, E)$ be an arbitrary graph. Let us define the* quotient graph $G^0 = (V', E')$ *of G, where V' is the set of C-components of G and there is an edge between two C-components X_i and X_j if there is an edge in G with one endpoint in X_i and the other in X_j.*

Theorem 5.8 *Let $G = (V, E)$ be an arbitrary graph. The quotient graph G^0 of G is chordal.*

Proof: Suppose graph G^0 is not chordal.
There must be a chordless cycle $C^0 = (X_1, \ldots, X_k, X_1)$ in G^0. We will construct a corresponding chordless cycle in G. Let us consider 3 consecutive components X_i, X_{i+1} and X_{i+2} of C^0, and let x be a vertex in X_{i+1}; x can see a vertex x' of another component of C^0 only if x' is either in X_i, or in X_{i+2}, else edge xx' corresponds to a chord of C^0.

In each component X_i of G^0, let us choose some vertex y_i which sees a vertex x_{i+1} of X_{i+1}. Thus our construction chooses in each component X_i two vertices, x_i and y_i: x_i is seen by y_{i-1}. Let P_i be a chordless path in X_i which connects x_i with y_i. Let us concatenate all these paths: we obtain a cycle, from which we can extract a chordless cycle C of G, which has vertices in different C-components, thus contradicting Property 5.4. □

Let us now discuss the case where a vertex does not belong to any chordless cycle. Recall that by Definition 2.25, a vertex is LB-simplicial if all its substars are cliques; by Property 2.24, a vertex is LB-simplicial if and only if it belongs to no chordless cycle. We will express this by the following property:

Property 5.9 *Let X be a C-component of a graph. The following propositions are equivalent:*

1. *X contains no chordless cycle.*
2. *$|X| = 1$.*
3. *X is an LB-simplicial vertex of the graph.*

We will call *trivial* a C-component which contains no chordless cycle.

Algorithm II that determines the C-components of a graph is based on Property 2.24, which uses substars to determine the vertices which belong to a chordless cycle together with some vertex y.

Algorithm II: C-COMPONENTS

input : A graph $G = (V, E)$.
output: The set $\mathcal{X}$ of C-components of G.

```
//At the beginning, all vertices are unmarked ;
i ← 0 ;
Q is an empty queue ;
while the graph is non-empty do
    i ← i + 1 ;
    Choose a vertex x, mark it and insert it into Q ;
    X_i ← {x} ;
    while Q is non-empty do
        Remove vertex y from Q ;
        Compute the substars of y ;
        foreach unmarked vertex v which is endpoint of a non-edge of some substar
        of y do
            Mark v and insert it into Q ;
            X_i ← X_i ∪ {v} ;
        𝒳 ← 𝒳 ∪ {X_i};
        Remove all vertices of X_i from the graph ;
```

Complexity: Computing the substars of a vertex y requires $O(m)$ time (see [6]). A vertex has no more than n substars, so there are no more than n^2 substars in the

graph. Inside each substar, a vertex v can be tested to be the endpoint of a non-edge in $O(deg(v))$. Algorithm C-COMPONENTS thus runs in $O(n^2m)$ time.

5.2 Cycle-bicolorable graphs

In this section, we present a new class of perfect graphs which generalizes chordal probe graphs in the case where no partition is given in advance. We exploit the property that in any valid partition, probes and non-probes must alternate on every chordless cycle.

Recall that by Lemma 3.1, in a chordal probe graph, probes and non-probes alternate in every chordless cycle. We will use Lemma 3.1 to introduce a new graph class.

Definition 5.10 *We will say that a graph $G = (V, E)$ is* cycle-bicolorable *if and only if each vertex can be labeled with one of two colors in such a fashion that the colors alternate in every chordless cycle.*

Note that on a C-path in a cycle-bicolorable graph, the colors must alternate.

5.3 Recognizing cycle-bicolorable graphs

The following proposition will allow us to characterize cycle-bicolorable graphs by considering each C-component separately.

Proposition 5.11 *A graph is cycle-bicolorable if and only if each of its C-components is cycle-bicolorable.*

Proof: By Property 5.4, every chordless cycle of G is entirely contained in a unique C-component of G. Thus, coloring the chordless cycles inside each C-component is equivalent to coloring all chordless cycles. □

Lemma 5.12 *Each cycle-bicolorable C-component has exactly 2 opposite bicolorings.*

Proof: Let us consider a C-component X_i, and let κ_1 be a bicoloring of X_i. By exchanging the colors of every vertex, another bicoloring κ_2 is obtained. Suppose there is a third possible coloring κ_3. Let x be a vertex whose color is different in κ_1 and κ_3. We claim that every other vertex in X_i has a different coloring in κ_1 and in κ_3. Suppose by contradiction that some vertex y of X_i has the same color in κ_1 as in κ_3. There is a C-path connecting x and y; since the colors of in κ_3 must alternate on this path, the color of x uniquely determines the color of y, a contradiction. Thus, the color of every vertex of X_i is different in κ_1 as in κ_3, so κ_3 is the same as κ_2. □

Lemma 5.12 justifies the following definition:

Definition 5.13 *Let X be a cycle-bicolorable C-component of a graph G. The bicoloring of G_X induce a unique partition of the vertices into $V_1 + V_2$, which we will call the* color bipartition of X.

To recognize cycle-bicolorable graphs, we can easily adapt Algorithm C-COMPONENTS to arbitrarily assign a color to some vertex x and accordingly color all the vertices which belong to a chordless cycle along with x, and then use the new colorings obtained to color other vertices, thus bicoloring an entire C-component of the graph whenever this is possible. Algorithm III recognizes cycle-bicolorable graphs.

Correctness:

Theorem 5.14 *The input graph G is cycle-bicolorable if and only if Algorithm CYCLE-BICOLORABLE RECOGNITION does not return 'failure'.*

Proof:

$\Leftarrow$ Let us first assume the algorithm did not fail. Suppose that there are two vertices y and v which have the same color, but which are adjacent on a chordless cycle, and w.l.o.g. let y be colored before v. By Property 2.24, v is an endpoint of a non-edge of a substar of y, so when y is processed, the algorithm either colors v with a color different from that of y or returns 'failure' - a contradiction.

$\Rightarrow$ Let us now assume that the graph is cycle bi-colorable, but that the algorithm returns 'failure' when processing C-component X_i. When the first vertex x of X_i is colored black, the alternating coloring becomes uniquely defined, according to Lemma 5.12. Let us consider the first time a vertex, call it y, gives one of its neighbors v a wrong color: call 1 the color of y and 2 the color which y gives to v; since this coloring is wrong, v should have color 1, the same as y; but this is impossible: since v is an endpoint of a non-edge of a substar of y, v and y are adjacent on some chordless cycle, a contradiction. $\square$

Complexity: The complexity is the same as that of Algorithm C-COMPONENTS, namely $O(n^2m)$.

Algorithm III: CYCLE-BICOLORABLE RECOGNITION

input : A graph $G = (V, E)$.

output: 'failure' if G is not bicolorable, otherwise the set $\mathcal{X}$ of C-components of G and a black/white coloring of each C-component such that the colors alternate on every chordless cycle.

```
//At the beginning, all vertices are uncolored ;
i ← 0, 𝒳 ← ∅ ;
Q is an empty queue ;
while the graph is non-empty do
    i ← i + 1 ;
    Choose a vertex x, color it with black and insert it into Q ;
    X_i ← {x} ;
    while Q is non-empty do
        Remove vertex y from Q ;
        Compute the substars of y ;
        foreach vertex v which is endpoint of a non-edge of some substar of y do
            if v has same color as y then
                return (failure) ;
            if v is uncolored then
                color v with color different from y's, and insert v into Q ;
            X_i ← X_i ∪ {v} ;
    𝒳 ← 𝒳 ∪ {X_i};
    Remove all vertices of X_i from the graph ;
```

5.4 Some properties of cycle-bicolorable graphs

Theorem 5.15 *The class of cycle-bicolorable graphs is perfect.*

In order to prove Theorem 5.15, we will need the following Lemmas:

Lemma 5.16 *A cycle-bicolorable graph has no odd hole.*

Proof: It is well known that an odd chordless cycle cannot be labeled with two colors in a fashion that the colors alternate. □

Lemma 5.17 *A cycle-bicolorable graph has no antihole.*

Proof: Let $G = (V, E)$ be a cycle-bicolorable graph. By Lemma 5.16, G has no induced $\overline{C_5}$, since $\overline{C_5}$ is isomorphic to C_5. Suppose there exists $k \geq 6$, such that C_k is a chordless cycle of $\overline{G}$, with $C_k = (x_1, \ldots, x_k, x_1)$ Observe that $C' = (x_2, x_4, x_1, x_5, x_2)$ is a cycle of length 4 in G. In any bicoloring of V into black and white, x_1 and x_2 have the same color, w.l.o.g. black. Observe that x_1 sees all the vertices in $\overline{C_k}$, except for x_2 and x_k. Therefore, all the vertices in $\overline{C_k}$, except possibly x_2 and x_k, are white. But $C'' = (x_3, x_5, x_2, x_6, x_3)$ is also a chordless cycle of length 4 in G. In any bicoloring of V, either x_2 and x_3 are black or x_5 and x_6 are black, a contradiction. □

Theorem 5.15 follows directly from the Strong Perfect Graph Theorem and from Lemmas 5.16 and 5.17.

Remark 5.18 *There are graphs with no odd antiholes and no odd holes which are not cycle-bicolorable, as is the case for an even antihole. Figure 12 shows a graph, for which we thank Frédéric Maffray, which has no antiholes and no odd holes, and which is a Meyniel graph and is perfectly orderable, but which is not cycle-bicolorable.*

Proposition 5.19 *Let G be a cycle-bicolorable graph, where we arbitrarily call P and N the classes induced by a color-bipartition of each C-component of G. Then $G = (P + N, E)$ is an N-T graph.*

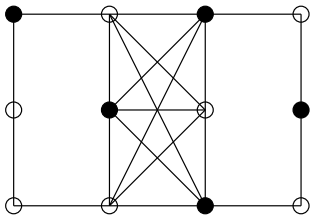

Figure 12: A perfectly orderable Meyniel graph with no antiholes and no odd holes, but not cycle-bicolorable.

Note that there are 2^t color-bipartitions where t is the number of C-components.

Proof: (of Proposition 5.19)

By definition of a cycle-bicolorable graph, in the bicoloring of G, there can be no chordless cycle with two consecutive vertices which have the same color; let us arbitrarily call the color classes in each C-component of G probes and non-probes: there can be no chordless cycle with two consecutive probes, so by Theorem 4.8, graph G is N-T with respect to any partition induced by the bicolorings of its C-components. □

Note that the converse of Proposition 5.19 does not hold, as N-T graphs are not perfect, and thus not always cycle-bicolorable, as is the case for the chordless cycle C_5.

5.5 Recognizing non-partitioned chordal probe graphs

From Lemma 3.1, we can easily deduce the following:

Theorem 5.20 *Chordal probe graphs are cycle-bicolorable graphs.*

The converse fails to hold: the complement of a P_6 is cycle-bicolorable but not chordal probe by Lemma 5.21.

In Section 5.2, we saw that a cycle-bicolorable graph can easily be bipartitioned, and we gave an $O(n^2m)$ algorithm to do this. We will now apply our results to the recognition of chordal probe graphs.

Lemma 5.21 *Let X_i be a C-component of a cycle-bicolorable graph. Then G_{X_i} is a chordal probe graph if and only if one of the colors of X_i forms a stable set.*

Example 5.22 *In Figure 13, the white vertices form a stable set and can be labeled as non-probes; the graph is chordal probe.*

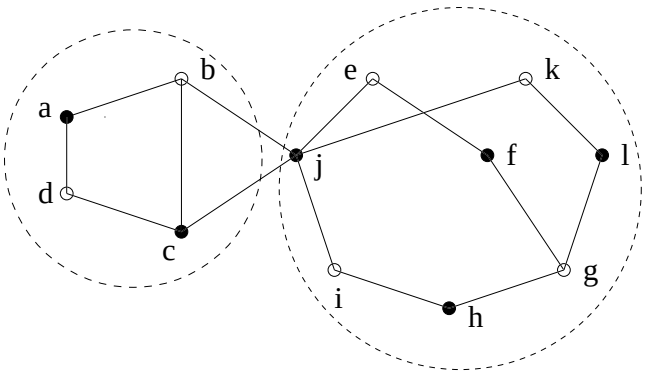

Figure 13: Illustration of example 5.22

Proof: (of Lemma 5.21)

$\Rightarrow$ Let G_{X_i} be a chordal probe graph, and let $P + N$ be a partition of X_i into probes and non-probes where N is a stable set. Since probes and non-probes alternate on every cycle, $P + N$ is the unique color bipartition of X_i, by Lemma 5.12. Thus, one of the colors, namely N, is the required stable set.

$\Leftarrow$ Let G_{X_i} be cycle-bicolorable such that one of the classes induced by the bicoloring, call it N, is a stable set. By Proposition 5.19, G_{X_i} is N-T with respect to this partition, so by definition G_{X_i} is chordal probe . $\square$

In [28], an algorithmic approach was presented to recognize chordal probe graphs in the case where the graph is weakly chordal. We observed in [7, 8] (without proof) that the method, (called Procedure *'PropagateConstraintGraph'*), can also be applied to arbitrary chordal probe graphs, using the following additional Lemma. The recognition algorithm that we will present here is a further modification, and we provide a detailed proof.

Lemma 5.23 *Let G be a chordal probe graph, let X_i and X_j be two bipartite C-components of the graph. Let x be a vertex of X_i which is an endpoint of at least two external edges connecting x to X_j. Then for any chordal probe partition $P+N$ of V, x is a probe.*

Proof: Let Y be the set of vertices of X_j which x sees. If G_Y has at least one edge e, then one of the endpoints of e is a probe and the other a non-probe, since X_j is bipartite; this forces x to be a probe. If G_Y has no edge, then let us choose a, b and a chordless path P in X_j such that P is shortest possible over all such pairs $\{a, b\}$; P together with edges ax and bx forms a chordless cycle, which is not fully contained in a C-component, a contradiction. □

Remark 5.24 *In the case of N-triangulatable and cycle-bicolorable graphs, Proposition 5.19 showed that it was sufficient to combine any local assignment of $P+N$ to the cycle-bicoloring of each C-component to obtain an N-T graph. This is not the case for chordal probe graphs; we cannot simply apply Lemma 5.21 to each C-component and combine the results, since globally we must maintain N as a stable set. For this reason, we must insure that the external edges, which join one C-component to another, obey the constraint that their endpoints may not both be non-probes.*

The considerations described in Remark 5.24 lead us to the Algorithm NON-PARTITIONED CHORDAL PROBE GRAPH RECOGNITION presented below, which decides whether an arbitrary graph G is chordal probe and if yes, computes a partition of the vertex set into probes and non-probes. Step 1 checks whether G is cycle-bicolorable, and if so, produces a bicoloring. Step 2 verifies that each C-component satisfies Lemma 5.21; if only one color is a stable set, then the labeling into probes and non-probes for that C-component is fixed by the LABEL-COMPONENT routine; if both colors are stable sets, then that component is a bipartite subgraph and no decision is made (yet) for its labeling. Step 3 applies the condition in Lemma 5.23; for every vertex x

in a unlabeled (hence bipartite) C-component which sees two vertices in another unlabeled C-component, the labeling into probes and non-probes for the C-component containing x is fixed by the LABEL-COMPONENT routine. *Notice that LABEL-COMPONENT has the side effect of building a global queue Q of external edges which will have to be checked for consistency later in the algorithm.*

Step 4 checks the external edges uv for which u has been labeled a non-probe to verify that v is either a probe or unlabeled; in the latter case, the labeling for the C-component containing v is fixed by the LABEL-COMPONENT routine. The routine PROPAGATE is used for this purpose, terminating when the queue of all such edges is empty. Step 5 simply declares that the graph is now *recognized as being chordal probe*, although some C-components may still be unlabeled. Step 6 completes the labeling in a greedy manner. Algorithm V and VI give the subroutines LABEL-COMPONENT and PROPAGATE which are used by the main algorithm.

Algorithm IV: NON-PARTITIONED CHORDAL PROBE GRAPH RECOGNITION

input : A graph $G = (V, E)$.

output: 'NO' if G is not chordal probe , otherwise a chordal probe labeling of V with P and N.

Q is an empty queue ;

1 CYCLE-BICOLORABLE RECOGNITION;

if *'failure' is returned* **then**

  return *('NO');*

2 **foreach** *C-component* X_i **do**

  if *neither black nor white induces a stable set* **then**

    return *('NO');*

  if *only one class of the color-bipartition induces a stable set* **then**

    choose a vertex x of this color and label it N;

    LABEL-COMPONENT (x, N);

// At this point, the only components left unlabeled are bipartite graphs;

3 **foreach** *external edge* $x_i x_j$*, with* x_i *in C-component* X_i *and* x_j *in C-component* X_j*,* $i \neq j$*, where* X_i *and* X_j *are unlabeled, and such that* x_i *sees at least two vertices in* X_j **do**

  LABEL-COMPONENT(x_i, P) ;

4 PROPAGATE;

if *'failure' is returned* **then**

  return *('NO');*

// At this point, any external edge uv with u labeled and v unlabeled ;
// will have u labeled as P;

5 *//*G is chordal probe ;

$V_1 \leftarrow$ labeled vertices ;

$V_2 \leftarrow$ unlabeled vertices ;

//At this point, some of the C-components may remain unlabeled;

6 **while** *there remain some unlabeled C-components* **do**

  Arbitrarily choose an unlabeled vertex x;

  LABEL-COMPONENT (x, P) ;

  PROPAGATE ;

Algorithm V: LABEL-COMPONENT

input : A vertex x and its label.

output: Labels the vertices which are in the same C-component X_i as x and adds to global queue Q all external edges with one endpoint in X_i labeled N.

Label all the vertices of X_i with P or N according to the label of x ;
foreach *external edge uv with u in X_i and labeled N* **do**
  Add (u, v) to Q ;

Algorithm VI: PROPAGATE

output: Returns 'failure' if there is a conflict, otherwise labels the vertices of some unlabeled C-component and may add some to-be-processed edges to the global queue Q.

while *Q is non-empty* **do**
  $(u, v) \leftarrow$ *dequeue*(Q); //u is labeled N ;
  if *v is labeled N* **then**
    return *('failure') ;*
  if *v has no label* **then**
    LABEL-COMPONENT(v, P);

Complexity: In the Algorithm NON-PARTITIONED CHORDAL PROBE GRAPH RECOGNITION, the bottleneck is Step 1, which requires $O(n^2m)$ time. All other steps have lower complexity. An N-triangulation can be obtained using the results for the partitioned case.

Correctness:

Theorem 5.25 *The input graph G is chordal probe if and only if Algorithm NON-PARTITIONED CHORDAL PROBE GRAPH RECOGNITION does not return 'NO'. Moreover, when G is a chordal probe graph, the Algorithm produces a valid chordal probe partition $P + N$.*

Proof:

$\Rightarrow$ Suppose G is a chordal probe graph. We show that a 'NO' answer gives a contradiction.

If the Algorithm returns 'NO' in Step 1, then, by Theorems 5.20 and 5.14, G could not be a chordal probe graph. Therefore, Step 1 succeeds, and produces a cycle-bicoloring of G. If the Algorithm returns 'NO' in Step 2, then, by Lemma 5.21, G could not be a chordal probe graph. Therefore, Step 2 succeeds, and assigns the probe/non-probe labeling for all non-bipartite C-components. Step 3 always succeeds, applying Lemma 5.23.

Note that a C-component is labeled with a probe/non-probe assignment only when the opposite assignment has been found to be contradictory. As stated in Remark 5.24, external edges joining one C-component to another, must obey the constraint that their endpoints may not both be non-probes. The routine PROPAGATE checks each such edge, and when necessary, forces other C-components to be labeled accordingly. Thus, if the Algorithm returns 'NO' in Step 4, then, G could not be a chordal probe graph. Therefore, Step 4 succeeds, and the following properties hold:

Claim 1: *If X_i and X_j are unlabeled C-components, then there is at most one edge joining them; we call it the* exclusive *edge.*

[Proof of Claim 1: If there were two such external edges uv and $u'v'$ with $u, u' \in X_i$ and $v, v' \in X_j$, by Lemma 5.23, having completed Steps 3-4 we have $u \neq u'$ and $v \neq v'$. From this it follows that the subgraph induced by uv and $u'v'$ and chordless paths in X_i and X_j connecting u with u' and v with v', respectively, contains a chordless cycle, contradicting Property 5.4.]

Claim 2: *If $\{X_1, X_2, \ldots, X_k\}$ are unlabeled C-components, forming a cycle C in the quotient graph G^0, i.e., the exclusive edges $u_i v_{i+1}$ exist joining X_i, X_{i+1} for all i (arithmetic mod k), then $u_i = v_i$ for all i.*

[Proof of Claim 2: (By induction on k.) If $k = 3$ then combining shortest paths in X_1, X_2, X_3 connecting u_i with v_i, respectively, will yield a chordless cycle in G of length > 4 if one of the equalities fails to hold, contradicting Property 5.4. If $k \geq 4$ then C has a chord, since the quotient graph G^0 is chordal (Property 5.8, thus splitting C into two smaller cycles C_1, C_2. So by induction, applying Claim 2 to

C_1, C_2, we obtain all k equalities.]

The remainder of the proof of Theorem 5.25 in this direction follows from the following Claim:

Claim 3: *Step 6 never fails.*

[Proof of Claim 3: Suppose Step 6 fails at an iteration where x was chosen to be labeled arbitrarily, and where failure occurred for external edge uv, where vertex u is in component X_u and v is in X_v, both labeled N. The propagation defines a search tree in the subgraph of the quotient graph induced by the components labeled by that iteration of the propagation. Let X_j be the smallest common ancestor of X_u and X_v.

Consider the cycle C in G^0 formed by the exclusive edge uv and the two paths in the search tree from X_j to X_u and from X_j to X_v. Note, however, that the exclusive edges on these paths have the endpoint of the parent labeled N and the endpoint of the child labeled P, by the routine PROPAGATE. This contradicts Claim 2, and completes the proof of Claim 3.]

$\Leftarrow$ If the Algorithm succeeds, then the probe/non-probe partition $P + N$ is a cycle-bicoloring (Step 1), so by Proposition 5.19, G is and N-T graph with respect to $P + N$. Furthermore, N is a stable set (all external edges have been checked by PROPAGATE), so by definition, G is a chordal probe graph, and we have produced a valid chordal probe partition $P + N$. $\square$

5.6 Recognizing even-chordal non-partitioned chordal probe graphs

Remark 5.26 *A chordless cycle of length 4 has exactly two valid partitions where either pair of non-adjacent vertices could be the non-probes. Moreover, assigning any one of its four vertices to be a probe (respectively non-probe) forces its neighbors in the cycle to be non-probes (resp. probes) and its non-neighbor to be a probe (resp.*

non-probe).

Lemma 5.27 *If (P_1, N_1) and (P_2, N_2) are two different partitions of a C-component of a weakly chordal graph G, such that probes and non-probes alternate in every chordless cycle in G, then P_1=N_2 and N_1=P_2.*

Proof: Suppose there exists $v \in V$, which is either assigned to be a probe in both partitions or a non-probe in both partitions.
We will now prove by induction on the length of a C-path $Q = [v = v_0, \ldots, v_i, \ldots, v_m]$ that the vertex v_m has the same assignment in both partitions.
In the case that $Q = [v_0, v_1]$, since the edge (v_0, v_1) belongs to a 4-cycle, v_1 has the same assignment in both partitions, due to Remark 5.26.
Assume that for every path of length $i < m$, the vertex v_i has the same assignment in both partitions.
Let $Q = [v = v_0, \ldots, v_m]$ be a C-path of length m. By induction, each vertex v_i $(1 \leq i < m)$ must have the same assignment in both partitions. Consequently, the vertex v_m must have the same assignment in both partitions, since the length of the C-path $[v_i, \ldots, v_m]$ is less than m.
Therefore, all the vertices in G have the same assignment in both partitions and hence $P_1 = P_2$ and $N_1 = N_2$. Contradiction!
Thus, the assignment of each vertex in (P_1, N_1) is different than its assignment in (P_2, N_2), so $P_1 = N_2$ and $P_2 = N_1$. $\square$

Let $G = (V, E)$ be a connected graph and let $S(G)$ denote the set of all 4-cycles in G. We define the set $S_x(G) = \{C \in S(G) | x \in V(C)\}$ for each $x \in V$. The following *FindC4Components* procedure finds all the C-components in a weakly chordal graph. The procedure is a variant of breadth first search, and combines those 4-cycles that are not vertex disjoint into a C-component. Those vertices which are not in a vertex set of a 4-cycle are singleton C-components.

FindC4Components Procedure.

input : A weakly chordal graph $G = (V, E)$, the set $S(G)$ and the sets $S_x(G)$ for each $x \in V$.

output: The set $\mathcal{X} = \{X_1, \ldots, X_t\}$ of C-components in G.

```
i ← 1;
Mark all vertices in G white;
while there are white vertices in G do
    arbitrarily choose a white vertex v ∈ V; X_i ← {v};
    if S_v(G) ≠ ∅ then
        Mark v grey and insert it into the queue Q;
        while Q is not empty do
            Remove vertex u from Q and mark it black;
            foreach C ∈ S_u(G) do
                if All vertices in C besides u are white or grey then
                    X_i ← X_i ∪ V(C);
                    insert all white vertices of C into Q;
    else
        mark v black;
    i ← i + 1;
```

Lemma 5.28 *The* FindC4Components *procedure finds the set of C-components of a weakly chordal graph G. The time complexity of the procedure is $O(m^2)$.*

Proof: Each X_i is C-connected, since it is either a single vertex, which is not C-connected to any other vertex in G and hence a singleton C-component, or a union of non-disjoint 4-cycles.

Suppose X_i is not a maximal C-connected component in G. Then there exists a pair of vertices $x \in V(X_i)$, $y \notin V(X_i)$ that are connected by a C-path. Let $P = [x = v_1, v_2, \ldots, v_{n-1}, v_n = y]$ be a C-path connecting x and y and let $C_1, \ldots, C_n \in S(G)$ be 4-cycles in G, satisfying $v_i, v_{i+1} \in V(C_i)$ for all $i = 1, \ldots, n-1$. The 4-cycles $C_1, \ldots, C_n$ are not vertex disjoint and therefore are all contained in X_i. Contradiction.
Only white or grey vertices are inserted into the queue, and each vertex is marked black after it is removed from the queue. Thus, each vertex is inserted into the queue only once. For each vertex u that is removed from the queue, the procedure checks all the cycles in the set $S_u(G)$. Therefore, each 4-cycle is checked at most 4 times and the time complexity of the procedure is $O(m^2)$. $\square$

The *FindPartitions* procedure finds a valid partition of a weakly chordal C-component X_i. If H_i does not have a valid partition, then the *FindPartitions* procedure fails. The procedure first finds a partition of vertices, such that probes and non-probes alternate on every chordless cycle. It marks an arbitrary vertex v to be a probe and then propagates the assignment of all the other vertices accordingly.

Applying Lemma 5.27, there exist at most two such partitions of X_i, where in the case of two valid partitions the set of probes in one of the partitions is exactly the set of non-probes in the other partition. Then the procedure checks if non-probes are a stable set and if probes are a stable set in the partition.

FindPartitions procedure

1 $(P, N) \leftarrow (\{v\}, \emptyset)$, where v is an arbitrary vertex in X_i;

$l_i \leftarrow 0$;

insert the vertex v into a queue Q;

while *the queue* Q *is not empty* **do**

- Remove vertex u from Q;
- **foreach** *cycle* $C_i \in S_u(G)$ **do**
  - **if** *the non-neighbor* x *of* u *in* C_i *has a different assignment than* u **then**
    - return 'failure';
  - **if** x *is not yet assigned* **then**
    - Assign x to have the same assignment as u;
    - Insert x into Q;
  - **if** *either neighbor* y_1 *or* y_2 *of* u *in* C_i *has the same assignment as* u **then**
    - return 'failure';
  - **if** *either* y_i *is not yet assigned* **then**
    - Assign y_i to have different assignment than u;
    - Insert y_i into Q;

2 **if** N *is not a stable set* **then**

- **if** P *is a stable set* **then**
  - Swap P and N;
- **else**
  - Return 'failure';

Claim 5.29 *Let* X_i *be a non-singleton* C*-component of a weakly chordal graph* G*. The* FindPartitions *procedure finds all the valid partitions of* X_i*. The time complexity of the procedure is* $O(|E(X_i)|^2)$*.*

Proof: Each partition that is found by the *FindPartitions* procedure is valid, since probes and non-probes alternate in every 4-cycle (this is checked in step 1) and non-probes are a stable set (this is checked in step 2).

According to Lemma 5.27, there exists at most two partitions where probes and non-probes alternate in every cycle, while the set of probes in one of the partitions is exactly the set of non-probes in the other partition. The *FindPartitions* procedure checks if any of the two partitions is valid at step 2, by checking if the probes and the non-probes are stable sets.

Each vertex is inserted into the queue exactly once. The procedure removes each vertex u from the queue and checks all the cycles into the set $S_u(G)$. Therefore, each 4-cycle is checked at most 4 times and the time complexity of the step (a) is $O(|E(X_i)|^2)$. At step 2, we check whether a set of vertices is a stable set, this can be done in time complexity of $O(|E(X_i)|^2)$. Thus, the time complexity of the *FindPartitions* procedure is $O(|E(X_i)|^2)$. $\square$

Now, we give a modified algorithm CYCLE-BICOLORABLE RECOGNITION for weakly chordal graphs.

Algorithm VII: CYCLE-BICOLORABLE RECOGNITION OF WEAKLY CHORDAL GRAPHS

input : A weakly chordal graph $G = (V, E)$.

output: 'failure' if G is not bicolorable, otherwise the set $\mathcal{X}$ of C-components of G and a black/white coloring of each C-component such that the colors alternate on every chordless cycle.

Construct the set $S(G)$ of all 4-cycles in G and the sets $S_x(G)$ for each $x \in V$;
1 Find the set $mathcalX = \{X_1, \ldots, X_t\}$ of C-components of G by calling the *FindC4Components* procedure;
2 **foreach** $X_i \in \mathcal{X}$ **do**
  if X_i *is a singleton* C*-component* **then**
    Color it black;
  else
    Find a valid partition (P, N) of X_i by calling the *FindPartitions* procedure;
    if FindPartitions *fails* **then**
      Return 'failure';

Lemma 5.30 *The time complexity of the Algorithm VII is* $O(m^2)$.

Proof: The sets $S(G)$ and $S_x(G)$ can be found in $O(m^2)$ according to Remark 4.1.

The time complexity of *FindC4Components* procedure is $O(m^2)$ due to Lemma 5.28.

The *FindPartitions* procedure is called for each C-component X_i of the graph. Since the time complexity of *FindPartitions* procedure is $O(|E(X_i)|^2)$ according to Lemma 5.29, the time complexity of Algorithm VII is $\sum_i O(|E(X_i)|^2) = O(m^2)$. □

Algorithm VIII: Recognition of even-chordal non-partitioned chordal probe graphs .

input : A graph $G = (V, E)$.

Verify that the graph G is weakly chordal using the algorithm in [5] or [37], otherwise return 'failure';

Construct the set $S(G)$ of all 4-cycles in G and the sets $S_x(G)$ for each $x \in V$;

Call to CYCLE-BICOLORABLE RECOGNITION OF WEAKLY CHORDAL GRAPHS algorithm; Goto step 2 of the NON-PARTITIONED CHORDAL PROBE GRAPH RECOGNITION;

Correctness: The correctness of the algorithm follows directly from Lemma 5.28, Lemma 5.29 and Theorem 5.25.

Complexity: The complexity of the algorithm is $O(m^2)$. In the Algorithm NON-PARTITIONED CHORDAL PROBE GRAPH RECOGNITION, the bottleneck is Step 1, which now requires only $O(m^2)$ by Lemma 5.30. All other steps have lower complexity.

6 Conclusion and open questions

Though chordal probe graphs were originally defined as a generalization of interval probe graphs, they also have their own computational biology application as a special case of reconstructing phylogenies, tree structures which model genetic mutations, when part of the information is missing. In fact, the polynomiality of N-T graph recognition which we show in this paper also provides an interesting tractable subcase of the chordal graph sandwich problem [25].

Regarding the structure of chordal probe graphs and N-T graphs, it appears clearly from the results in this paper that they are similar to chordal graphs in many respects, with similar characterizations. The evident difference is that chordal graphs have no chordless cycles, but we have shown that such cycles can be structured into

C-components, which enables us to handle them efficiently. We believe that C-components in a general graph may have many interesting properties, one example of which is the chordality of the quotient graph.

Though we have solved partitioned and non-partitioned chordal probe graph recognition, we leave open the question of polynomial non-partitioned interval probe recognition; however, our results solve this problem in some subcases.

Part II

Intersection graphs of paths in a tree

7 Introduction

7.1 Definitions and Motivation

We consider a generalization of edge intersection graphs of paths in a tree. Let $\mathcal{P}$ be a collection of nontrivial simple paths in a tree T. We consider two different types of intersection graphs from the pair $\langle \mathcal{P}, T \rangle$, namely the VPT and k-EPT graphs. *The vertex intersection graph* $VPT(\mathcal{P})$ has vertices which correspond to the members of $\mathcal{P}$, such that two vertices are adjacent in $VPT(\mathcal{P})$ if the corresponding paths in $\mathcal{P}$ share a vertex in T. An undirected graph G is called *a vertex intersection graph of paths in a tree* (VPT) if $G{=}VPT(\mathcal{P})$ for some $\mathcal{P}$ and T, and $\langle \mathcal{P}, T \rangle$ is a VPT representation of G. The VPT graphs were studied in [20, 23].

Similarly, we introduce the k-edge ($k \geq 1$) intersection graph k-$EPT(\mathcal{P})$ to have vertices which correspond to the members of $\mathcal{P}$, such that two vertices are adjacent in k-$EPT(\mathcal{P})$ if the corresponding paths in $\mathcal{P}$ share k edges in T. An undirected graph G is called *a k-edge intersection graph of paths in a tree* (k-EPT) if $G = k$-$EPT(\mathcal{P})$

for some $\mathcal{P}$ and T, and $\langle \mathcal{P}, T \rangle$ is a k-EPT representation of G. The case of $k = 1$ (known as the EPT graphs) was introduced in [23, 24]. Finally, we denote by EPT* the class of graphs consisting of the k-EPT graphs for all possible $k \geq 1$.

Clearly, for a given collection $\mathcal{P}$, the vertices of the k-EPT graph $k\text{-}EPT(\mathcal{P})$ and those of VPT graph $VPT(\mathcal{P})$ are the same, and every edge of $k\text{-}EPT(\mathcal{P})$ is also an edge of $VPT(\mathcal{P})$, but not conversely.

The k-EPT graphs are used in network applications. The problem of scheduling undirected calls in a tree network is equivalent to the problem of coloring a 1-EPT graph (see e.g. [23]). The communication network is represented as an undirected interconnection graph, where each edge is associated with a physical link between two nodes. An undirected call is a path in the network. In the restricted setting, two calls compete on network resources if they share k physical links, and therefore must be scheduled not at the same time. When the network is a tree, this model is clearly a k-EPT representation. Coloring the k-EPT graph, such that two adjacent vertices have different colors, implies that paths sharing at least k edges in the k-EPT representation have different colors. Therefore, the coloring may be associated with a scheduling of the calls, such that calls of the same color can be scheduled at the same time. There are different extensions of this model for different network problems. For example in [17], the call scheduling problem is examined in two different variants of the model. First, considering bidirectional calls, where both connections associated with a call use the same undirected path in the network, and the second, where a call is directed from source to target, without need for a connection in the opposite direction. Both variations of call scheduling problem are proved to be NP-hard for trees and rings in [17].

Another application is assigning wavelengths to connections in an optical network, where virtual connections share physical links by wavelength-division multiplexing. This problem is equivalent to the problem of coloring an EPT graph as follows. In the optical interconnection graph, every edge is associated with optical link between two vertices. Virtual connections that share the same optical link must have different

wavelengths. The coloring may be associated with assignment of wavelengths, such that connections of the same color can be assigned the same wavelength. For a survey on related work on this optical model see e.g. [2]. In [18], optimization problems on optical networks with multiple parallel fibers between adjacent nodes are investigated. In such networks, two paths on the link can use the same wavelength if they are carried on different fibers.

A polynomial algorithm for recognizing VPT graphs was introduced by Gavril in [20]. Golumbic and Jamison showed that the recognition problem and the coloring problem are both NP-complete for EPT graphs [23, 24]. Tarjan gave a $\frac{3}{2}$-approximation algorithm for coloring EPT graphs [54].

We generalize EPT graphs into k-EPT graphs, by quantizing the number of intersecting edges of paths. The vertices of the k-EPT graph k-$EPT(\mathcal{P})$ and those of the EPT graph $EPT(\mathcal{P})$ are the same, but the adjacency of the vertices is different. It means that for a given collection $\mathcal{P}$, the vertices of k-$EPT(\mathcal{P})$ graph for any $k \geq 1$ and those of the $VPT(\mathcal{P})$ graph are the same, and every edge of k-$EPT(\mathcal{P})$ graph is also an edge of $VPT(\mathcal{P})$, but not conversely. Namely, the edge sets are nested: $E(k\text{-}EPT(\mathcal{P})) \subseteq E((k-1)\text{-}EPT(\mathcal{P})) \subseteq \cdots \subseteq E(1\text{-}EPT(\mathcal{P})) \subseteq E(VPT(\mathcal{P}))$. We prove that the k-EPT graphs are a generalization of EPT graphs in the following Lemma 7.1.

Lemma 7.1 *Let G be a 1-EPT graph, then G is a k-EPT graph, for every $k \geq 2$.*

Proof: Let $< \mathcal{P}, T >$ be a 1-EPT representation of G, i.e., $G = EPT(\mathcal{P})$. It is simple to construct a k-EPT representation $< \mathcal{P}', T' >$ of G, such that $G = k\text{-}EPT(\mathcal{P}')$. For every edge e in T, we add $k-1$ dummy vertices, dividing the edge e into k edges in T'.

Every two paths, which intersect on at least one edge in $EPT(\mathcal{P})$, intersect on at least k edges in k-$EPT(\mathcal{P}')$. Every two paths, that intersect on less than one edge in $EPT(\mathcal{P})$, intersect on less than k edges in k-$EPT(\mathcal{P}')$. Therefore, G=k-$EPT(\mathcal{P}')$ is a k-EPT graph. $\square$

A totally different approach is adopted in tolerance models of paths in a tree. Intersection graphs of intervals on the line (interval graphs) are generalized to tolerance graphs, by adding an edge between two vertices in the tolerance graph when the size of the intersection of their intervals exceeds at least one of the tolerances. In the same fashion, one can generalize intersection graphs of paths in a tree (EPT) to tolerance intersection graphs of paths in a tree, by adding an edge between two vertices in the model, when the size of the intersection of their paths exceeds at least one of the tolerances. Thus, k-EPT graphs are tolerance intersection graphs of paths in a tree with constant tolerance corresponding to k edges in the tree. See [34] for more details.

In [39, 40, 41], Jamison and Mulder have placed these tolerance models into a more general setting. An (h, s, t)-subtree representation consists of a collection of subtrees of a tree, such that (i) the maximum degree of T is at most h, (ii) every subtree has maximum degree at most s, (iii) there is an edge between two vertices in the graph if and only if the corresponding subtrees have at least t vertices in common in T. The class of graphs that have an (h, s, t)-representation is denoted by $[h, s, t]$. If no restriction is imposed on the maximum degree of the host tree, then $h = \infty$. An (h, s, t)-representation is orthodox if two subtrees share a common leaf of T if and only if the two subtrees share t vertices.

The $[h, 2, 1]$ graphs are also known as path graphs [20] or VPT graphs [23], and $[h, 2, 2]$ graphs are the known EPT graphs. The k-EPT graphs are exactly the $[\infty, 2, k + 1]$ graphs, for $k \geq 1$. In this work, we investigate the class of $[h, 2, t]$ graphs, i.e., the intersection graphs of paths in a tree.

It is well-known [14, 21, 55] that the chordal graphs are equivalent to $[\infty, \infty, 1]$, namely the vertex intersection graphs of subtrees in a tree. This equivalence was strengthened in [46], where chordal graphs are proved to be equivalent to $[3, 3, 1]$ graphs. Moreover, it was proved by Jamison and Mulder that chordal graphs correspond to orthodox-$[3, 3, 1]$ graphs defined below. See also [34]. The class $[3, 3, 3]$ is studied in [40].

Since the class of chordal graphs corresponds to the class of vertex intersection graphs of subtrees in a tree and the VPT graphs are the class of vertex intersection graphs of paths in a tree, the VPT graphs are chordal.

Golumbic and Jamison [23] proved that the family of VPT is incomparable with the family of EPT graphs, but when restricted to degree 3 trees they coincide.

It is known that the recognition and the coloring of the 1-EPT graphs are NP-complete. We extend this result and prove that the recognition and the coloring of the k-EPT graphs are NP-complete for any fixed $k \geq 1$. We show that the problem of finding the largest clique on k-EPT graphs is polynomial, as was the case for 1-EPT graphs, and determine that there are at most $O(n^3)$ maximal cliques in a k-EPT graph on n vertices.

We consider variations of $[h, 2, t]$ by three main parameters: h, t and whether the graph has an orthodox representation. We give the complete hierarchy of relationships between the classes of weakly chordal, chordal, $[h, 2, t]$ and orthodox-$[h, 2, t]$ graphs for varied values of h and t.

7.2 Outline of our results on intersection graphs of paths in a tree

We continue to investigate the k-EPT graphs in section 8. We prove an equivalence result for EPT graphs that are also weakly chordal in section 9. We investigate orthodox representations and prove two equivalence theorems in section 10 and also show the complete hierarchy of chordal, weakly chordal, $[h, 2, t]$ and orthodox-$[h, 2, t]$ graphs for varied values of h and t in section 10. Section 11 deals with properties of $[h, 2, t]$ families of graphs when we restrict the maximum degree h of the host tree. We conclude with open problems and further research in section 12.

8 The k-edge intersection graphs of paths in a tree: First results

In this section we investigate the properties of the k-EPT graphs that are introduced in section 7. We prove that k-EPT graphs have the Helly property in section 8.1. Properties of chordless cycles in k-EPT graphs are shown in section 8.2. We characterize the clique structure of k-EPT graphs and show that a clique with maximum cardinality can be found in polynomial time in section 8.3. Finally, the NP-completeness of the recognition problem and the coloring problem for k-EPT graphs is proved in sections 8.4 and 8.5.

8.1 The Helly property

Definition 8.1 *Let $\Psi = \{S_i\}_{i \in I}$ be a collection of subsets of a set S. We say that Ψ has **Helly number h** if for all $J \subseteq I$, $\bigcap\{S_i | i \in J\} = \emptyset$ implies that there exist h indices $i_1, \ldots, i_h \in J$ such that $S_{i_1} \bigcap \cdots \bigcap S_{i_h} = \emptyset$.*

Theorem 8.2 [4] *Any collection of subtrees of a tree (and therefore any collection of paths of a tree) has Helly number 2.*

Definition 8.3 *Let $\Psi = \{S_i\}_{i \in I}$ be a collection of subsets of a set S. We say that Ψ has **strong Helly number s** if for all $J \subseteq I$, there exist s indices $i_1, \ldots, i_s \in J$ such that $S_{i_1} \bigcap \cdots \bigcap S_{i_s} = \bigcap\{S_i | i \in J\}$.*

Theorem 8.4 [23] *A finite collection of paths in a tree has strong Helly number 3.*

Clearly, the Helly number is less than or equal to the strong Helly number.

8.2 Chordless cycles in k-EPT graphs

The chordless cycle C_d is a 1-EPT graph and it has a unique 1-EPT representation which is called a **pie** in [24], defined as follows.

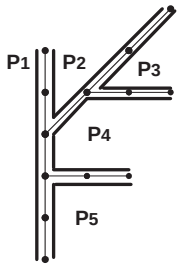

Figure 14: An $(3, 2, 3)$-representation of the graph C_5.

Definition 8.5 *Let $\langle \mathcal{P}, T \rangle$ be a 1-EPT representation of a graph G. A pie is a star subgraph of T with d edges $(a_0, b), \ldots, (a_{d-1}, b)$ such that each "slice" $(a_i, b) \cup (a_{i+1}, b)$ for $i = 0, 1, \ldots, d-1$ is contained in a different member of $\mathcal{P}$. (Addition is assumed to be modulo d). The vertex b is called the center of the pie.*

The following theorem shows that a pie essentially is the only 1-EPT representation for C_d. However, C_d does not have a unique k-EPT representation for $k > 1$. For example, the graph C_5 has different representations, one of them is shown in Figure 14.

Theorem 8.6 [24] *Let $\langle \mathcal{P}, T \rangle$ be a 1-EPT representation of a graph G. If G contains a chordless cycle of length $d \geq 4$, then $\langle \mathcal{P}, T \rangle$ contains a pie with d edges.*

Corollary 8.7 *Let G be an $[h, 2, 2]$ graph, then any induced chordless cycle in G has length at most h.*

The following proposition generalizes Corollary 8.7 to $[h, s, 2]$ graphs for $h \geq 3$.

Proposition 8.8 [39] *Let G be an $[h, s, 2]$ graph with $h \geq 3$, then any induced cycle in G has length at most h.*

8.3 Cliques in k-EPT graphs

In this section, we characterize the maximal cliques of the k-EPT graphs. This characterization will allow us to state a bound on the number of maximal cliques in a k-EPT graph and to find a maximum clique in polynomial time.

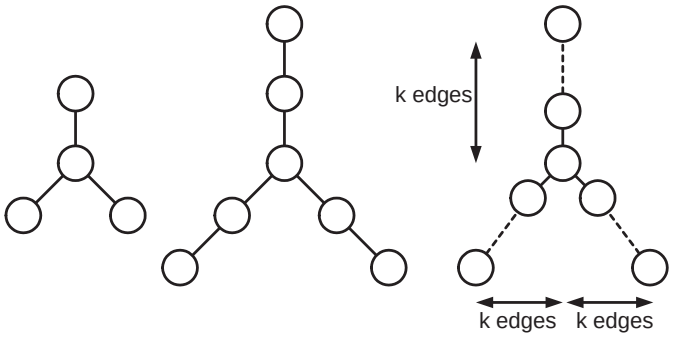

Figure 15: The graphs T_1^3, T_2^3, T_k^3.

Definition 8.9 *Let T_k^d be the tree that consists of one central vertex and d edge disjoint paths of k edges each intersecting only on the central vertex. We call these paths the* ***legs*** *of T_k^d. The* ***claw*** *graph $K_{1,3}$ is precisely T_1^3 and we call T_k^3 the k-**claw** graph. The graphs T_1^3, T_2^3 and T_k^3 are sown in Figure 15.*

Let $\langle \mathcal{P}, T \rangle$ be a k-EPT representation of G. For any simple path of k edges $[e_1, e_2, \ldots, e_k]$ in the tree T, let $\mathcal{P}[e_1, e_2, \ldots, e_k] = \{P \in \mathcal{P} | e_1, e_2, \ldots, e_k \in P\}$. For any copy of the k-claw T_k^3 in T, let $\mathcal{P}[T_k^3] = \{P \in \mathcal{P} | P$ contains two legs of $T_k^3\}$. The collection $\mathcal{P}[e_1, e_2, \ldots, e_k]$ corresponds to a clique in G and is called a **k-edge clique**. Similarly, $\mathcal{P}[T_k^3]$ also corresponds to a clique in G and is called a **k-claw clique**.

The following generalizes a result of Golumbic and Jamison, who proved the case of $k = 1$ [23].

Proposition 8.10 *Let $\langle \mathcal{P}, T \rangle$ be a k-EPT representation of a graph G. Any maximal clique of G corresponds to either a subcollection of paths of the form $\mathcal{P}[e_1, e_2, \ldots, e_k]$ for some edges $e_1, e_2, \ldots, e_k$ in T (a k-**edge clique**) or $\mathcal{P}[T_k^3]$ for some copy of the k-claw T_k^3 in T (a k-**claw clique**).*

Proof: Let C be a maximal clique of G=k-$EPT(\mathcal{P})$, and let $\mathcal{J} \subseteq \mathcal{P}$ be the subcollection corresponding to C, whose members intersect pairwise on at least k edges in T. Let $Q = \bigcap \{P | P \in \mathcal{J}\}$. By Theorem 8.2, Q contains at least one vertex and, by Theorem 8.4, there are paths $P_1, P_2, P_3 \in \mathcal{J}$ such that $Q = P_1 \cap P_2 \cap P_3$. Obviously, the intersection of any two paths is also a path, so Q is a path in T.

If Q contains at least k edges, then C is a k-edge clique.

Suppose Q contains less than k edges, but at least one edge. Due to the connectivity of the intersection $P_1 \cap P_2$ and since $|P_1 \cap P_2| \geq k$, there exists an edge $e_{12} \in P_1 \cap P_2$, $e_{12} \notin Q$, which has common endpoint u with Q. There also exists an edge $e_{23} \in P_2 \cap P_3$, $e_{23} \notin Q$, ($e_{23} \neq e_{12}$ since otherwise e_{23} would be also in Q), such that the edge e_{23} has a common endpoint v with Q. Moreover, $u \neq v$ because P_2 is a path. Finally, there must be an edge $e_{13} \in P_1 \cap P_3$, $e_{13} \notin Q$, ($e_{13} \neq e_{12}$ and $e_{13} \neq e_{23}$ since otherwise e_{13} would be also in Q), such that e_{13} has a common endpoint with Q. If the common endpoint of e_{13} and Q is u, then P_1 is not a path. If the common endpoint of e_{13} and Q is v, then P_3 is not a path. Contradiction!

Otherwise, Q consists of one single vertex q. Suppose q is not an endpoint of the path $P_1 \cap P_2$, meaning that there are two edges in $P_1 \cap P_2$, whose endpoint is q. Since P_3 contains q and $|P_1 \cap P_3| \geq k$, the path P_3 intersects with $P_1 \cap P_2$ on an edge, whose endpoint is q, and therefore Q has more than a single vertex. Hence, q is an endpoint of the path $P_1 \cap P_2$. Symmetrically, q is also an endpoint of the paths $P_1 \cap P_3$ and $P_2 \cap P_3$. Thus, there exist legs E_{12}, E_{13}, E_{23} which intersect only on q, such that $E_{12} \subseteq P_1 \cap P_2$, $E_{13} \subseteq P_1 \cap P_3$, $E_{23} \subseteq P_2 \cap P_3$. Since any other $P \in \mathcal{J}$ must share k edges with each of the P_1, P_2, P_3, then P intersects with exactly two of the three legs E_{12}, E_{13}, E_{23}. Thus, P contains the central vertex q and contains two of the three legs. Therefore, C is a k-claw clique. $\square$

Let C be a maximal clique of a k-EPT graph $G{=}k\text{-}EPT(\mathcal{P})$, and let $\mathcal{J} \subseteq \mathcal{P}$ be the subcollection corresponding to C, whose members intersect pairwise on at least k edges in T. By Theorem 8.4, there exist $P_1, P_2, P_3 \in \mathcal{J}$, such that $\bigcap\{P | P \in \mathcal{J}\}{=}P_1 \cap P_2 \cap P_3$. We say that the triple P_1, P_2, P_3 **strongly defines** C.

Proposition 8.11 *For any triple of vertices u, v, w, the paths P_u, P_v, P_w strongly define at most one maximal clique.*

Proof: Suppose there exist paths P_u, P_v, P_w that strongly define two maximal cliques C and C' corresponding to subcollections $\mathcal{J}$ and $\mathcal{J}'$ of $\mathcal{P}$.

Let $y \in C'$. If C is a k-edge clique, then $\bigcap\{P|P \in \mathcal{J}\}=P_u \cap P_v \cap P_w \subseteq P_y$. Hence, P_y intersects with every path $P \in \mathcal{J}$ on at least k edges, and therefore $y \in C$. If C is a k-claw clique with three legs L_1, L_2, L_3, then every path $P \in \mathcal{J}$ contains two legs of the k-claw with the central vertex $\{q\}=P_1 \cap P_2 \cap P_3$. Without loss of generality, let $L_1 \subseteq P_u \cap P_v$, $L_2 \subseteq P_v \cap P_w$, $L_3 \subseteq P_w \cap P_u$. Since $|P_y \cap P_u| \geq k$, $|P_y \cap P_v| \geq k$ and $|P_y \cap P_w| \geq k$, the path P_y contains two of the legs L_1, L_2, L_3, and therefore $y \in C$. Thus, we have shown $C' \subseteq C$. However, since C' is a maximal clique, $C' = C$. Contradiction! □

Corollary 8.12 *A k-EPT graph G has at most $O(n^3)$ maximal cliques, where n is the number of vertices in G.*

Proof: This follows immediately from Proposition 8.11. □

Corollary 8.13 *The problem of finding a clique of maximum cardinality can be solved in polynomial time for k-EPT graphs.*

Proof: One can enumerate all the maximal cliques in the graph by a standard clique enumeration algorithm [1] and choose the largest. This can be done in polynomial time, since according to Corollary 8.12 there are at most $O(n^3)$ maximal cliques. □

8.4 Recognizing k-EPT graphs

Definition 8.14 *Let C be any subset of the vertices of a graph G. The **branch graph** $B(G/C)$ has a vertex set $V(B)$ consisting of all the vertices of G not in C but adjacent to some member of C, i.e., $V(B) = \bigcup_{v \in C}\{Adj(v) - C\}$. Adjacency in $B(G/C)$ is defined as follows: $(x, y) \in E(B)$ if and only if in G:*

(i) $(x, y) \notin E(G)$,

(ii) x *and* y *have a common neighbor* $u \in C$,

(iii) the sets $Adj(x) \cap C$ *and* $Adj(y) \cap C$ *are not comparable, i.e., there exist* $w, z \in C$ *such that* w *is adjacent to* x *but not to* y*, and* z *is adjacent to* y *but not to* x.

The following generalizes a result of Golumbic and Jamison, who proved the case of $k = 1$ [23].

Theorem 8.15 *Let* $\langle \mathcal{P}, T \rangle$ *be a* k*-EPT representation of* G*, and let* C *be a maximal clique of a graph* G*. If* C *corresponds to a* k*-edge clique in* $\langle \mathcal{P}, T \rangle$*, then the branch graph* $B(G/C)$ *can be 2-colored. If* C *corresponds to a* k*-claw clique in* $\langle \mathcal{P}, T \rangle$*, then the branch graph* $B(G/C)$ *can be 3-colored.*

Proof: Using Proposition 8.10, we consider two cases.

Case 1. Suppose the clique C corresponds to a subcollection of paths that form a k-edge clique $[e_1, \ldots, e_k]$ in $\langle \mathcal{P}, T \rangle$. Let S_1 be the subtree containing e_1, obtained by removing e_k from T. Let S_2 be the subtree containing e_k, obtained by removing e_1 from T.

We color the graph $B(G/C)$ as follows. Let $x \in V(B)$ and let P_x be the corresponding path in $\langle \mathcal{P}, T \rangle$. If P_x is contained in S_i, then we color x with the color i, for $i = 1, 2$, as follows. Since x has a neighbor in the clique, P_x is contained in at least one of S_1, S_2. Hence, x is colored. Since P_x is not in the clique C, P_x contains at most one of e_1, e_k and therefore is contained in at most one of S_1 or S_2. Hence, x is colored with exactly one color. Therefore, the coloring is well-defined.

We now prove that the coloring of $B(G/C)$ is proper. Suppose $(x, y) \in E(B)$ and x, y have the same color, say 1. Let u, w, z be as given in the definition of the branch graph. Obviously, $P_u \cap P_x$ and $P_u \cap P_y$ do not contain each other. Without loss of generality, suppose that P_x is closer to e_k than P_y. The path P_z intersects with P_y and contains both e_1 and e_k. Therefore, P_z contains $P_u \cap P_x$, meaning that P_z and P_x intersect on at least k edges. Contradiction!

Case 2. Suppose the clique C corresponds to a collection $C = \mathcal{P}[T_k^3]$ where T_k^3 is a k-claw with central vertex q and three legs L_1, L_2, L_3 of k edges each. Recall that any path in $\mathcal{P}[T_k^3]$ contains two of the legs L_1, L_2, L_3. Let S_1, S_2, S_3 be the subtrees of T rooted at q, where L_i is contained in S_i. We color the graph $B(G/C)$ as follows.

Let $x \in V(B)$ and let P_x be the corresponding path in $\langle \mathcal{P}, T \rangle$. We color x with color i, for $i = 1, 2, 3$, if any one of the following holds:

- $P_x \subset S_i$,
- $L_i \subseteq P_x$,
- $P_x \subset (S_j \cup S_l)$ and $0 < |L_j \cap P_x| < k$ and $0 < |L_l \cap P_x| < k$ $(j, l \neq i)$.

We first prove that the coloring is well-defined. If $P_x \subset S_i$ or $L_i \subseteq P_x$, for some i, then x is colored with the color i. Otherwise, consider a neighbor $u \in C$ of x and let j and l be the indices such that $(L_j \cup L_l) \subseteq P_u$. Since $|P_x \cap P_u| \geq k$, it follows that $P_x \subset (S_j \cup S_l)$. Since P_x contains neither L_j nor L_l, it is colored with color i $(i \neq j, l)$. Therefore, x is colored with at least one color.

Suppose x is colored with two colors i and j. If $P_x \subset S_i$ then $P_x \nsubseteq S_j$ and $L_j \nsubseteq P_x$ and $|P_x \cap L_l| = 0$, and therefore x is not colored with j. If $L_i \subseteq P_x$ then $P_x \nsubseteq S_j$ and $L_j \nsubseteq P_x$ and $|P_x \cap L_i| \geq k$, and therefore x is not colored with color j. If $P_x \subset (S_j \cup S_l)$ and $0 < |P_x \cap L_j| < k$ and $0 < |P_x \cap L_l| < k$, then $P_x \nsubseteq S_j$ and $L_j \nsubseteq P_x$ and $|P_x \cap L_i| = 0$, and therefore x is not colored with color j. Contradiction! Therefore, P_x is colored with exactly one color and the coloring is well-defined.

We now prove that the coloring of $B(G/C)$ is proper. Suppose $x, y \in V(B)$ have the same color, say 1, and $(x, y) \in E(B)$. Then $(x, y) \notin E(G)$, so $|P_x \cap P_y| < k$. Let u, w, z as given in the definition of the branch graph. We will handle each of the following cases separately.

1. Suppose $P_x \subset S_1$ and $P_y \subset S_1$. Clearly, $P_x \cap P_u$ and $P_y \cap P_u$ do not contain each other, since each has at least k edges. Without loss of generality, suppose that P_x is closer to q than P_y. The path P_z intersects with P_y and contains q, therefore it contains $P_x \cap P_u$. Hence, $|P_z \cap P_x| \geq k$. Contradiction!

2. Suppose $L_1 \subseteq P_x$ and $L_1 \subseteq P_y$. Then $L_1 \subseteq (P_x \cap P_y)$ and therefore $|P_x \cap P_y| \geq k$. Contradiction!

3. Suppose $L_1 \subseteq P_x$ and $P_y \subset S_1$. Then $L_1 \subseteq P_z$, since z is in the clique C and $|P_z \cap P_y| \geq k$. It means that $L_1 \subset (P_z \cap P_x)$, and therefore $|P_z \cap P_x| \geq k$. Contradiction!

4. Suppose $P_x \subset (S_2 \cup S_3)$, $P_y \subset (S_2 \cup S_3)$ and $0 < |P_x \cap L_2|, |P_y \cap L_2|, |P_x \cap L_3|, |P_y \cap L_3| < k$. Both P_x and P_y do not share an edge with S_1. In addition, $L_2, L_3 \subset P_z$, since z is in the clique C and $|P_z \cap P_y| \geq k$. Hence, $|P_z \cap P_x| \geq k$. Contradiction!

5. Suppose $P_x \subset S_1$ and $P_y \subset (S_2 \cup S_3)$ and $0 < |P_y \cap L_2|, |P_y \cap L_3| < k$. Then $P_u \subset (S_2 \cup S_3)$, since u is adjacent to y. Therefore, $|P_u \cap P_x| = 0$. Contradiction!

6. Suppose $L_1 \subseteq P_x$ and $P_y \subset (S_2 \cup S_3)$ and $0 < |P_y \cap L_2|, |P_y \cap L_3| < k$. Then $P_u \subset (S_2 \cup S_3)$, since u is adjacent to y. The path P_x intersects with at most one of L_2, L_3 and on less than k edges. Hence, $|P_u \cap P_x| < k$. Contradiction!

Since all the cases cover all possibilities, the theorem holds. □

Corollary 8.16 *Let C be a maximal clique of a k-EPT graph G, then the branch graph $B(G/C)$ can be 3-colored.*

Theorem 8.17 [23] *Let G be an undirected graph. The following statements are equivalent:*

(i) G is both a VPT graph and a 1-EPT graph.

(ii) G is a VPT graph and for any maximal clique C of G, the branch graph $B(G/C)$ is 3-colorable.

(iii) G has a VPT representation on a degree 3 tree.

(iv) G has a 1-EPT representation on a degree 3 tree.

Corollary 8.18 *Let G be a VPT graph. Then, G is a 1-EPT graph if and only if G is a k-EPT graph.*

Proof: ($\Rightarrow$) If G is a VPT and 1-EPT graph, then by Lemma 7.1, G is a VPT and k-EPT graph.
($\Leftarrow$) If G is a VPT and k-EPT graph, then by Corollary 8.16, G is a VPT graph and for any maximal clique C of G, the branch graph $B(G/C)$ is 3-colorable. Hence, G is a VPT and 1-EPT graph due to Theorem 8.17. $\Box$

Theorem 8.19 *It is an NP-complete problem to decide whether a VPT graph is a k-EPT graph for any fixed $k \geq 1$.*

Proof: We will prove NP-completeness by demonstrating that an arbitrary undirected graph H is 3-colorable if and only if a certain VPT graph $G = (V, E)$ is a k-EPT graph. The decision problem is solved by checking every branch graph for being 3-colorable. We may assume that every vertex of H has degree at least 3 (since vertices with degree ≤ 2 may be repeatedly stripped off without affecting 3-colorability).

Let $1, \ldots, n$ denote the vertices of H. Following [23], we construct T and $\mathcal{P}$ as follows. The tree T has edges (p_i, q_i) and (q_i, r) for $i = 1, \ldots, n$. For each edge (i, j) in H, let $P_{i,j}$ be the path in T from q_i (through r) to q_j, and for each vertex i in H, let Q_i be the edge (p_i, q_i) in T, see example in Fig. 16. We set $\mathcal{P} = \{P_{i,j}\} \bigcup \{Q_i\}$. The paths $\{P_{i,j}\}$ all share central vertex r and correspond to a maximal clique C of the VPT graph $G = VPT(\mathcal{P})$. Furthermore, the branch graph $B(G/C)$ is isomorphic to H. The remaining branch graphs of G over other maximal cliques, one for each i, are just stable sets.

If G is a VPT and k-EPT graph, then H is 3-colorable due to Corollary 8.16. If H is 3-colorable and G is a VPT graph, then G is a VPT and 1-EPT graph due to

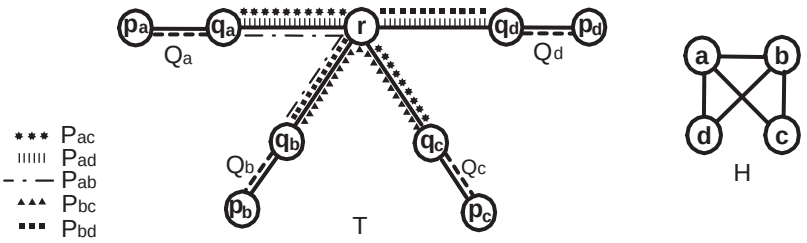

Figure 16: Illustration of proof 8.19

Theorem 8.17, hence G is a k-EPT graph according to Lemma 7.1. Therefore, G is a VPT and k-EPT graph if and only if G is a VPT graph and H is 3-colorable, and hence the theorem follows. □

Corollary 8.20 *Recognizing whether an arbitrary graph is a k-EPT graph is an NP-complete problem.*

It is a simple exercise to modify the proof of Theorem 8.19 to obtain the following.

Proposition 8.21 *It is an NP-complete problem to decide whether a VPT graph is an EPT* graph.*

Corollary 8.22 *Recognizing whether an arbitrary graph is an EPT* graph is an NP-complete problem.*

8.5 Coloring of k-EPT graphs

Definition 8.23 *Let H be a multigraph. The line graph $L(H)$ of H has vertices corresponding to the edges of H with two vertices adjacent in $L(H)$ if the corresponding edges of H share an endpoint.*

Theorem 8.24 [24] *The following statements are equivalent:*
(i) G has a 1-EPT representation, where all paths share a common vertex.
(ii) G is a line graph of a multigraph.

Corollary 8.25 *The problem of finding a minimum coloring of a k-EPT graph is NP-complete.*

Proof: Clearly, if G has a 1-EPT representation, where all paths share a common vertex, then one can obtain a k-EPT representation of G, where all paths share a common vertex. This can be done by adding $k-1$ dummy vertices to each edge in the 1-EPT representation, thus dividing the edge into k edges.

Holyer [38] showed that the minimum coloring problem on line graphs of multigraphs is NP-complete. Therefore, the minimum coloring problem is NP-complete on k-EPT graphs due to Theorem 8.24. □

9 The weakly chordal graphs and the edge intersection graphs of paths in a tree

It is known that coloring of 1-EPT graphs is NP-complete problem. However, the 1-EPT graphs restricted to host trees of vertex degree 3 are precisely the chordal 1-EPT graphs, and therefore can be colored in polynomial time complexity. We prove a new analogous result that weakly chordal 1-EPT graphs are precisely the 1-EPT graphs with host tree restricted to degree 4. This also implies that coloring an edge intersection graph of paths in a degree 4 tree is polynomial.

Definition 9.1 *A two-pair in a graph G is a pair of vertices (x, y), such that every chordless path between x and y contains exactly two edges.*

Theorem 9.2 [36] *A graph G is weakly chordal if and only if every induced subgraph of G either has a two-pair or is a clique.*

Remark 9.3 [24] *A 1-EPT graph contains no $\overline{C}_n$, $n \geq 7$. The 1-EPT representations of graphs $\overline{C}_5$ and $\overline{C}_6$ can be easily found.*

Lemma 9.4 *A 1-EPT graph restricted to degree 4 trees contains no $\overline{C}_5$ and no $\overline{C}_6$.*

Proof: The graph C_5 is uniquely represented by a pie with 5 edges. Since $\overline{C}_5 = C_5$, any 1-EPT representation of $\overline{C}_5$ would require a degree 5 tree.

Let G be a 1-EPT graph, which contains the complement of the chordless 6-cycle $[x_1, x_2, \ldots, x_6]$. In this case, $C' = [x_1, x_4, x_2, x_5]$ forms a chordless 4-cycle in G. Suppose $\langle \mathcal{P}, T \rangle$ is a 1-EPT representation of G on a degree 4 tree. The cycle C' corresponds to a pie, where paths P_1 and P_4 are on leg L_1, P_4 and P_2 are on leg L_2, P_2 and P_5 are on leg L_3, P_5 and P_1 are on leg L_4. The path P_6 intersects with P_2 and P_4, but does not intersect with P_5 and P_1, hence P_6 is on leg L_2. The path P_3 does not intersect with P_2 and P_4 and therefore is not on L_1, L_2, L_3. However, the path P_3 intersects with P_6. Contradiction! □

Definition 9.5 *A subset S of vertices of a connected graph G is called a* (x, y)-separator *if x and y are in different connected components of G_{V-S}.*

We now present the main result of this section.

Theorem 9.6 *A graph G is weakly chordal and 1-EPT graph if and only if G has a 1-EPT representation on a degree 4 tree.*

Proof: ($\Leftarrow$) Let $\langle \mathcal{P}, T \rangle$ be a 1-EPT representation of a graph G, where T is of degree 4. We must show that G is weakly chordal. By Theorem 8.6, if G contains a chordless cycle C_n ($n \geq 5$), then $\langle \mathcal{P}, T \rangle$ contains a pie with n edges. Since T is restricted to degree 4, this is impossible. Therefore, G has no C_n ($n \geq 5$).

Moreover, G contains no $\overline{C}_n$, $n \geq 7$ due to Remark 9.3, and G contains no $\overline{C}_5$ or $\overline{C}_6$ due to Lemma 9.4. It now follows that G is weakly chordal.

($\Rightarrow$) Let $\langle \mathcal{P}, T \rangle$ be a 1-EPT representation of a weakly chordal graph G, where T has maximal degree $d > 4$. We iteratively apply the construction described in the proof of the following Lemma 9.8 to obtain a representation $< \mathcal{P}', T' >$ of G, where T' has a maximal degree 4. This will prove the theorem. □

Remark 9.7 *We may assume that the graph G is connected, since otherwise every connected component can be handled independently. Therefore, we also assume that*

there is no empty edge on the host tree, in other words every edge is contained in some path.

Lemma 9.8 *Let $\langle \mathcal{P}, T \rangle$ be a 1-EPT representation of a weakly chordal graph G, where T has a maximal degree $d > 4$. Then there exists a 1-EPT representation $< \mathcal{P}', T' >$ of G with maximal degree d', where $d > d' \geq 4$.*

Proof: The proof will be by induction on the number of vertices of degree d. Let u be a vertex of degree d in T with neighbors $v_1, \ldots, v_d$. Let $\mathcal{P}_U$ be the collection of paths of $\mathcal{P}$ that contain at least one of the edges $(v_1, u), \ldots, (v_d, u)$ in T. Let G_U be the induced subgraph of G, such that every vertex in G_U corresponds to a path in $\mathcal{P}_U$. By the hereditary property, the induced subgraph G_U is also a 1-EPT graph. Furthermore, since G is connected and T is a tree, G_U is connected.

We may assume that every path in $\mathcal{P}_U$ contains two of the edges $(v_1, u), \ldots, (v_d, u)$ in T by the following argument. Suppose there exists a path in $\mathcal{P}_U$ that contains only one of the edges $(v_1, u), \ldots, (v_d, u)$ in T. Without loss of generality, assume this edge is (v_1, u), and let $\mathcal{P}_I$ be the collection of paths in $\mathcal{P}_U$ that contain only the edge (v_1, u). We will change the representation as follows: We add a dummy vertex w on the edge (v_1, u). In every path $P_i \in \mathcal{P}_I$, we replace the edge (v_1, u) by the edge (v_1, w), thus making w the endpoint of P_i, and in all the other paths we replace (v_1, u) by the two edges (v_1, w) and (w, u). Any two paths that shared the common edge (v_1, u) before, i.e., correspond to adjacent vertices, share the common edge (v_1, w) now. Any two paths that did not share any edge, do not share a common edge now. Therefore, the obtained tree is a 1-EPT representation of G. Thus, we may repeat this process for edges $(v_2, u), \ldots, (v_d, u)$ as needed.

According to Theorem 9.2, the subgraph G_U is either a clique or has a *two-pair*. We consider these two cases independently.

Case 1. The subgraph G_U is a clique. Every clique in G corresponds to either a 1-edge or a 1-claw in a 1-EPT representation of G according to Lemma 8.10. The clique G_U can not correspond to a claw, since otherwise at least one edge of $(v_1, u), \ldots, (v_d, u)$

would be empty (see Remark 9.7). Thus, the clique G_U corresponds to a 1-edge clique, and all the paths in $\mathcal{P}_U$ share a common edge, suppose (v_1, u). We will change the representation as follows: We add a dummy vertex w on the edge (v_1, u). Now we arbitrarily remove the edge (v_2, u) and add the edge (v_2, w) as shown in Fig. 17 (a) and (b). In all the paths in $\mathcal{P}_U$ that contain the edge (v_2, u), we replace (v_2, u) with the new edge (v_2, w). All other paths remain unchanged.

Now all the paths in $\mathcal{P}_U$ still share a common edge, namely (v_1, w), and u has its degree reduced to $d' = d - 1$. The degree of w is 3, and the degree of all the other vertices remain unchanged. Hence, the obtained tree is a 1-EPT representation of G with fewer vertices of degree d.

Case 2. The subgraph G_U has a two-pair (x, y). By definition, the corresponding paths P_x, P_y do not share a common edge in T. Without loss of generality, suppose that P_x contains the edges $(v_1, u), (v_2, u)$ and P_y contains the edges $(v_3, u), (v_4, u)$. Let G_S be the induced subgraph of G that consists of all common neighbors of x and y, and let $\mathcal{P}_S$ be the collection of paths that correspond to G_S. We note that since (x, y) is a *two-pair*, S is an (x, y)-separator. Hence, G_{U-S} has two connected components G_X and G_Y, such that $x \in G_X$ and $y \in G_Y$.

We handle separately the case that G_S is a clique.

Case 2.1. G_S is a clique. The components G_X and G_Y induce partition of the edges of the star $\{(v_1, u), \ldots, (v_d, u)\}$ as follows:

Color the edges (v_1, u), (v_2, u) red and color the edges (v_3, u),(v_4, u) blue. Color the edge (v_i, u) red if it is on a path P in $\mathcal{P}_U \backslash \mathcal{P}_S$, where P shares a common edge with a red edge. Color the edge (v_i, u) blue if it is on a path P in $\mathcal{P}_U \backslash \mathcal{P}_S$, where P shares a common edge with a blue edge. We proceed until no further coloring is possible. Since S is an (x, y)-separator, every path in $\mathcal{P}_U \backslash \mathcal{P}_S$ contains either two red edges or two blue edges or two uncolored edges.

We will change the representation as follows: We split the vertex u into two vertices u' and u'', and add the edge (u', u'') to the representation. Replace each red edge (v_i, u) by (v_i, u'). Replace each blue or uncolored edge (v_i, u) by (v_i, u''). All the

paths of G_{U-S} containing red edges will pass through u' and will not contain the edge (u', u''). All the paths of G_{U-S} containing blue edges or uncolored edges will pass through u'' and will not contain the edge (u', u''). All the paths in $\mathcal{P}_S$ will contain the edge (u', u'') and have a common edge with P_x and with P_y. See Fig. 17 (c) and (d).

The degree of each one of u' and u'' is at most $d-1$. The degree of all the other vertices remain unchanged. Since all the paths in $\mathcal{P}_S$ now share a common edge, and all other common edges remain, the obtained tree is a 1-EPT representation of G with fewer vertices of degree d.

Case 2.2. G_S is not a clique. Since $d > 4$ and G_U is connected, there exists at least one path in $\mathcal{P}_U$ that contains only one of the edges (v_1, u), (v_2, u) (v_3, u), (v_4, u). Without loss of generality, let $\mathcal{P}_I$ be the non-empty collection of paths that contain the edge (v_1, u), but do not contain the edges (v_2, u), (v_3, u), (v_4, u). Clearly, $\mathcal{P}_I$ corresponds to a clique in G.

We define a partition of the edges of the star $\{(v_1, u), \ldots, (v_d, u)\}$ as follows:

The edge (v_1, u) is colored black. Color the edge (v_i, u), $(i > 1)$ red if it is on a path $P \in \mathcal{P}_I$. Note that so far (v_2, u), (v_3, u), (v_4, u) are uncolored. Now propagate the coloring as follows. Color the edge (v_i, u) $(i > 1)$ red if it is on a path $P \in \mathcal{P}_U$, where P shares a common edge with a red edge. We proceed until no further coloring is possible.

Clearly, every path in $\mathcal{P}_U \setminus \mathcal{P}_I$ has either two red edges or no red edges. We will now prove that none of the edges $(v_2, u), (v_3, u), (v_4, u)$ is colored. By contradiction, suppose the edge (v_3, u) is the first edge among $(v_2, u), (v_3, u), (v_4, u)$ that is colored, assume it was colored because it is contained in a path P_w. Then there exists a shortest path $[x, w_1, \ldots, w_m, w, y]$ of length strictly greater than 3 in G,$m \geq 1$ where $P_{w_1} \in \mathcal{P}_I$, $w_1, \ldots, w_m, w \notin G_S$ and none of $P_{w_1}, \ldots, P_{w_m}$ contains any of the edges $(v_2, u), (v_3, u), (v_4, u)$. This contradicts the fact that (x, y) is a 2-pair. By the same arguments, the edge (v_4, u) also can not be the first colored edge among $(v_2, u), (v_3, u), (v_4, u)$. Thus, suppose the edge (v_2, u) is the first edge

among $(v_2, u), (v_3, u), (v_4, u)$ that is colored, and assume it was colored because it is contained in a path P_w. Then there exists a shortest path $[x, w_1, \ldots, w_m, w]$ in G, $m \geq 1$, where $P_{w_1} \in \mathcal{P}_I$, $w_1, \ldots, w_m, w \notin G_S$ and none of $P_{w_1}, \ldots, P_{w_m}$ contains any of the edges $(v_2, u), (v_3, u), (v_4, u)$. Since G_S is not a clique, there exist two non-adjacent vertices $z, z' \in G_S$. Without loss of generality, suppose that z contains the edge (v_1, u) and z' contains the edge (v_2, u). Therefore, there exists a chordless cycle $[y, z, w_1, \ldots, w_m, w, z', y]$ in G of length greater than 4. This contradicts the fact that G is a weakly chordal graph. Hence, (v_2, u), (v_3, u), (v_4, u) remain uncolored.

Now we are ready to change the representation as follows: We add a dummy vertex w on the edge (v_1, u). Replace every red edge $(v_i, u), i \geq 5$ by a new edge (v_i, w) in all the paths containing it. The uncolored edges remain unchanged. Specifically, (1) paths P_x, P_y and all paths in $\mathcal{P}_S$ are unchanged, (2) paths in $\mathcal{P}_I$ have edges (v_1, u), (v_i, u) replaced by (v_1, w), (v_i, w), (3) paths with two red edges (v_i, u), (v_j, u) are replaced by (v_i, w), (v_j, w), and (4) other paths with no red edges remain unchanged. It is straightforward to verify that this is a 1-EPT representation of G. All the paths in $\mathcal{P}_I$ now share a common edge (v_1, w). Every two paths that shared a common red edge (v_i, u) before, now share the common edge (v_i, w). All other common edges remain. See Fig. 17 (e) and (f).

The degree of the vertex u is now at most $d - 1$, because $\mathcal{P}_I$ is non-empty and at least one edge is colored. The degree of the vertex w is at most $d - 2$, because the edges $(v_2, u), (v_3, u), (v_4, u)$ are uncolored. Therefore, the obtained tree is a 1-EPT representation of G with fewer vertices of degree d. This completes Case 2.2.

We have shown that in all cases there exists a construction that reduces the number of vertices of degree d in $\langle \mathcal{P}, T \rangle$ and proves the Lemma and thus also Theorem 9.6. □

By using essentially the same arguments as in the proof of Lemma 9.8, we can obtain an alternate proof of the follows:

Corollary 9.9 [53] *A chordal graph G is 1-EPT if and only if G has a 1-EPT rep-*

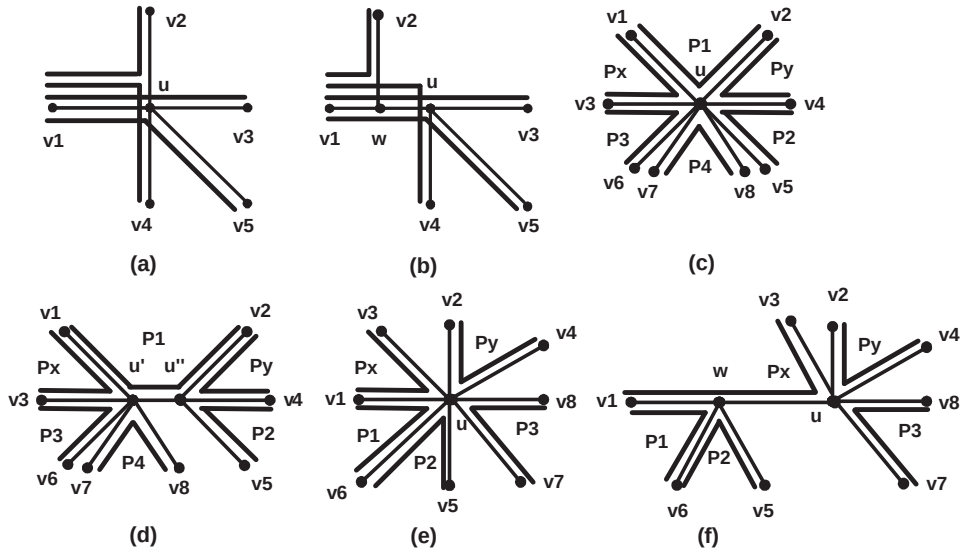

Figure 17: Illustration of proof 9.8

resentation on a degree 3 tree.

Remark 9.10 *The graph H shown in Fig. 19 is weakly chordal, but is not a 1-EPT graph.*

10 The complete hierarchy of intersection graphs of paths in a tree

Definition 10.1 *An (h, s, t)-representation of G is called **orthodox** if the endpoints of every subtree are leaves in T, and for every $x, y \in V(G)$, the subtrees P_x, P_y share a common leaf if and only if $(x, y) \in E(G)$. It means that the subtrees P_x, P_y share a common leaf if and only if P_x, P_y share t vertices. A graph G is called an orthodox-$[h, s, t]$ graph if G has an orthodox (h, s, t)-representation.*

The terminology of orthodox representation is used by Jamison and Mulder in [39, 40], who cite an earlier result of McMorris and Scheinerman [46], namely, $(i) \Leftrightarrow (ii) \Leftrightarrow (iii)$ in Theorem 10.2 below, and prove in [39] the remaining equalities.

Theorem 10.2 *Let G be a graph, then the following statements are equivalent:*

(i) G is chordal,

(ii) G has a $(3,3,1)$-representation,

(iii) G has an orthodox $(3,3,1)$-representation,

(iv) G has a $(3,3,2)$-representation,

(v) G has an orthodox $(3,3,2)$-representation.

In this section we first investigate orthodox representations proving two equivalence theorems in section 10.1. Then we show the relationship between variations of $[h,2,t]$ by three main parameters: h, t and whether the graph has an orthodox representation. We give the complete hierarchy of relationships between the classes of weakly chordal, chordal, $[h,2,t]$ and orthodox-$[h,2,t]$ graphs for varied values of h and t. Specifically, we demonstrate the results illustrated in the complete hierarchy shown in Figure 19 in section 10.2.

10.1 Orthodox representations

In this section, motivated by Theorem 10.2, we investigate the orthodox representations of VPT, k-EPT and EPT* graphs. Table 2 summarizes the relationships between orthodox-$[h,2,t]$ graphs.

Table 2: Results on orthodox-$[h,2,t]$ graphs

Result	Where
chordal $\equiv [3,3,1] \equiv$ orthodox-$[3,3,1] \equiv [3,3,2] \equiv$ orthodox-$[3,3,2]$	Theorem 10.2
orthodox-$[\infty,2,1] \equiv$ orthodox-$[3,2,1] \equiv$ orthodox-$[3,2,2]$	Theorem 10.6
orthodox-$[\infty,2,1] \subset\neq$ orthodox-$[\infty,2,2]$	Lemma 10.5
orthodox 1-EPT $\equiv$ orthodox k-EPT, $\forall k > 2 \equiv$ orthodox EPT*	Theorem 10.3
chordal 1-EPT $\equiv$ 1-EPT degree 3 $\equiv$ VPT degree 3 $\equiv$ 1-EPT $\cap$ VPT	[23]
weakly chordal $[\infty,2,2] \equiv [4,2,2]$	Theorem 9.6

We prove that all the families of orthodox k-EPT graphs coincide.

Theorem 10.3 *Let G be a graph, the following statements are equivalent:*

(i) G has an orthodox 1-EPT representation,

(ii) G has an orthodox k-EPT representation for any fixed $k > 1$,

(iii) G has an orthodox EPT representation.*

Proof: $(i) \Rightarrow (ii)$ Jamison and Mulder proved in [40][1] that if G has an orthodox l-EPT representation, then G has an orthodox $(l+1)$-EPT representation for every $l > 0$, by adding a pendant edge to every leaf of the host tree and extending each path touching a leaf to include its new edge. Hence, this implication of the proof follows by induction.

$(ii) \Rightarrow (iii)$ If G has an orthodox k-EPT representation for some k, then by definition, G has an orthodox EPT* representation.

$(iii) \Rightarrow (i)$ Suppose G has an orthodox EPT* representation, then by definition, G has an orthodox k-EPT representation $\langle \mathcal{P}, T \rangle$ for some fixed k with leaves $y_1, \ldots, y_m$. We will construct an orthodox 1-EPT representation $\langle \mathcal{P}', T' \rangle$ of G. The tree T' is a T_1^m star consisting of a center vertex x and leaves $y'_1, \ldots, y'_m$. For every path P with the endpoints y_i,y_j in T, the corresponding path P' in T' has the endpoints y'_i, y'_j. Clearly, the representation $\langle \mathcal{P}', T' \rangle$ is leaf-generated. Two paths share a leaf in $\langle \mathcal{P}, T \rangle$ if and only if the corresponding paths share a leaf in $\langle \mathcal{P}', T' \rangle$ if and only if the corresponding paths share an edge in $\langle \mathcal{P}', T' \rangle$. Therefore, two paths share a leaf in $\langle \mathcal{P}', T' \rangle$ if and only if the corresponding vertices are adjacent in G. Thus, $\langle \mathcal{P}', T' \rangle$ is an orthodox 1-EPT representation. □

Remark 10.4 *In any tree T, two paths share a leaf in T if and only if they share a pendant edge.*

The Remark implies the following simple result.

Lemma 10.5 *Let $\langle \mathcal{P}, T \rangle$ be an orthodox VPT representation of a graph G, then $\langle \mathcal{P}, T \rangle$ is also an orthodox 1-EPT representation of G.*

[1] In fact, they show a stronger result: orthodox-$[h, s, p] \subseteq$ orthodox-$[h, s, p+1]$, $p \geq 1$.

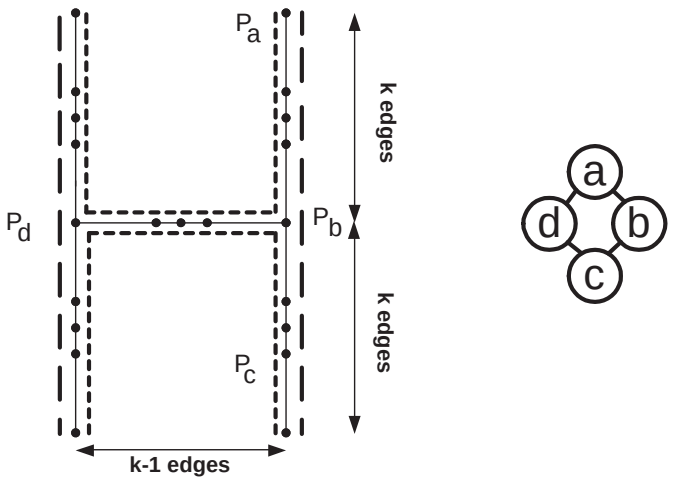

Figure 18: The graph C_4 and a k-EPT representation of C_4 for $k \geq 1$.

Proof: Two vertices are adjacent in G if and only if the corresponding paths share a leaf vertex in $\langle \mathcal{P}, T \rangle$. By Remark 10.4, the paths share a leaf if and only if they share a pendant edge. Therefore, $\langle \mathcal{P}, T \rangle$ is also an orthodox 1-EPT representation. □

However, the graph C_4 has an orthodox 1-EPT representation shown in Figure 18, but is not a VPT graph. Thus, the converse of Lemma 10.5 does not hold, except in the case of degree 3 trees, as shown in the following theorem.

Theorem 10.6 *Let G be a graph, the following statements are equivalent:*
(i) G has an orthodox VPT representation,
(ii) G has an orthodox VPT representation restricted to degree 3 trees,
(iii) G has an orthodox 1-EPT representation restricted to degree 3 trees.

Proof: $(i) \Rightarrow (ii)$ Let $\langle \mathcal{P}, T \rangle$ be an orthodox VPT representation of G. Let t be a vertex in T with degree $m, m > 3$, where $x_1, x_2, \ldots, x_m$ are the neighbors of t in T. Let $\mathcal{P}[t]$ be the collection of paths in $\langle \mathcal{P}, T \rangle$ that contain vertex t. Since t is not a leaf in T and $\langle \mathcal{P}, T \rangle$ is an orthodox VPT representation, every path in $\mathcal{P}[t]$ contains two of $x_1, x_2, \ldots, x_m$. Clearly $\mathcal{P}[t]$ corresponds to a clique in G. Let $P \in \mathcal{P}[t]$ be a path that contains the vertices x_i, x_j. Since the representation is orthodox and $\mathcal{P}[t]$ corresponds to a clique in G, all the other paths in $\mathcal{P}[t]$ either contain x_i or x_j. Now, we add a new

node t_{ij} to T and replace the edges (t, x_i),(t, x_j) by the edges (t_{ij}, x_i),(t_{ij}, x_j),(t, t_{ij}) in T and in the paths $\mathcal{P}[t]$ respectively. Since every path either contains x_i or x_j, the collection $\mathcal{P}[t]$ still corresponds to a clique and the representation remains orthodox. However, the degree of the vertex t now is $m - 1$. We continue in the same fashion until the vertex t has degree 3. We iteratively apply this procedure on every vertex in T until we obtain a tree with maximum degree 3.

$(ii) \Rightarrow (i)$ This follows directly.

$(ii) \Rightarrow (iii)$ By Lemma 10.5, if G has an orthodox VPT representation $\langle \mathcal{P}, T \rangle$ on a degree 3 tree T, then $\langle \mathcal{P}, T \rangle$ is also an orthodox 1-EPT representation of G on a degree 3 tree T.

$(iii) \Rightarrow (ii)$ Let $\langle \mathcal{P}, T \rangle$ be an orthodox 1-EPT representation of G on a degree 3 tree T. By definition, two vertices are adjacent in G if and only if the corresponding paths share a leaf vertex in $\langle \mathcal{P}, T \rangle$. It remains to prove that two paths that correspond to non-adjacent vertices do not share a non-leaf vertex in $\langle \mathcal{P}, T \rangle$. Suppose the paths P_x and P_y share a non-leaf vertex z, but do not share an edge in $\langle \mathcal{P}, T \rangle$. Since T is a degree 3 tree, the vertex z must be an endpoint vertex of either P_x or P_y or both. This contradicts the fact that $\langle \mathcal{P}, T \rangle$ is a leaf-generated representation. □

10.2 The hierarchy

We say that a hierarchy is **complete**, when all containment relationships are given. That is, (1) a downward edge from class A to class B indicates that class A contains class B, (2) the lack of a hierarchical (containment) relation indicates that the classes are incomparable, (3) classes that appear in the same box are equivalent, (4) an example appearing along the edge between two classes is a separating example for those classes.

Theorem 10.7 *The hierarchy represented in Figure 19 is complete.*

Proof:

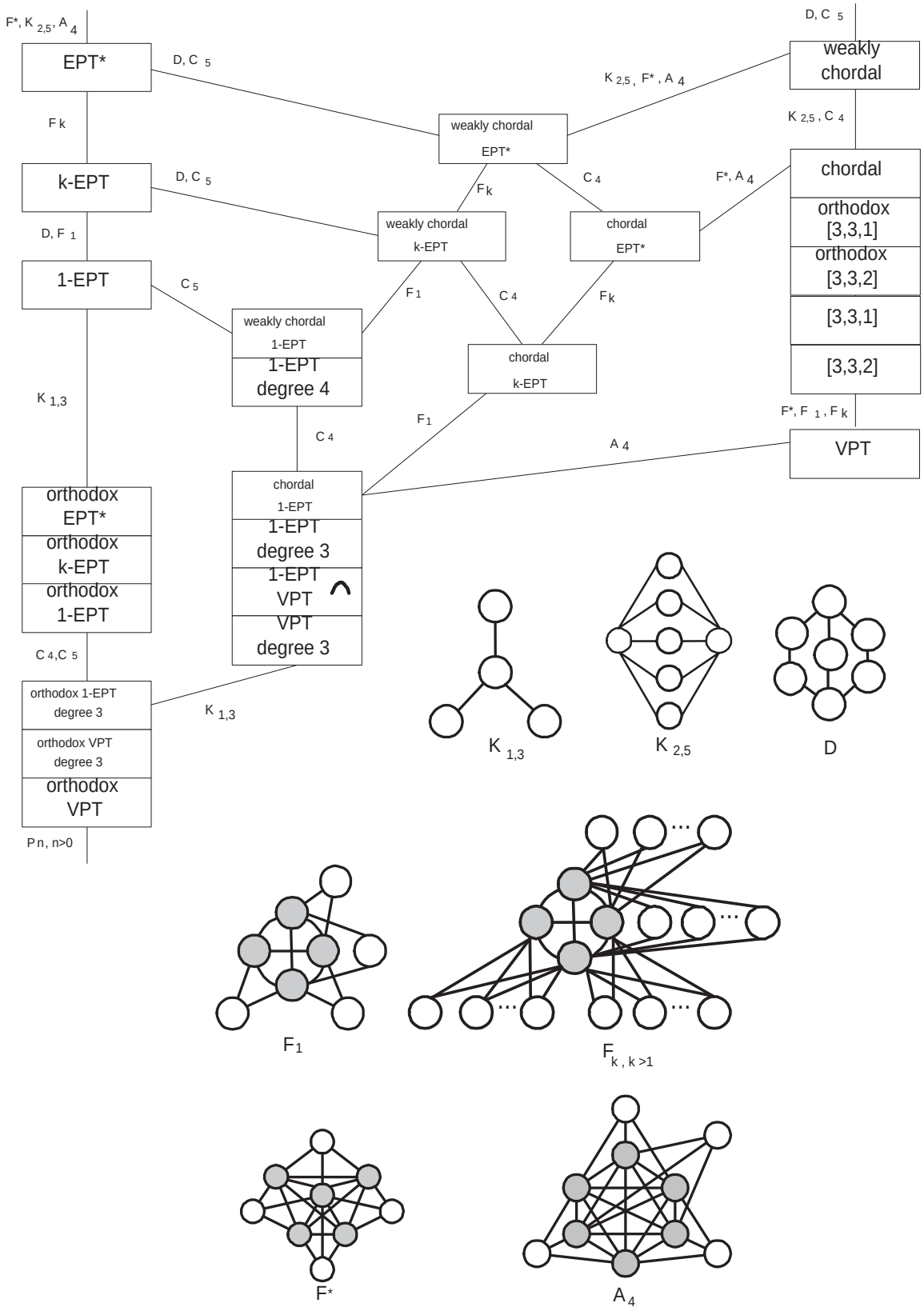

Figure 19: The complete hierarchy of k-EPT and other related families of graphs. The class k-EPT means for an arbitrary fixed $k \geq 2$.

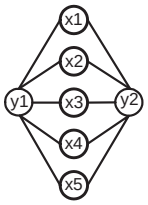

Figure 20: The $K_{2,5}$ graph.

1. **Equivalences.**

   All the equivalence relations are summarized in Table 2.

2. **Containments.**

   We have already remarked that the VPT graphs are chordal. Having already shown the equivalence of classes in the same box, all the other containment relations in the hierarchy are trivial by their definitions.

3. **Separation examples.**

   The following are the separation examples as shown in Figure 19.

   The graph $K_{1,3}$.

   The graph $K_{1,3}$ obviously has a VPT representation on a degree 3 tree (in fact, on a degree 2 tree) and therefore by [23] is also a 1-EPT graph. However, $K_{1,3}$ does not have an orthodox 1-EPT representation, because the path corresponding to the central vertex can not share a pendant edge with three different paths that do not intersect one with each other.

   The graph $K_{2,5}$.

   Clearly, the graph $K_{2,5}$ is weakly chordal but not chordal.

   We will now prove that the graph $K_{2,5}$ is not a k-EPT graph for any $k \geq 1$, that is, $K_{2,5} \notin EPT^*$.

   Lemma 10.8 *The graph $K_{2,5}$ in Fig. 20 is not a k-EPT graph for any $k \geq 1$, that is, $K_{2,5} \notin EPT^*$.*

Proof: Suppose that the graph $K_{2,5}$ does have a k-EPT representation for some $k \geq 2$. We will handle the following cases separately.

Case 1: Suppose $P_{x_i} \cap P_{x_j} \neq \emptyset$, for every $i, j \in \{1, \ldots, 5\}$. Then, by Theorem 8.2, there exists a path P such that $\bigcap_{i=1,\ldots,5} P_{x_i} = P$. Obviously $|P| < k$, since otherwise $\{x_1, \ldots, x_5\}$ would form a clique.

Neither P_{y_1} nor P_{y_2} is contained in P, since otherwise $|P_{y_1}| < k$ or $|P_{y_2}| < k$. Therefore, each of P_{y_1}, P_{y_2} intersects with at least one subtree obtained by removing P. Each of P_{y_1}, P_{y_2} intersects with more than one subtree obtained by removing P, since otherwise every P_{x_i} contains P and intersects with this subtree and there is an edge common to all P_{x_i} in this subtree. If P_{y_1}, P_{y_2} intersect on the same subtree, then every P_{x_i} contains P and intersects with this subtree, starting to intersect with P_{y_1} and P_{y_2} at the same point and continue intersecting with both of them on k edges. This is a contradiction, since $|P_{y_1} \cap P_{y_2}| < k$.

Thus, each one of P_{y_1} and P_{y_2} intersects with two different subtrees. Every P_{x_i} intersects with P_{y_1} and P_{y_2} on k edges. Thus, every P_{x_i} intersects with P_{y_1} on some subtree, contains P and intersects with P_{y_2} on another subtree. If two of P_{x_i} intersect with P_{y_1} or P_{y_2} on the same subtree, then they start to intersect with it at the same point and continue intersecting on at least k edges, and therefore intersecting each other on k edges. Hence, every P_{x_i} intersects with P_{y_1} and P_{y_2} on a different subtree. This is a contradiction, since the five paths $\{P_{x_i}\}_{i=1,\ldots,5}$ cannot intersect on different subtrees.

Case 2: Suppose, without loss of generality, $P_{x_1} \cap P_{x_2} = \emptyset$. Then there exists a unique simple path Q from P_{x_1} to P_{x_2}, since T is a tree.

Both P_{y_1} and P_{y_2} intersect with P_{x_1} and P_{x_2}. There is only one unique path Q from P_{x_1} to P_{x_2}, therefore both P_{y_1} and P_{y_2} contain Q and each intersect with more than one subtree obtained by removing Q. Moreover, $|Q| < k$.

Claim 1 *The paths P_{y_1} and P_{y_2} intersect on different subtrees obtained by removing Q.*

Proof of Claim 1: Let v_1 be the vertex on P_{x_1}, which intersects with Q, and let v_2 be the vertex on P_{x_2}, which intersects with Q. Both P_{y_1} and P_{y_2} intersect on at least k edges with P_{x_1} and both contain Q. Since $|P_{y_1} \cap P_{y_2}| < k$, $P_{y_1} \cap P_{x_1} \cap P_{y_2} = v_1$. Symmetrically, $P_{y_1} \cap P_{x_2} \cap P_{y_2} = v_2$. Therefore, P_{y_1} and P_{y_2} intersect in different subtrees obtained by removing Q. This proves Claim 1.

Claim 2 *Each of the paths P_{x_3}, P_{x_4}, P_{x_5} intersect with Q on at least one vertex.*

Proof of Claim 2: Suppose P_{x_3} does not intersect with Q. Then P_{x_3} is contained in exactly one subtree S, obtained by removing Q. The paths P_{y_1} and P_{y_2} intersect with P_{x_3} and therefore both intersect with S. This contradicts Claim 1 and therefore P_{x_3} intersects with Q on at least one vertex. The same argument follows for paths P_{x_4} and P_{x_5}. This proves Claim 2.

Claim 3 *The paths P_{x_3}, P_{x_4}, P_{x_5} contain Q.*

Proof of Claim 3: Suppose P_{x_3} does not contain Q, but intersects on at least one vertex with Q due to Claim 2. Since both P_{y_1} and P_{y_2} intersect on at least k edges with P_{x_3}, P_{x_3} intersects with P_{y_1} on exactly one subtree S_1 obtained by removing Q, and P_{x_3} intersects with P_{y_2} on exactly one subtree S_2 obtained by removing Q. Due to Claim 1, $S_1 \neq S_2$. Since P_{y_1} and P_{y_2} contain Q, the subtrees S_1 and S_2 intersect with Q on exactly one vertex v. Therefore, $P_{x_3} \cap Q = \{v\}$.

Suppose, without loss of generality, P_{x_1} contains v. Each of S_1 and S_2 intersects with P_{x_1} on at least k edges. Therefore, P_{x_3} intersects with P_{x_1} on at least k edges. Contradiction!

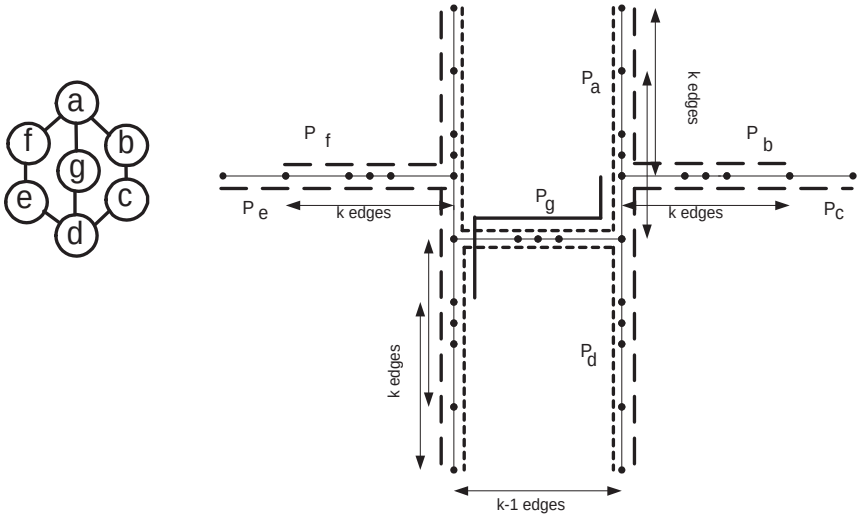

Figure 21: The graph D and a k-EPT representation of D for $k \geq 2$.

The same argument follows for paths P_{x_4} and P_{x_5}. This proves Claim 3.

Let S_1^1, S_1^2 be subtrees intersecting with P_{y_1} after removing Q. Let S_2^1, S_2^2 be subtrees intersecting with P_{y_2} after removing Q.

Each of P_{x_3}, P_{x_4}, P_{x_5} intersect on either S_1^1 or S_1^2 and on either S_2^1 or S_2^2 due to Claim 3. Two of P_{x_3}, P_{x_4}, P_{x_5} cannot intersect on the same subtree among S_2^1, S_1^2, S_2^1, S_2^2, since otherwise they start to intersect with P_{y_1} or P_{y_2} at the same point and continue intersecting on k edges, thus intersecting each other on at least k edges. This is a contradiction, since three paths cannot intersect with these four subtrees.

Thus, the graph does not have a k-EPT representation for $k \geq 2$. Finally, Lemma 7.1 now implies that the graph is not 1-EPT. □

The graph D.

The graph D is not 1-EPT as proved in [23], but it has a k-EPT representation on a degree 3 tree, for $k \geq 2$, as illustrated in Figure 21. Moreover, the graph D contains a chordless cycle of size 6, and therefore is not weakly chordal.

The graph $A_d, d \geq 4$.

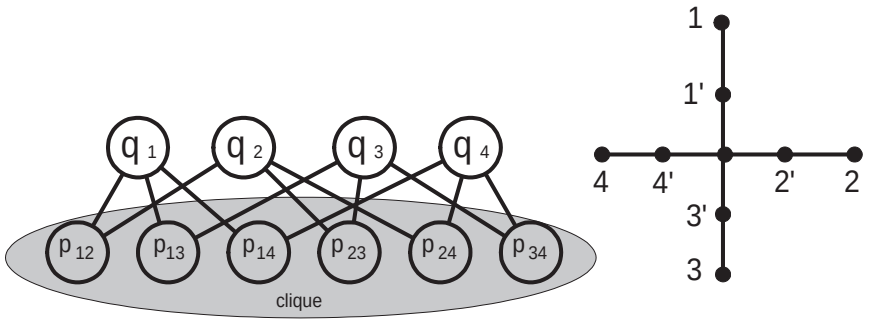

Figure 22: The graph A_4, and a VPT representation of A_4, where P_{ij} is the path from i to j and Q_i is the edge (i, i').

Let A_d be the graph, such that $V(A_d)$=$\{p_{ij}|1 \leq i < j \leq d\} \cup \{q_i|1 \leq i \leq d\}$, where $\{p_{ij}\}$ is a clique and $(p_{ij}, q_l) \in E(A_d)$ if and only if $l = i$ or $l = j$.

The graph A_d is a VPT graph: a VPT representation of A_d can be obtained from a tree T_2^d with legs $L_1, \ldots, L_d$, such that $P_{ij} = L_i \cup L_j$ and Q_i is the pendant edge of L_i, and where the path P_{ij} corresponds to the vertex p_{ij} and the path Q_i corresponds to the vertex q_i. For example, the graph A_4 and a VPT representation of A_4 are shown in Figure 22. Therefore, A_d is a VPT graph and hence is a chordal graph.

The branch graph $B(A_d/C)$, $C = \{p_{ij}|\forall i, j\}$ is the complete graph on d vertices $q_1, \ldots, q_d$ and therefore A_d is not a k-EPT graph for any $k \geq 1$ by Theorem 8.15.

The graphs C_4, C_5.

By definition the graph C_4 is not chordal, but is weakly chordal and the graph C_5 is neither chordal nor weakly chordal. By Corollary 8.7 if every 1-EPT representation of G is restricted to degree 3, then G has no $C_n, n > 3$.

The graphs P_n.

An orthodox VPT representation of P_n is shown in Figure 23.

Now we will prove the following lemma, which will be used in further separation examples.

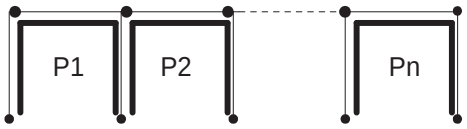

Figure 23: An orthodox VPT representation of the graph P_n

Lemma 10.9 *The maximal clique $K_4 = \{1, 2, 3, 4\}$ of F_1 corresponds to a k-claw clique in any k-EPT representation for $k \geq 1$. In addition, there must be a path among $\{P_a, P_b, P_c, P_d\}$ that contains the central vertex of the k-claw corresponding to K_4 and intersects with two legs of that k-claw.*

Proof: In the branch graph $B(F_1/K_4)$ contains a triangle $\{a, b, c\}$ and therefore can not be 2-colored. Hence K_4 must be a k-claw clique in any k-EPT representation due to Theorem 8.15.

Suppose none of the paths P_a, P_b, P_c, P_d contains the central vertex of the k-claw corresponding to K_4. Let S be the set of three subtrees obtained after removing the central vertex of the k-claw. Then each of P_a, P_b, P_c, P_d is contained in exactly one subtree in S.

Suppose there exists a subtree in S that intersects with both P_a and P_c. Then the paths P_1, P_2, P_3 also intersect with that subtree. Without loss of generality, suppose that $P_a \cap P_2$ is on the path from $P_c \cap P_2$ to the central vertex of the k-claw. This is a contradiction, since P_a intersects with P_3 on at least k edges. By the same arguments we may conclude that every two vertices that are adjacent in the branch graph $B(F_1/K_4)$ do not intersect with the same subtree in S. This is a contradiction, since there are only three subtrees in S and $\{a, b, c, d\}$ form a clique in $B(F_1/K_4)$. $\square$

The graph F_1.

The k-EPT representation, $k > 1$, of the graph F_1 is shown in Figure 24.

Suppose there exists a 1-EPT representation of F_1. According to Lemma 10.9,

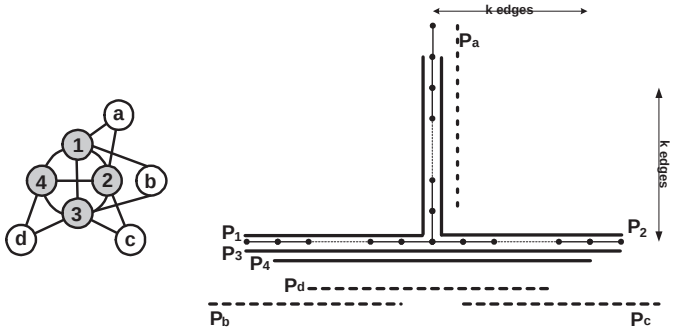

Figure 24: The graph F_1 and a k-EPT representation of F_1, $k > 1$.

there must be a path among $\{P_a, P_b, P_c, P_d\}$ that intersects with two legs of the 1-claw corresponding to K_4 and contains the central vertex of that 1-claw. Since every leg of that 1-claw has exactly one edge, that path contains two legs of the 1-claw and therefore the corresponding vertex is adjacent to all vertices in K_4. This is a contradiction and therefore F_1 is not a 1-EPT graph.

Suppose there exists a VPT representation $\langle \mathcal{P}, T \rangle$ of F_1. Then the paths P_1, P_2, P_3, P_4 share a common vertex u on T due to Theorem 8.2. Let S be the set of subtree obtained by removing the vertex u from T. Suppose P_a and P_c intersect with a same subtree in S, then P_1, P_2, P_3 also intersect with that subtree. Without loss of generality, suppose $P_a \cap P_2$ is on the path from $P_c \cap P_2$ to the central vertex u in T. This is a contradiction, since P_a intersects with P_3. By the same arguments we may conclude that every two vertices that are adjacent in the branch graph $B(F_1/K_4)$ do not intersect with the same subtree in S.

The paths P_b, P_c, P_d intersect with different subtree of S and all intersect with P_3. This is a contradiction, since P_3 can not intersect with three different subtrees. Therefore, F_1 is not a VPT graph.

The graph F_k.

Let F_k be the graph obtained from F_1 by replacing each of a, b, c, d with k non-adjacent twins $a_1, \ldots, a_k, b_1, \ldots, b_k, c_1, \ldots, c_k, d_1, \ldots, d_k$, as shown in Figure 25.

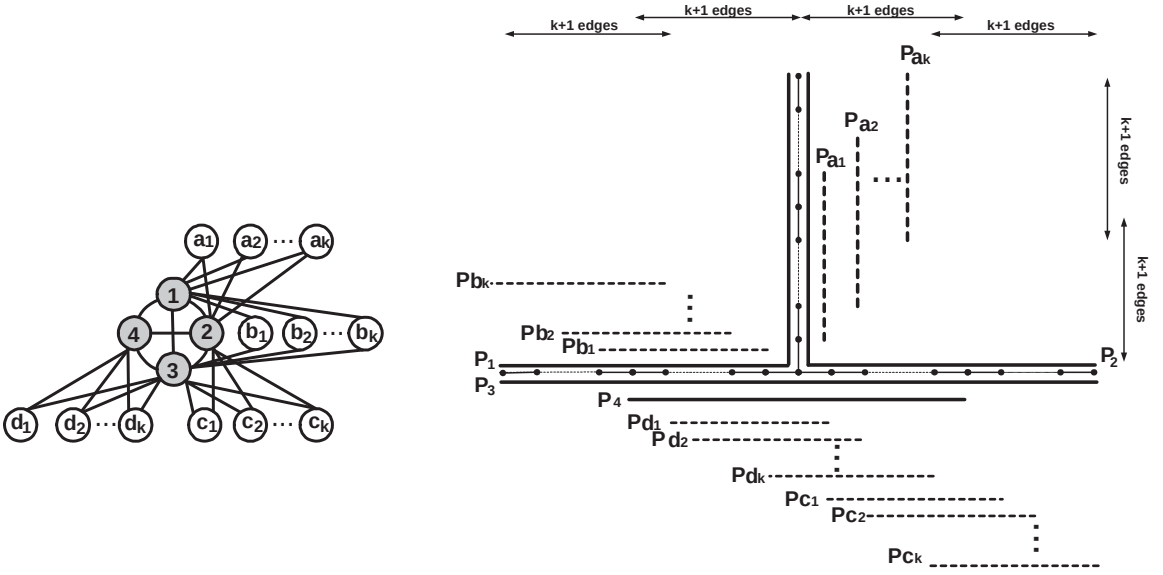

Figure 25: The graph F_k and a $(k+1)$-EPT representation of F_k.

The graph F_k has a $(k+1)$-EPT representation shown in Figure 25 for a fixed k.

Suppose F_k has a k-EPT representation for some fixed k. According to Lemma 10.9, there exists a vertex in $\{a_1, b_1, c_1, d_1\}$ that must contain the central vertex of the k-claw. Without loss of generality let P_{d_1} be the path that contains the central vertex of the k-claw. Then each one of the paths $P_{d_1}, \ldots, P_{d_k}$ must all contain the central vertex and intersect on at least k edges with two legs, say L_1 and L_2, of the k-claw, but not more than k edges with each of the legs. None of $P_{d_1}, \ldots, P_{d_k}$ may have a common endpoint. Therefore, every leg must have k different endpoints, contradicting the fact that a leg of the k-claw has k edges. Therefore, F_k is not a k-EPT graph for fixed k.

The graph F_k is not VPT, since F_1 is an induced subgraph in F_k, and we have already shown that F_1 is not a VPT graph.

The graph F^*.

Let $K_5 = \{1, 2, 3, 4, 5\}$ be the colored maximal clique in the chordal graph F^* shown in Figure 19. Then the remaining vertices $\{a, b, c, d\}$ are adjacent in

$B(F^*/K_5)$, meaning that $B(F^*/K_5)$ can not be 3-colored or 2-colored. Therefore, F^* is not a k-EPT graph for any k according to Theorem 8.15, namely $F^* \notin$ EPT*.

Suppose there exists a VPT representation $\langle \mathcal{P}, T \rangle$ of F^*.

The paths P_1, P_2, P_3, P_4, P_5 share a common vertex u on T due to Theorem 8.2. Suppose $P_a \cap P_3 = \emptyset$, $P_a \cap P_4 = \emptyset$, $P_a \cap P_1 \neq \emptyset$, $P_a \cap P_2 \neq \emptyset$ and $P_a \cap P_5 \neq \emptyset$, there exists an edge (u, v_1) which is contained in P_1, P_2, P_5 and is not contained in P_3 and P_4. Similarly, there exists an edge (u, v_2) which is contained in P_1, P_2, P_3 and is not contained in P_4 and P_5. There also exists an edge (u, v_3) which is contained in P_1, P_3, P_4 and is not contained in P_2 and P_5. This is a contradiction, since P_1 does not contain (u, v_1), (u, v_2) and (u, v_3). Therefore, F^* is not a VPT graph.

4. **Incomparabilities.**

The separation examples between incomparable classes of graphs are given in Table 3.

We have shown all the containment relationship in the hierarchy, and therefore the hierarchy is complete. □

Theorem 10.7 derives the following interesting relationships:

Corollary 10.10 *The family of chordal 1-EPT graphs is strictly contained in the family of chordal k-EPT graphs, for $k > 1$.*

Theorem 10.11 *The family of 1-EPT graphs is strictly contained in the family of k-EPT graphs for any fixed $k \geq 2$. The containment is also strict, when restricted to a degree 3 tree.*

Proof: The family of 1-EPT graphs is contained in the family of k-EPT graphs by Lemma 7.1. The graph D is not 1-EPT as proved in [23], but it has a k-EPT representation on a degree 3 tree, for $k \geq 2$, as illustrated. □

Table 3: Separating examples between k-EPT graphs and related incomparable classes.

A	B	$G_1 \in A - B$	$G_2 \in B - A$
1-EPT/EPT*/k-EPT/ orthodox 1-EPT	weakly chordal/ chordal/VPT	C_5	A_d
k-EPT/1-EPT/ orthodox 1-EPT	weakly chordal EPT*/ chordal EPT*	C_5	F_k
1-EPT/ orthodox 1-EPT	weakly chordal k-EPT/ chordal k-EPT	C_5	F_1
orthodox 1-EPT	weakly chordal 1-EPT/ chordal 1-EPT	C_5	$K_{1,3}$
weakly chordal 1-EPT	chordal EPT*/ chordal k-EPT	C_4	F_1
weakly chordal 1-EPT	VPT/chordal	C_4	A_d
weakly chordal 1-EPT/ weakly chordal k-EPT	chordal EPT*	C_4	F_k
chordal EPT*/ chordal k-EPT	VPT	F_1	A_d

Theorem 10.12 *The family of VPT graphs is incomparable with the family of k-EPT graphs for any fixed $k \geq 1$. When restricted to degree 3 trees, the family of VPT graphs is strictly contained in the family of k-EPT graphs, $k \geq 2$.*

Proof: The families of VPT and k-EPT are incomparable (C_4 and A_4 are separation examples). When restricted to degree 3 trees, the family of VPT graphs coincides with the family of 1-EPT graphs, and therefore is contained in the family of k-EPT graphs ($k \geq 2$) according to Lemma 7.1. The containment is strict, since the graph in Fig. 18 has a k-EPT representation on a degree 3 tree for every fixed $k \geq 2$ and is not a VPT graph. □

11 Properties of the maximum degree of the host tree

This section investigates the relationship between structural properties of a graph G and the maximum degree of the host tree in a $(h, 2, t)$-representation of G.

Consider the class $[h, s, t]$. In the case of $t = 1$, we know that restricting $s = 3$ allows restricting $h = 3$ without shrinking the class of graphs, since $[\infty, \infty, 1] = [3, 3, 1]$. However, restricting $s = 2$ does not allow restricting h without also shrinking the class, since $[h-1, 2, 1] \subset\neq [h, 2, 1]$ for all $h \geq 3$.

We have seen in Section 8.2 that the graph $C_d, d \geq 4$, has a unique 1-EPT representation (pie), which forces the degree of the host tree to be at least d. In this section, we will prove analogous result that the graph A_d has a unique VPT representation, which forces the degree of the host tree to be at least d. We also prove that in every k-EPT representation of the graph W_r the degree of the host tree is restricted.

The new results in this section are illustrated in hierarchy in Figure 26. This hierarchy has the following separating examples, as we will prove:
The graph $C_d, d \geq 4$, is a separating example between $[d, 2, 2]$ and $[d-1, 2, 2]$. The graph $A_d, d \geq 4$, is a separating example between $[d, 2, 1]$ and $[d-1, 2, 1]$. The graph $W_r, r \geq 4$, is a separating example between $[d, 2, k]$ and $[d+1, 2, k+1]$ for $r \leq d(d-1)^{\frac{k-1}{2}}$ for odd k and for $r \leq 2(d-1)^{\frac{k}{2}}$ for even k.

A structure more rigid than a pie will be needed to represent our other separating examples.

Definition 11.1 *Let $\langle \mathcal{P}, T \rangle$ be a $(h, 2, t)$-representation of a graph G. A **complete hub** on d edges is a star subgraph of T with d edges, such that each pair of edges is contained in a different member of $\mathcal{P}$.*

Remark 11.2 *Given a VPT representation $\langle \mathcal{P}, T \rangle$ of a graph G, a pie and a complete hub in T each corresponds to a clique. Given a 1-EPT representation $\langle \mathcal{P}, T \rangle$ of a graph*

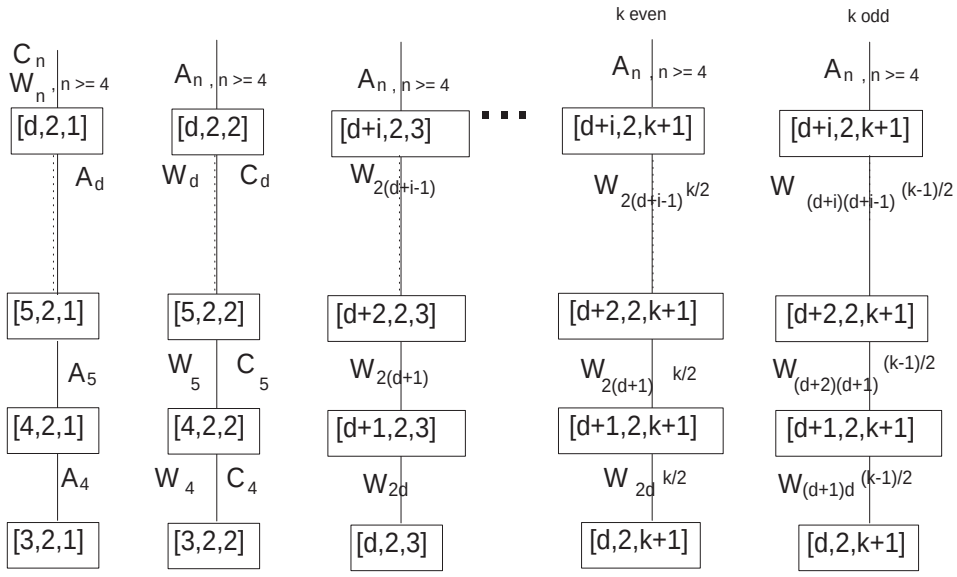

Figure 26: Properties of the maximum degree of the host tree in $[h, 2, t]$ graphs.

G, a pie corresponds to a chordless cycle and a complete hub corresponds to the graph W_r which will be defined below.

11.1 The graph A_d

The VPT representation of the graph $A_d, d \geq 4$, is described in the proof of Theorem 10.7. The following theorem shows that there is essentially only one VPT representation for $A_d, d \geq 4$.

Theorem 11.3 *Let $\langle \mathcal{P}, T \rangle$ be a $(h, 2, 1)$-representation (VPT) of a graph G. If G contains an induced subgraph $A_d, d \geq 4$, then T contains a complete hub on d edges with central vertex b and distinct edges $(b, a_1), \ldots, (b, a_d)$ such that:*

1. *$Q_l, 1 \leq l \leq d$ is in the subtree containing a_l after removing b,*

2. *$(b, a_i), (b, a_j) \in P_{ij}, 1 \leq i < j \leq d$.*

Proof: Let $\langle \mathcal{P}, T \rangle$ be a VPT representation of G having an induced subgraph A_d and let P_{ij} and Q_i be the paths in T corresponding to vertices p_{ij} and q_i in G. By Theorem 8.2, the paths $\{P_{ij}\}$ share a common vertex b in T. Let $(b, a_1), \ldots, (b, a_n)$ be the edges with endpoint b in T. Each path Q_i can not contain the vertex b, so it

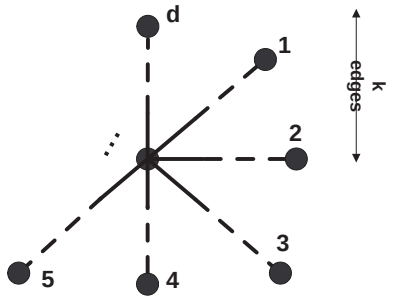

Figure 27: A k-EPT representation tree T of the graph $W_r, r > 1$, where P_{ij} goes from i to j.

must be in a subtree containing one of $a_1, \ldots, a_n$, obtained by removing b. Suppose Q_i and $Q_j, i \neq j$ are both in the same subtree containing a_l. Then the path P_{ij} contains a_l, at least one vertex v of Q_i and at least one vertex w of Q_j, while $v \neq w$. Without loss of generality, suppose v is on the shortest path from a_l to w. Therefore, for some $m \neq i, j$, the path P_{jm} contains both v and w. This is a contradiction, since $(p_{jm}, q_i) \notin E(A_d)$. Therefore, every subtree containing one of $a_1, \ldots, a_n$, obtained by removing b, contains at most one path Q_i. We now rename each of Q_i into Q_l if it is in a subtree containing a_l. Thus, $n \geq d$ and T contains a star with central vertex b and the collection of paths $\{(a_i, b)(b, a_j) \in P_{ij}\}$, which form a complete hub on d edges. $\square$

Corollary 11.4 *Let G be an $[h, 2, 1]$ graph. If G contains $A_d, d \geq 4$, then $h \geq d$.*

11.2 The graph W_r

We have shown in section 8.2 that a chordless cycle does not have a unique $(h, 2, t)$-representation for $t > 2$. Therefore, the size of the cycle is not an essential condition for the maximum degree of the host tree for $[h, 2, t]$ graphs. However, we have found a graph W_r that gives an essential condition for the maximum degree of the host tree in any $(h, 2, t)$-representation.

For an integer $r \geq 4$, let W_r be the graph, such that $V(W_r) = \{x_{ij} | 1 \leq i < j \leq r\}$ and $(x_{ij}, x_{i'j'}) \in E(W_r)$ if and only if $|\{i, j\} \cap \{i', j'\}| = 1$. The graphs W_4 and W_5

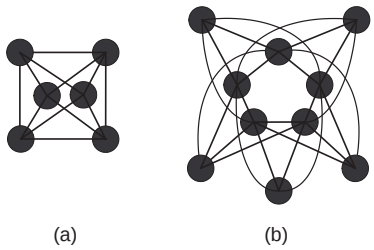

Figure 28: (a) the graph W_4; (b) the graph W_5.

are shown in Figure 28.

The graph W_r has a k-EPT representation, for $k \geq 1$, which contains a complete hub on r edges on a T_k^r with each leg of k edges, such that the path P_{ij}, corresponding to x_{ij}, contains legs i and j,as shown in Figure 27.

Theorem 11.5 *Let $\langle \mathcal{P}, T \rangle$ be a k-EPT representation of the graph W_r and d the maximum degree of the tree T. If k is odd, then $r \leq d(d-1)^{\frac{k-1}{2}}$, else $r \leq 2(d-1)^{\frac{k}{2}}$.*

Proof: Let $K_l = \{x_{ij} | i = l \text{ or } j = l\}$. The sets $K_l, 1 \leq l \leq r$, are maximal cliques of size $r-1$ in the graph W_r. Let $\mathcal{L}_l$ be the subpath in T which is the intersection of all the paths corresponding to K_l. Since W_r is a k-EPT graph, K_l is either a k-claw clique or a k-edge clique due to Proposition 8.10. Therefore, $\mathcal{L}_l$ is either a single vertex or has at least k edges.

We first consider the case of $r = 4$.

Claim 11.6 *Let $\langle \mathcal{P}, T \rangle$ be a k-EPT representation of the graph W_4. If there exists a maximal clique $K_l, 1 \leq l \leq r$, which is a k-claw clique in $\langle \mathcal{P}, T \rangle$, then $\langle \mathcal{P}, T \rangle$ contains a complete hub on 4 edges.*

Proof: Suppose the clique K_l corresponds to a k-claw clique in $\langle \mathcal{P}, T \rangle$ with central vertex b and legs L_1, L_2, L_3, and therefore $\mathcal{L}_l = \{b\}$. Without loss of generality, suppose the path P_{12} contains the legs L_1, L_2, the path P_{13} contains the legs L_1, L_3 and the path P_{14} contains the legs L_2, L_3. See Figure 29(a).

Since $|P_{23} \cap P_{12}| \geq k$, $|P_{23} \cap P_{13}| \geq k$ and $|P_{23} \cap P_{14}| < k$, the path P_{23} contains neither L_2 nor L_3. Since $|P_{34} \cap P_{13}| \geq k$, $|P_{34} \cap P_{14}| \geq k$ and $|P_{34} \cap P_{12}| < k$, the path

P_{34} contains neither L_1 nor L_2. Since $|P_{34} \cap P_{23}| \geq k$, the paths P_{34} and P_{23} either intersect on $L_1 \cup L_3$ as shown in Figure 29(b) or intersect on another leg L_4 as shown in Figure 29(c). In case of $k = 1$, the paths P_{34} and P_{23} must intersect on another leg L_4.

Suppose the paths P_{34} and P_{23} intersect on $L_1 \cup L_3$. Since $|P_{24} \cap P_{12}| \geq k$, $|P_{24} \cap P_{14}| \geq k$ and $|P_{24} \cap P_{13}| < k$, the path P_{24} intersects with neither L_1 nor L_3. This is a contradiction, since $|P_{23} \cap P_{24}| \geq k$.

If the paths P_{34} and P_{23} intersect on L_4, then tree T'=$\bigcup_l \mathcal{L}_l$ and the collection of paths $\{P_{ij}\}$ form a complete hub on 4 edges. This proves Claim 11.6. □

Thus, Claim 11.6 shows that $r \leq d$ and proves the theorem for the special case where $r = 4$ and some K_l is a k-claw clique.

Claim 11.7 *For $r > 4$, every maximal clique $K_l, 1 \leq l \leq r$, is a k-edge clique and the subpath $\mathcal{L}_l$ has at least k edges in any k-EPT representation of G.*

Proof: Suppose the clique K_1 corresponds to a k-claw clique in $\langle \mathcal{P}, T \rangle$ with central vertex b and legs L_1, L_2, L_3, and therefore $\mathcal{L}_1 = \{b\}$. Without loss of generality, suppose the paths P_{12} and P_{15} contain the legs L_1, L_2, the path P_{13} contains the legs L_1, L_3 and the path P_{14} contains the legs L_2, L_3. See Figure 29(a).

Since $|P_{23} \cap P_{14}| < k$ and $|P_{23} \cap P_{15}| < k$, the path P_{23} contains neither L_1 nor L_2 nor L_3. Since $|P_{23} \cap P_{12}| \geq k$, $|P_{23} \cap P_{13}| \geq k$, the path P_{23} is contained in a subtree S_1, that contains L_1, obtained by removing the central vertex b.

Since $|P_{34} \cap P_{13}| \geq k$, $|P_{34} \cap P_{14}| \geq k$, the path P_{34} is not contained in a subtree S_1. Since $|P_{34} \cap P_{23}| \geq k$, the path P_{34} must contain L_1. This is a contradiction, since $|P_{34} \cap P_{15}| < k$.

Therefore, the maximal clique K_1 corresponds to a k-edge clique in $\langle \mathcal{P}, T \rangle$. Similarly, all the maximal cliques $K_l, 1 \leq l \leq r$ correspond to k-edge cliques in $\langle \mathcal{P}, T \rangle$. This proves Claim 11.7. □ Now, by Claims 11.6 and 11.7, we may assume that for

all $r \geq 4$ every maximal clique $K_l, 1 \leq l \leq r$, corresponds to a k-edge clique in any k-EPT representation of G and $\mathcal{L}_l$ has at least k edges.

Claim 11.8 *In any k-EPT representation $< \mathcal{P}, T >$ of W_r, the subpaths $\mathcal{L}_i$ and $\mathcal{L}_j$ share at most one vertex in T for all pairs $1 \leq i < j \leq r$.*

Proof: Suppose there exists a k-EPT representation of W_r, such that the subpaths $\mathcal{L}_i$ and $\mathcal{L}_j$ share more than one vertex, but obviously less than k edges. Since the path P_{ij} contains $\mathcal{L}_i$ and $\mathcal{L}_j$, it follows that $\mathcal{L}_i$ and $\mathcal{L}_j$ overlap at their ends. For any $m \neq i, j$, since $\mathcal{L}_i \subseteq P_{im}$, $\mathcal{L}_j \subseteq P_{jm}$ and $|P_{im} \cap P_{jm}| \geq k$ edges, $P_{im} \cap P_{jm} \subset P_{ij}$. This is a contradiction, since $\mathcal{L}_m \subseteq P_{im} \cap P_{jm}$, but $x_{ij} \notin K_m$. $\square$

Let R_{ij} be the shortest path in T from $\mathcal{L}_i$ to $\mathcal{L}_j$. By Claim 11.8, R_{ij} contains at least one vertex. Let **core** subtree T' be the subtree obtained from T, such that $T' = \bigcup_{1 \leq i < j \leq r} R_{ij}$.

We claim that T' is a connected subtree. Suppose T' is not connected and the paths R_{ij} and R_{im} are in different connected components of T'. Then R_{ij} and R_{im} share two different endpoints of the subpath $\mathcal{L}_i$. Therefore, the path P_{mj} contains $\mathcal{L}_m$, R_{im}, $\mathcal{L}_i$, R_{ij} and $\mathcal{L}_j$. This is a contradiction, since x_{jm} is not in the clique K_i. Therefore, the paths R_{ij} and R_{im} are in the same connected component of T'. Also, this allows us to show that R_{ij} and R_{mn} are in the same connected component of T', for any i, j, m, n, since R_{ij} and R_{im} are in the same component and R_{im} R_{mn} are in the same component. Hence, T' is a connected subtree.

By definition, each leaf of T' is an endpoint of some subpath $\mathcal{L}_l, 1 \leq l \leq r$. Since T' is connected tree, only one endpoint of each $\mathcal{L}_l$ is a leaf of T'. Suppose there exists a leaf v in T' and a pendant edge (u, v), such that v is an endpoint of only one subpath $\mathcal{L}_i$. Then the path P_{ij} contains the edge (u, v) for all j, meaning that the edge (u, v) is contained in $\mathcal{L}_i$ and is not contained in T'. This is a contradiction, and hence every leaf of T' is an endpoint of at least two among $\mathcal{L}_l, 1 \leq l \leq r$.

We claim that the length of the longest path from a leaf v to a leaf w in T' is

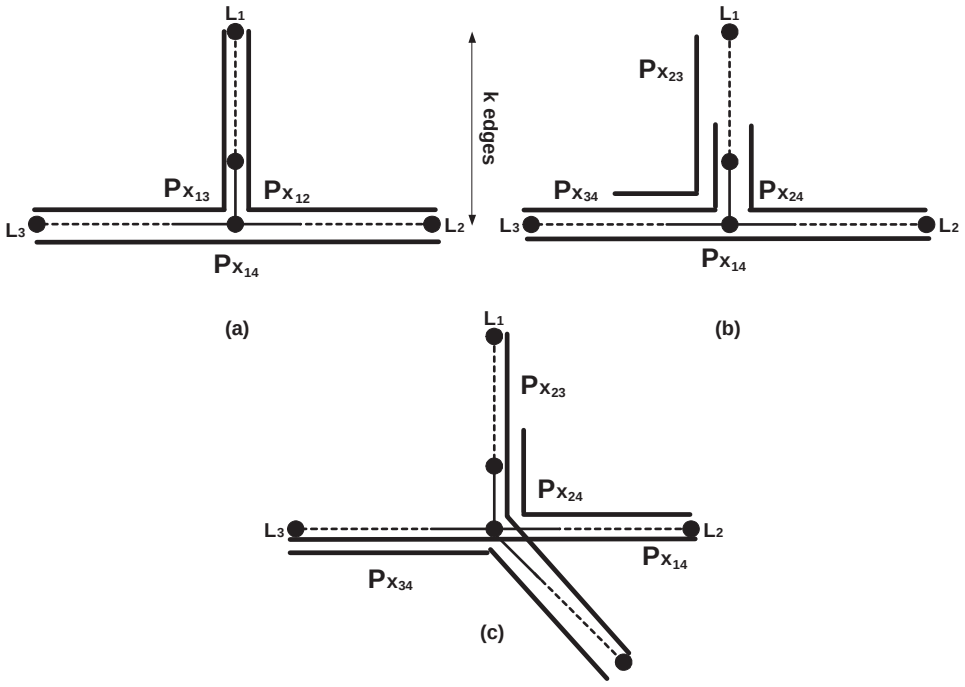

Figure 29: A k-EPT representation of the clique K_1 in the graph W_4.

at most $k-1$ edges. Suppose v is an endpoint of subpaths $\mathcal{L}_i$ and $\mathcal{L}_j$ and w is an endpoint of subpaths $\mathcal{L}_m$ and $\mathcal{L}_n$ in T. If the paths P_{im} and P_{jn} share more than $k-1$ edges in T, then this is a contradiction.

It is well known that a degree d tree, whose longest path has m edges, can have at most $2(d-1)^{\frac{m-1}{2}}$ leaves if m is odd and at most $d(d-1)^{\frac{m}{2}-1}$ leaves if m is even. In our case $m = k-1$, so $r \leq d(d-1)^{\frac{k-1}{2}}$ if k is odd and $r \leq 2(d-1)^{\frac{k}{2}}$ if k is even.

□

Corollary 11.9 *For a fixed odd k and $r = (d+1)d^{\frac{k-1}{2}}$, or for a fixed even k and $r = 2d^{\frac{k}{2}}$, the k-EPT graph W_r does not have a k-EPT representation on a degree d tree, but W_r does have a k-EPT representation on a degree $d+1$ tree.*

Proof: For a fixed odd k and r=$(d$+1$)d^{\frac{k-1}{2}}$, the Corollary follows, since $d(d-1)^{\frac{k-1}{2}} < d(d)^{\frac{k-1}{2}} < (d+1)d^{\frac{k-1}{2}}$. Hence, $d(d-1)^{\frac{k-1}{2}} < r = (d+1)d^{\frac{k-1}{2}}$.

For a fixed even k and $r = 2d^{\frac{k}{2}}$, the Corollary follows, since $2(d-1)^{\frac{k}{2}} < 2d^{\frac{k}{2}}$. Hence, $2(d-1)^{\frac{k}{2}} < r = 2d^{\frac{k}{2}}$. □ For example, the graph W_r has a $(3, 2, k+1)$-

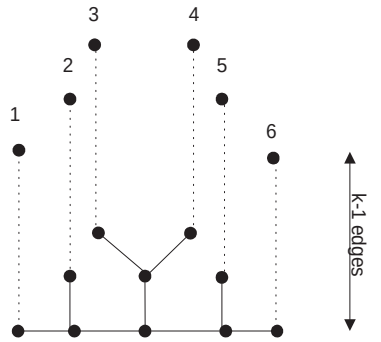

Figure 30: A $(3, 2, k+1)$ representation of the graph W_r, for $r = 6$.

representation for $2log(r) \leq k$, i.e., a k-EPT representation on a binary tree, shown in Figure 30. The path P_{ij} contains i and j. Two paths that correspond to non-adjacent vertices, share only $2log(r)$ edges of the core.

12 Open questions and further research

An interesting open problem is the relationship between the family of k-EPT graphs and $(k+1)$-EPT graphs. It is possible that there exists a value t, such that for every $k > t$, the families of k-EPT and $(k+1)$-EPT graphs coincide. In that case, we want to find a transformation from k-EPT representation into $(k+1)$-EPT representation. It is also possible that there exists a graph which is not a k-EPT graph, but is a $(k+1)$-EPT graph for any fixed k. In that case, the families of k-EPT graphs differ from each other for different values of k. A related problem is to characterize the class of EPT* graphs. Fig. 20 shows one example of a graph which is not EPT*.

Jamison and Mulder in [39] suggest the following conjecture and prove it for $k = 2$ and $k = 3$ and for all $k' \geq k^2 - 4k + 6$.

Conjecture 12.1 [39] *The family of k-EPT graphs $\subseteq$ the family of k'-EPT graphs, where $2 \leq k \leq k'$.*

We would like to know what is the relationship between the maximum degree d of the host tree and the length l of the longest chordless cycle ($l \geq 4$) for various cases. Corollary 8.7 shows that $l \leq d$ for 1-EPT graphs. For weakly chordal 1-EPT

graphs we have shown in Theorem 9.6 that $d \leq 4$. A 1-EPT graph does not contain $\overline{C_d}, d \geq 7$ according to [24]. A 1-EPT graph $\overline{C}_6$ has $l = 4$ and $d > 4$. A 1-EPT graph $\overline{C}_5$ has $l = d = 5$. For which values of l does 1-EPT graph G have a degree $d = l$ 1-EPT representation? Possibly, if $l \geq 5$, then $l = d$?

We also would like to find the relationship between orthodox k-EPT graphs restricted to degree 3 trees and other families of graphs in the hierarchy. Specifically, whether orthodox k-EPT graphs and orthodox k-EPT on degree 3 trees are equivalent or there exists a separating example between these families of graphs.

It is well-known that a graph G is chordal if and only if the graph G is an intersection graph of subtrees of a tree. We ask, does there exist an analogous characterization for weakly chordal graph, i.e., a characterization in terms of intersection graphs.

References

[1] H. Ariyoshi, M. Ide, I. Shirakawa, S. Tsukiyama. A new algorithm for generating all the maximal independent sets. *SIAM J. Computing* 6 (1977), 505-517.

[2] B. Beauquier, J.C. Bermond, L. Gargano, S. Hell, P. Perennes, U. Vaccaro. Graph problems arising from wavelength-routing in all-optical networks. *Proc. of 2nd Workshop on Optics and Computer Science, part of IPPS* 1997.

[3] S. Benzer. On the topology of the genetic fine structure. *Proc. Nat. Acad. Sci.* 45 (1959), 1607-1620.

[4] C. Berge. Graphs and Hypergraphs. *North-Holland, Amsterdam,* 1973.

[5] A. Berry, J.P. Bordat, P. Heggernes. Recognizing weakly triangulated graphs by edge separability. *Nordic Journal of Computing* 7(3), Fall(2000), 164-177.

[6] A. Berry, J.P. Bordat, P. Heggernes, G. Simonet, Y. Villanger. A wide-range algorithm for minimal triangulation from an arbitrary ordering. *Research Report LIMOS/RR-03-02, Clermont-Ferrand, France, submitted for publication.*

[7] A. Berry, M.C. Golumbic, and M. Lipshteyn. Recognizing and triangulating chordal probe graphs. *Research Report LIMOS/RR-03-14, Clermont-Ferrand, France*, July 2003.

[8] A. Berry, M.C. Golumbic, and M. Lipshteyn. Two tricks to triangulate chordal probe graphs in polynomial time. *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2004), 962-969.

[9] A. Berry, M.C. Golumbic, and M. Lipshteyn. Cycle-bicolorable graphs and triangulating chordal probe graphs. *Submitted.*

[10] A. Berry, P. Heggernes, G. Simonet. The minimum degree heuristic and the minimal triangulation process. *Graph-Theoretic Concepts in Computer Science 2880* (2003), 58-70.

[11] H.L. Bodlaender, M.R. Fellows, T.J. Warnow. Two strikes against perfect phylogeny. *Proceedings of 19th ICALP, Lecture Notes in Computer Science* 623 (1992), 273-283.

[12] K. Booth, G.S. Lueker. Testing for the consecutive ones property, interval graphs and graph planarity using pq-tree algorithm. *J. Comput. Syst. Sci.* 13 (1976), 335-379.

[13] A. Brändstadt, V. Le, J.P. Spinrad. Graph classes: A Survey. *SIAM Monographs on Discrete Mathematics and Applications, Philadelphia* (1999).

[14] P. Buneman. A characterization of rigid circuit graphs. *Discrete Math.* 9 (1974), 205-212.

[15] M. Chudnovsky, N. Robertson, P. Seymour and R. Thomas. The Strong Perfect Graph Theorem. *Submitted for publication.*

[16] G.A. Dirac. On rigid circuit graphs. *Abh. Math. Sem. Univ. Hamburg*, 25 (1961), 71-76.

[17] T. Erlebach, K. Jansen. Scheduling of virtual connections in fast networks. *Proc. of the 4th Parallel Systems and Algorithms Workshop (PASA)* (1996), 13-32.

[18] T. Erlebach, A. Pagourtzis, K. Potika, S. Stefanakos. Resource allocation in multifiber WDM tree networks. *Graph-Theoretic Concepts in Computer Science* 2880 (2003), 218-229.

[19] D.R. Fulkerson, O.A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Math.* 15 (1965), 835-855.

[20] F. Gavril. A recognition algorithm for the intersection graphs of paths in trees. *Discrete Math.* 23 (1978), 211-227.

[21] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. Comb. Theory Ser. B* (1974), 47-56.

[22] M.C. Golumbic. Algorithmic Graph Theory and Perfect Graphs. *Second Edition, Annals of Discrete Math., Elsevier* 57 (2004).

[23] M.C. Golumbic, R.E. Jamison. Edge and vertex intersection of paths in a tree. *Discrete Mathematics* 55 (1985), 151-159.

[24] M.C. Golumbic, R.E. Jamison. The edge intersection graphs of paths in a tree. *Journal of Combinatorial Theory, Series B* 38 (1985), 8-22.

[25] M.C. Golumbic, H. Kaplan, R. Shamir. Graph sandwich problems. *Journal of Algorithms* 19 (1995), 449-473.

[26] M.C. Golumbic, M. Lipshteyn. On the hierarchy of tolerance, probe and interval graphs. *Congressus Numerantium* 153 (2001), 97-106.

[27] M.C. Golumbic, M. Lipshteyn. Chordal probe graphs (extended abstract). *Graph Theoretic Concepts in Computer Science* 2880 (2003), 249-260.

[28] M.C. Golumbic, M. Lipshteyn. Chordal probe graphs. *Discrete Applied Mathematics* 143 (2004), 221-237.

[29] M.C. Golumbic, M. Lipshteyn, M. Stern. The recognition of k-EPT graphs. *Congressus Numernatium* 171 (2004), 129-139.

[30] M.C. Golumbic, M. Lipshteyn, M. Stern. The k-edge intersection graphs of paths in a tree. Accepted to *Discrete Applied Mathematics.*

[31] M.C. Golumbic, M. Lipshteyn, M. Stern. Equivalences and the complete hierarchy of intersection graphs of paths in a tree. *Submitted.*

[32] M.C. Golumbic, M. Lipshteyn, M. Stern. Representing Edge Intersection Graphs of Paths on a Degree 4 Trees. *Proc. of EuroComb'05, DMTCS Conference Volume AE* (2005), pp. 193-198.

[33] M.C. Golumbic, C.L. Monma, W.T. Trotter. Tolerance graphs. *Discrete Applied Mathematics* 9 (1984), 157-170.

[34] M.C. Golumbic, A.N. Trenk. Tolerance Graphs. *Cambridge University Press* (2004).

[35] R. Hayward. Weakly triangulated graphs. *J. Combin. Theory B*, 39 (1985), 200-208.

[36] R.B. Hayward, C.T. Hoàng, F. Maffray. Optimizing weakly triangulated graphs. *Graphs and Combinatorics* 5 (1989), 339-349.

[37] R.B. Hayward, J. Spinrad, R. Sritharan. Weakly chordal graph algorithms via handles. *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2000), 42-49.

[38] I. Holyer, The NP-completeness of edge-coloring. *SIAM Journal on Computing* 10 (1981), 718-720.

[39] R.E. Jamison, H.M. Mulder. Constant tolerance representations of graphs in trees. *Congressus Numerantium* 143 (2000), 175-192.

[40] R.E. Jamison, H.M. Mulder. Tolerance intersection graphs on binary trees with constant tolerance 3. *Discrete Math.* 215 (2000), 115-131.

[41] R.E. Jamison, H.M. Mulder. Constant tolerance intersection graphs of subtrees of a tree. *Discrete Math.* 290 (2005), 27-46.

[42] J.L. Johnson, J. Spinrad. A polynomial time recognition algorithm for probe interval graphs. *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2001), 477-486.

[43] T. Kloks, D. Kratsch, J. Spinrad. Treewidth and pathwidth of cocomparability graphs of bounded dimension. *Res. Rep. 93-46, Eindhoven University of Technology* (1993).

[44] C.G. Lekkerkerker, J.Ch. Boland. Representation of a finite graph by a set of intervals on the real line. *Fund. Math.* 51 (1962), 45-64.

[45] R. McConnell, J. Spinrad. Construction of probe interval models. *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2002), 866-875.

[46] F.R. McMorris, E. Scheinerman. Connectivity threshold for random chordal graphs. *Graphs and Combin.* 7 (1991), 177-181.

[47] F.R. McMorris, C. Wang and P. Zhang. On probe interval graphs. *Discrete Applied Mathematics* 88 (1998), 315-324.

[48] T.A. McKee and F.R. McMorris. Topics in Intersection Graph Theory. *SIAM Monographs on Discrete Mathematics and Applications, Philadelphia* (1999).

[49] A. Parra and P. Scheffler. How to use the minimal separators of a graph for its chordal triangulation. *Proceedings of ICALP '95), Lecture Notes in Computer Science* 944 (1995), 123-134.

[50] L. Sheng. Cycle free probe interval graphs. *Congressus Numerantium* 140 (1999), 33-42.

[51] J. Spinrad. Efficient Graph Representations. *Providence, R.I.: American Mathematical Society* (2003).

[52] M. Steele. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification* 9 (1992), 91-116.

[53] M.M. Syslo. Triangulated edge intersection graphs of paths in a tree. *Discrete Math.* 55 (1985), 217-220.

[54] R.E. Tarjan. Decomposition by clique separators. *Discrete Math.* 55 (1985), 221-232.

[55] J.R. Walter. Representations of rigid cycle graphs. *PhD Thesis, Wayne State Univ.* (1972).

[56] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. Algebr. Discrete Methods* 2 (1981), 77-79.

[57] P. Zhang. Probe interval graphs and their application to physical mapping of DNA. *Manuscript* (1994).

[58] P. Zhang, E.A. Schon, S.G. Fischer, E. Cayanis, J. Weiss, S. Kistler, P.E. Bourne. An algorithm based on graph theory for the assembly of contigs in physical mapping of DNA. *CABIOS* 10 (1994), 309-317.

[59] P. Zhang, X. Ye, L. Liao, J. Russo, S.G. Fischer. Integrated mapping package - a physical mapping software tool kit. *Genomics* 55 (1999), 78-87.