University of Alberta

Library Release Form

Name of Author: Eowyn W Čenek

Title of Thesis: Subtree Overlap Graphs and the Maximum Independent Set Problem

Degree: Master of Science

Year this Degree Granted: 1998

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

. . . . . .

Eowyn W Čenek
15011 86 Ave
Edmonton, Alberta
Canada, T5R 4B9

Date: . . . . . .

University of Alberta

Subtree Overlap Graphs and the Maximum Independent Set Problem

by

Eowyn W Čenek

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Spring 1998

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Subtree Overlap Graphs and the Maximum Independent Set Problem** submitted by Eowyn W Čenek in partial fulfillment of the requirements for the degree of **Master of Science**.

. . . . . .
Dr L. Stewart

. . . . . .
Dr J. Culberson

. . . . . .
Dr A. Liu

Date: . . . . . .

Abstract

A graph G is a subtree overlap graph if there exists a tree T and a set of subtrees $\{T_i\}$ so that there exists a one-to-one mapping between vertices and subtrees and two subtrees overlap if and only if their respective vertices are adjacent. The class of subtree overlap graphs is proven to contain the classes of circle, spider or circle polygon, and chordal graphs.

An upper bound on the size of the subtree overlap model is proven to be $3m$. As well a general algorithm to find the maximum independent set for any class of overlap graph is given, provided testing for containment and intersection in the overlap graph can be done in polynomial time, and the maximum weight independent set problem is solved for the related class of intersection graph. The complexities of the Hamiltonian Cycle, several domination problems, isomorphism and colouring are shown to be as hard for subtree overlap graphs as they are for graphs in general.

Acknowledgements

Many people have helped me with this thesis, not least by listening to me blather on as I tried to convince myself certain points were true and had value. Firstly, I'd like to thank my supervisor, Professor Lorna Stewart, who suggested the study of the subtree overlap graph. Secondly, I'd like to thank Phil Nadeau. Trying to explain the underlying themes of this thesis in e-mail led to the developing of the structure of this thesis as it stood in previous versions. Most importantly, having a structure made writing a great deal easier. Similarly, Mike Jones, who proofread a very early version and pointed out all the inconsistencies between definitions. They have been, I hope, fixed. Lastly, my mother, without whom this thesis would have been a lot harder to write. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Intersection, Containment, and Overlap Graphs

Different ways exist to represent graphs. The traditional model defines the graph in terms of a set of vertices and a set of edges, where each edge consists of a pair of vertices. We will assume each graph is a simple, undirected, connected graph and therefore does not contain loops or multiple edges between two vertices. Or, alternatively, the two vertices comprising each edge will be different, and each edge pair will be unique in the set of edges, without regard to direction.

An alternate way to model the graph's data relies on objects of some type Z to represent the individual vertices, and the relationship ϕ between these objects to represent the edge structure of the graph. These objects can be geometric - as for example unit circles in the plane, or they can consist of sets - such as for instance sets of integers. The possibilities for object types is broad, and any type of object can be used as long as it is possible to define intersection, containment, and overlap between any two objects of the this type. Then ϕ is defined, for two objects $z_i, z_j \in Z$ so that

$$\phi(z_i, z_j) = \begin{cases} 1 & \text{if } z_i \text{ and } z_j \text{ are related} \\ 0 & \text{if } z_i \text{ and } z_j \text{ otherwise} \end{cases}$$

Three such possible relationships are intersection, containment, and overlap. A fourth relation - disjointedness - will not be discussed. In an intersection graph, two vertices are adjacent if and only if their respective two objects intersect, whereas in a containment graph two vertices are adjacent if and only if the object representing one vertex is fully contained in the other. Lastly, in an overlap graph, two vertices are adjacent if their objects intersect, but neither object is contained in the other. These

three relationships are reflected in the following three refinements of the ϕ function, representing the intersection, containment, and overlap relationships respectively.

$$\phi_I(z_i, z_j) = \begin{cases} 1 & \text{if } z_i \text{ and } z_j \text{ intersect} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_C(z_i, z_j) = \begin{cases} 1 & z_i \text{ is contained in } z_j \\ & \text{or } z_j \text{ is contained in } z_i \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_O(z_i, z_j) = \begin{cases} 1 & \phi_I(z_i, z_j) = 1 \\ & \text{and } \phi_C(z_i, z_j) = 0 \\ 0 & \text{otherwise} \end{cases}$$

The definitions of these three methods to model a graph are almost identical, in that all three models are defined by the type of objects that represent the vertices in the original graph, but the edge relationships are defined by ϕ_I, ϕ_C, or ϕ_O.

Each of these models restricts the structure of the underlying graphs, but provides polynomial solutions for some problems which are NP-complete in general. Moreover, for some problems, this method of modelling the data can be more intuitive, as the definitions of the objects and their intersections can be tailored to individual classes of problems. For instance, if the objects consist of non-overlapping regions in the plane, then the intersection graph of these objects can be used to represent a planar graph, where two regions will intersect if they share one or more boundary points.

1.2 Motivation

The three classes of graphs modelled by intervals on a line and intersection, containment, or overlap relationships have been widely studied, both as graphs using this model of intervals on a line and as equivalent graphs that are represented differently by, for instance, intersecting chords inscribed in a circle.

Similarly, the subtree intersection and subtree containment graph classes have been studied extensively, albeit primarily as their equivalent classes, which are the classes of chordal and comparability graphs, respectively. Very little work has been done, however, on the class of subtree overlap graphs. As such, the subtree overlap graph class makes for an interesting study; I wanted to explore both the question of

recognizing and possibly constructing a subtree overlap graph model, and the computational complexity of some common graph theoretic problems, such as colouring, for the class of subtree overlap graphs.

1.3 Main Results

The class of subtree overlap graphs is not completely unexamined; a few facts are known about this class of graphs. It is known that every interval overlap graph is a subtree overlap graph, yielding a place in the hierarchy of graph classes. Moreover a sufficient condition has been proven to show that a given graph is not a subtree overlap graph. Research for this thesis, then, was directed to three questions: can the position of the subtree overlap graph class in the hierarchy be refined, can the subtree overlap graph class be characterized or can a recognition algorithm be constructed, and what are the computational complexities of well known graph problems for the class of subtree overlap graphs?

In the process of exploring the subtree overlap graph, three main results are proven:

1. The spider graph class is a proper subclass of the subtree overlap graph class.

2. The subtree overlap model modelling a subtree overlap graph is shown to contain at most $3m$ vertices, if the subtree overlap model is minimal. Here m = the number of edges in the subtree overlap graph

3. An algorithm is given to solve the Maximum Independent Set problem for any overlap model, including the subtree overlap model.

By proving 1, it is possible to further refine the position of the subtree overlap graph class in the hierarchy of classes. Especially so since the class of spider graphs itself includes a host of other graph classes.

While exploring different computational problems for the subtree overlap graph, a more general solution for the Maximum Independent Set problem was developed. This solution applies to any class of overlap graphs, and yields an algorithm to find the maximum independent set of any overlap graph in polynomial time if an intersection/containment matrix can be built in polynomial time for the overlap graph's

model and the maximum weight independent set can be found in polynomial time for the intersection graph represented by the same model.

Result 2 provides an upper bound on the size of a subtree overlap model. This upper bound applies only to minimal subtree overlap models, but an algorithm is given to reduce any subtree overlap model to a minimal subtree overlap model representing the same graph. Due to this upper bound, it is possible to build an intersection/containment matrix of a minimal subtree overlap model in $O(m)$ time. This in turn leads to the result that the Maximum Independent Set problem is solvable in $O(mn^2)$ time for subtree overlap graphs.

Therefore, of the three original questions we sought to address, two have been addressed. The question of recognizing or characterizing the subtree overlap graph class remains open.

1.4 Outline

Chapters 2 and 3 provide the definitions of the graph classes discussed in this thesis. In part, we are concerned with the placement of the subtree overlap graph class in the hierarchy of graph classes; this necessitates the definition of many graph classes. We begin by introducing graphs modelled by intervals on a line, and extend them to graphs modelled by subtrees of a tree.

Following these graph definitions, Chapter 4 discusses and proves the relationships between the graph classes. This is also where the first of the original results is presented, when it is proven that the spider graph class is a proper subclass of the subtree overlap graph class.

In chapter 5, the upper bound on the size of a minimal subtree overlap graph is proven, which leads to the result in Chapter 6 that the maximum independent set problem is solvable in $O(mn^2)$ time for subtree overlap graphs.

Chapter 7 discusses the computational complexity of several well known graph problems for the subtree overlap graph class. For most of these problems the complexity is NP complete, with the exception of the graph isomorphism problem, which is isomorphism complete.

Chapter 2

Intervals on a Line

Perhaps the easiest objects to visualize are intervals lying on a line. Each interval c_i on can be defined by its left and right endpoints so that $c_i = (l_i, r_i)$. Then a given graph $G = (V, E)$ can be modeled if every vertex $v_i \in V$ can be mapped to an interval c_i so that $\phi(c_i, c_j) = 1$ if and only if $(v_i, v_j) \in E$.

The interval model - be the relationship that of intersection, containment, or overlap - is a very useful mathematical structure and models real world problems well. The intervals all rest on a line; this line may represent time, or distance, or some other one-dimensional concept. The intervals, in turn, may be restricted by physical restrictions - such as file size in sequential storage in a computer, or time dependencies - such as the life span of a group of individuals. If an interval graph modeled, for instance, the lifespan of a group of humans on a timeline, finding the point in time at which the largest number of humans was living would involve finding the maximum clique in either the intersection or containment model.

Note that many of the definitions used in this chapter come from Golumbic's book, which provides an overview of these graph classes.

2.1 The Intersection Graph Model: The Interval Graph

Definition 2.1 (Interval Intersection Graph) *(see [Gol80]) A graph $G = (V, E)$ is an interval graph if and only if there exists a one-to-one mapping from the vertices v_i of G to intervals c_i on the line such that $\phi_I(c_i, c_j) = 1 \iff (v_i, v_j) \in E$.*

Interval graphs are intersection graphs of intervals on a line. Booth and Lueker [BL76] show that this class of graphs can be recognized in linear time. As Golumbic shows by the wide range of his sample applications, this class of graphs can be used to model real world data very easily, ranging over problems in chemistry, genetics, archaeology, and database and file structures. Interval graphs can also be used to model a wide variety of scheduling problems.

Many scheduling problems reduce to the question of colouring the interval graph. For instance, given a set of lectures which occur at certain times, determining the minimum number of rooms needed to host those lectures is equivalent to colouring the interval model of these lecture sessions. Each lecture session becomes an interval on a time line, with starting point equal to the time the lecture begins, and ending point equal to the time the lecture ends (plus possibly ten minutes to clear the room). Since it is impossible to present two different lectures at the same time in the same room, any two intervals passing through the same point must be differently coloured. Olariu [Ola91] provides a linear algorithm to colour an interval graph.

Sometimes, however, when the problem is inherently cyclic - as in for instance a weekly lecture schedule, placing the intervals on a straight line will not yield the optimal model. It is possible, of course, to place intervals, instead, on a circle, leading to the definition of the circular-arc graph class.

Definition 2.2 (Circular-arc) *(see [Gol80]) A graph $G = (V, E)$ is a circular-arc graph if and only if there exists a one-to-one mapping from the vertices v_i of G to circular arcs c_i on a circle such that $\phi_I(c_i, c_j) = 1 \iff (v_i, v_j) \in E$.*

Every interval graph is a circular-arc graph, since the underlying line can be deformed to a circle. There will be at least one point p on this circle, then, which is not covered by any interval. As Golumbic points out, the class of interval graphs is a proper subclass of the class of circular-arc graphs. The class of interval graphs excludes chordless cycles of length greater than 3, but these graphs can be modeled by circular arcs. Moreover, the model of any circular-arc graph which is not an interval graph will not contain a point p on the circle such that p is not covered by any interval.

2.1.1 Proper Intersection

Related to the intersection model is the *proper intersection graph model*. The key, and only, difference between the two is that there are no contained objects in the proper intersection model.

Definition 2.3 (Proper Intersection Graph) *An intersection graph $G = (V, E)$ is a proper intersection graph if there exists an intersection model so that $\phi_C(z_i, z_j) = 0$ for all objects $z_i, z_j \in Z$ representing all vertices $v_i, v_j \in V$.*

Due to this restriction, it is easy to see that for any class of objects, the proper intersection graph class is a subclass of the related overlap graph class of the same objects, given the definitions of ϕ_I and ϕ_O.

It is also interesting to note that both the proper interval and proper circular-arc graph classes are subclasses of the overlap or circle graph, which is defined in Section 2.3 [Gol80].

2.2 The Containment Graph Model: The Permutation Graph

Definition 2.4 (Interval Containment Graph) *A graph $G = (V, E)$ is an interval containment graph if and only if there exists a one-to-one mapping from the vertices v_i of G to intervals c_i on the line such that $\phi_C(c_i, c_j) = 1 \iff (v_i, v_j) \in E$.*

The interval containment model is similar to the interval model, and can be used to solve a similarly wide variety of problems. Consider for instance the problem of finding, given a set of square regions in the plane of varying size, the maximum sequence of regions that fit one inside the other. This problem can be solved by finding the maximum clique of the related interval containment graph. Each interval will represent one square region, and will have length equal to the length of one side of the region. All the intervals will then be placed on the line so that all the intervals are centred at one point. Then, for any two intervals, if one is contained within the other, one square region would fit inside the other. Thus a sequence of regions fitting one inside the other would be equivalent to a clique of intervals, and a maximum

clique would contain the same number of vertices as the number of containers in the maximum sequence of nested containers.

An optimal algorithm to solve the this problem may not be immediately apparent. It is useful, in fact to look at a different class of graphs, known as permutation graphs, which are defined as follows.

Definition 2.5 (Permutation Graph) *(See [Gol80]) A graph G=(V,E) is a member of the permutation graph class if there exists a permutation π of the numbers [1,2, ..., n] such that the edge $v_iv_j \in E$ if and only if $i < j$ and and $\pi(i) > \pi(j)$. (See also Figure 2.2)*

The following theorem, first proven by Dushnik and Miller, explains the interest in permutation graphs:

Theorem 2.6 *[DM41] The classes of interval containment and permutation graphs are equivalent.*

Then optimal algorithms exist to solve this problem of finding a maximum clique for permutation graphs. Felsner, Müller, and Wernisch [FMW97] present optimal algorithms for finding the maximum clique for trapezoid graphs, a graph class which includes the class of permutation graphs as a subclass. These algorithms run in $O(nlogn)$ time.

A permutation graph can be converted to an interval containment graph in $O(n)$ time. McConnell and Spinrad prove that a permutation graph can be recognized in $O(m + n)$ time[MS94].

2.3 The Overlap Graph Model: The Circle Graph

Definition 2.7 (Interval Overlap Graph) *[Gav73] A graph $G = (V, E)$ is an interval overlap graph if and only if there exists a one-to-one mapping from the vertices v_i of G to intervals c_i on the line such that $\phi_O(c_i, c_j) = 1 \iff (v_i, v_j) \in E$.*

The third and last graph model is that of the interval overlap graph, also known as the overlap graph. Gavril, in [Gav73], provides the following definition of circle graphs, and then proceeds to prove the following theorem:

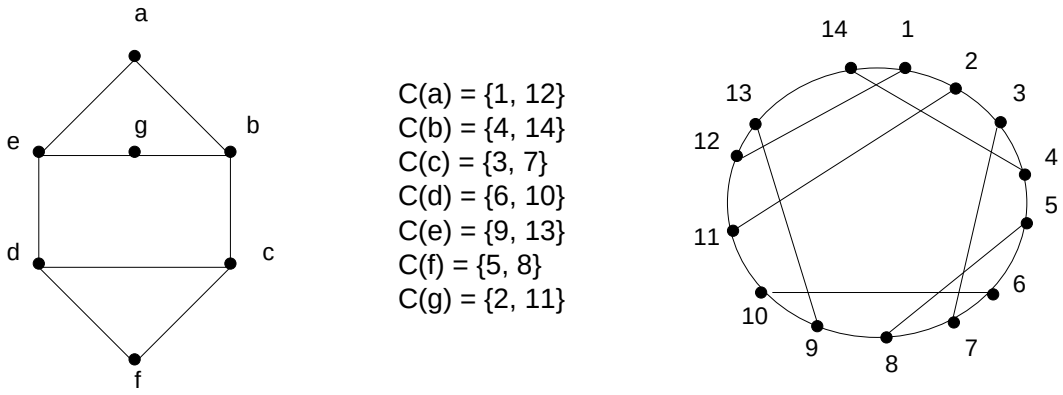


Figure 2.1: The circle graph.

Definition 2.8 (Circle Graph) *[Gav73] A graph G=(V,E) is defined to be a circle graph if and only if there exists a one-to-one mapping from the vertices in V to unique chords c_i inscribed in a circle such that two chords c_i and c_j intersect if and only if their respective vertices v_i and v_j are adjacent. (See Figure 2.1)*

Theorem 2.9 *[Gav73] The classes of interval overlap graphs and circle graphs are equivalent.*

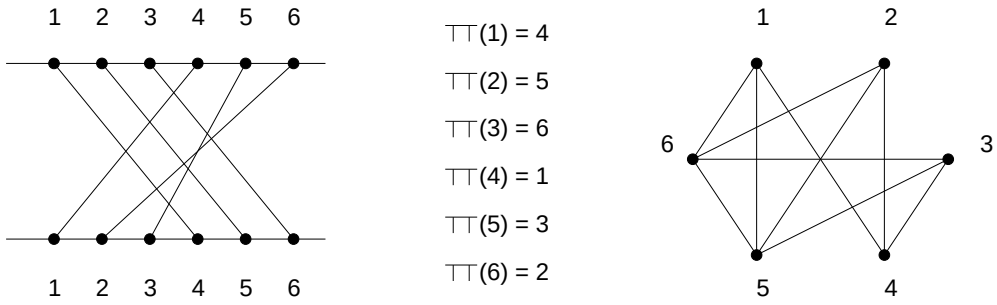


Figure 2.2: The permutation graph - chords across parallel lines.

This definition of the circle graph, and the equivalence of the circle and interval overlap graphs, leads to an odd conclusion; in the definition of the overlap graph two vertices v_i and v_j are adjacent if and only if $\phi_O(c_i, c_j) = 1$. But this implies that $\phi_C(c_i, c_j) = 0$, and $\phi_I(c_i, c_j) = 1$ or 0. And yet, the class of permutation graphs, which is equivalent to the class of interval containment graphs, is a proper subclass of the circle graph.

That the permutation graph class is a proper subclass of the circle graph class is easy to see if the permutations of the permutation graph are represented as chords

across two parallel lines, as in Figure 2.2. Then these parallel lines can be joined at the ends and deformed to a circle, resulting in a circle graph. This circle graph is, in turn, equivalent to an interval overlap graph.

This suggests that the relationships between intersection, containment, and overlap graphs defined by the same class of objects are not as intuitive as they might seem. It seems counter-intuitive that, for intervals on a line, the class of containment graphs is contained in the class of overlap graphs. Given an interval containment and its model however, it is important to note that when the interval containment model has been transformed to a permutation model, which is in turn transformed into a circle model, and thence to an interval overlap model, this resulting interval overlap model represents the same graph but consists of different intervals.

Returning to the problem of recognition mentioned in the previous sections, circle graphs can be recognized and and a circle graph model constructed in $O(n^2)$ time [Spi94]. Once constructed, a circle graph model can be transformed to an interval overlap model in $O(n)$ time.

2.4 Extending the Circle Graph

As mentioned in Section 2.3 and generally well known, the circle graph is a generalization of the permutation graph. This section is concerned with other ways of extending and generalizing the circle graph. These generalizations do not rely on the interval overlap graph model but, instead, seek to generalize the notion of intersecting chords inscribed in a circle.

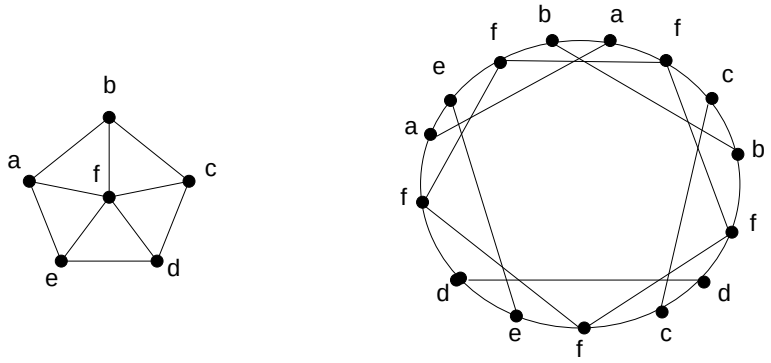
Section 2.4.3 provides an example of an alternative and previously unconsidered extension of the circle graph class. This section is included as another example of ways to extend the circle graph class.

2.4.1 The Circle Polygon Graph

One extension of the circle graph class is the circle polygon graph, the intersection graph of polygons inscribed in a circle. This extensionsion was first found in [KK97], where Kostochka and Kratochvil provide the following definition and then go on to discuss upper bounds on the number of colours used to colour a spider graph if the

colouring is not optimum.

Definition 2.10 (Circle-Polygon Graph) *[KK97] A graph G=(V,E) is defined to be a circle-polygon graph if and only if there exists a one-to-one mapping from the vertices in V to convex polygons inscribed in a circle such that two polygons intersect if and only if their respective vertices are adjacent.*



Note that in the circle polygon graph, a,b,c,d, and e are represented by 2 sided polygons, and f is represented by a 5 sided polygon.

Figure 2.3: A circle polygon graph

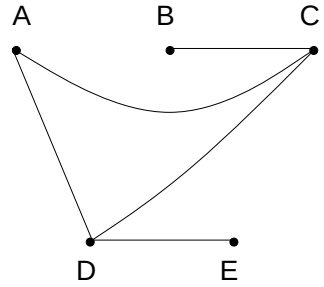
Again as in the circle graph class an alternative definition of this graph class is presented. Consider that each convex polygon consists of an ordered collection of connected non crossing chords. Then two polygons - or collections of connected chords - P_i and P_j intersect if and only if there exists at least one pair of chords (c_x, c_y), where $c_x \in P_i$ and $c_y \in P_j$ so that c_x and c_y intersect, as they do for circle graphs.

Definition 2.11 (Circle-Polygon Graph) *A graph G=(V,E) is defined to be a circle-polygon graph if and only if there exists a one-to-one mapping from the vertices in V to sets of connected non-intersecting chords inscribed in a circle such that two vertices are adjacent if and only if their respective sets of chords contain at least one pair of intersecting chords.*

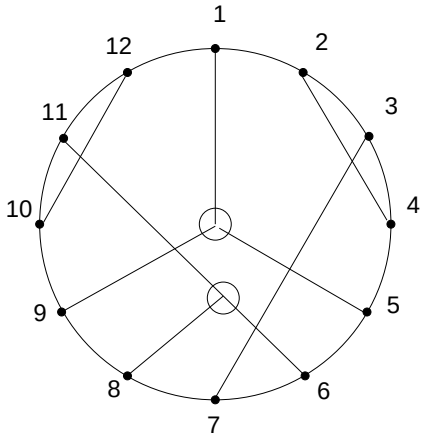
Circle graphs are obviously contained in circle polygon graphs; we can treat each chord as a 2 polygon, and the definition holds. However, as you can see from Fig-

ure 2.3, the class of circle polygon graphs includes but is larger than the class of circle graphs.

2.4.2 The Spider Graph



This graph G can be represented
in the following way as a spider graph.



A = {1, 5, 9}
B = {2, 4}
C = {3, 7}
D = {6, 8, 11}
E = {10, 12}

Figure 2.4: The spider graph

Similarly, the spider graph is another extension of the circle graph model. Consider that each chord in a circle graph intersects with the circle at two places, specifically the two endpoints of the chord. One endpoint can be arbitrarily chosen and labeled 1. Moving around the circle in clockwise direction, each successive endpoint can be numbered in increasing order from 2 to $2n$.

Then endpoint i lies between $(i+1)\ mod\ n$ and $(i-1)\ mod\ n$. Each chord has associated with it a pair of unique numbers $\{x, y\}$, $1 \leq x < y \leq 2n$, and two chords $\{x, y\}$ and $\{a, b\}$ intersect if and only if $a < x < b < y$ or $x < a < y < b$.

The spider graph extends this model by allowing two *or more* points inscribed on the circle to correspond to a given vertex. In the circle graph each vertex is described by exactly two points. For the spider graph, the points inscribed on the circle are still numbered $1, 2, \ldots, k$ in clockwise direction, starting arbitrarily at one point, but each vertex is represented by two or more points, as can be seen in Figure 2.4.

In the figure the endpoints are joined by lines - chords for those sets of points of size 2, and lines converging on a unique central point for those sets whose size exceeds 2. There are two such sets in the figure, those representing the vertices A and D in the graph, and the central point unique to each of the two sets is circled in the figure. The physical location of this central point is not unique within the circle and can be shifted in any direction, as long as the central point falls within the inscribed polygon whose vertices are those endpoints inscribed on the circle which represent this vertex, and is in terms of the definition of the spider graph completely irrelevant.

For the spider graph, then, consider a circle with points inscribed on the circle and numbered in increasing order starting at one arbitrary point and traveling around the circle in clockwise order. Then spider graphs are formally defined by Koebe as follows:

Definition 2.12 *[Koe92] A graph $G = (V, E)$ is called a spider graph if for each vertex $v_i \in V$ there exists a set of unique numbered points $C_i = \{c_{i1}, c_{i2}, \ldots, c_{ik_i}\}$ inscribed on a circle such that $(v_i, v_j) \in E$ if and only if there exist $a, b \in C_i$ and $x, y \in C_j$ so that $a < x < b < y$ or $x < a < y < b$.*

Kostochka and Kratochvil [KK97] mention in passing that the classes of circle polygon and spider graphs are equivalent, but do not provide a proof, in part because the proof is fairly straightforward.

Theorem 2.13 *The classes of spider and circle polygon graphs are equivalent.*

Proof:

Consider a graph G which is a circle polygon graph. Then each vertex v in G is represented by a polygon P_v inscribed in a circle. Choose one polygon vertex on the circle arbitrarily and start numbering the vertices from that polygon vertex on in clockwise order from $1, 2, \ldots, k$, where $k = \sum_{i=1}^{n}$(number of sides in P_{v_i}) is the

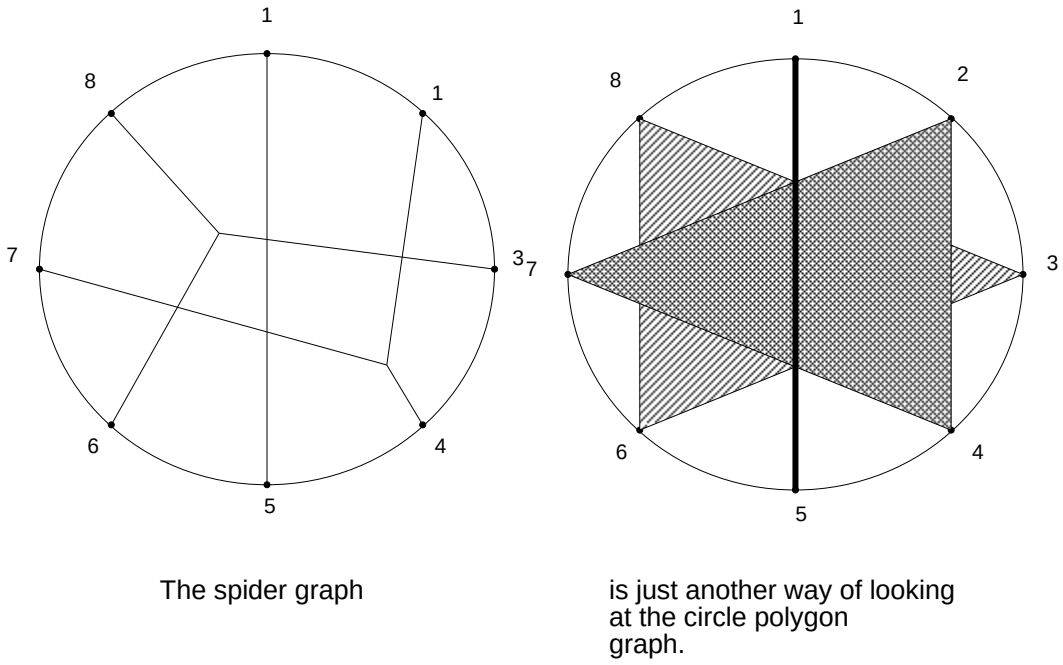


Figure 2.5: A circle polygon graph is a spider graph.

total number of polygon vertices lying on the circle. Build a spider graph by letting, for each vertex v, the collection C_v be the set of points on the circle which are the polygon vertices of P_v.

Two vertices v and w in G will be adjacent if and only if the two polygons P_v and P_w intersect. Each polygon consists of a collection of 'sides' or chords inscribed in the circle which share endpoints. Therefore the two polygons intersect if there exists at least one intersection between two chords C_v and C_w contained in the two polygons P_v and P_w respectively. Consider the endpoints of those two chords, and let $\{a, b\}$ be a chord comprising one side of P_v and $\{x, y\}$ be a chord comprising one side of P_w, where $a, b, x, y \in [1, 2, \ldots, k]$ such that these two chords intersect. Then either $a < x < b < y$ or $x < a < y < b$.

Thus, if two such chords exist in P_v and P_w, then their endpoints will be contained in C_v and C_w respectively, and these endpoints can be used to show intersection between the vertices v and w in the spider graph representation of G.

If two such intersecting chords do not exist, then P_v and P_w do not intersect. Then no polygon vertices $\{a, b\}$ exist in P_v and $\{x, y\}$ in P_w so that $a < x < b < y$ or $x < a < y < b$, and there is no intersection between C_v and C_w in the spider graph.

Therefore every circle polygon graph is a spider graph.

Now consider a graph G which is a spider graph. Then each vertex v in G is represented by a collection of points C_v inscribed on the circle. So let P_v - the polygon representing v in a circle polygon graph - be the polygon inscribed on the circle whose vertices are exactly those included in C_v. Since P_v must by definition be a convex polygon and since the vertices are numbered in increasing order clockwise around the circle, the construction of the polygon is well defined.

Then it remains only to show that v and w are adjacent if and only if P_v and P_w intersect. There exist $\{x, y\}$ in C_v and $\{a, b\}$ in C_w such that $a < x < b < y$ or $x < a < y < b$ if and only if v and w are adjacent by the definition of the spider graph. If the two vertices v and w are adjacent, then the chords $\{x, y\}$ and $\{a, b\}$ will intersect and are part of P_v and P_w respectively. On the other hand, if the two vertices v and w are not adjacent, then no such foursome of numbers will exist in C_v and C_w, and hence no chord comprising a side of P_v will intersect with any chord comprising a side of P_w.

Therefore the classes of circle polygon and spider graphs are equivalent.□

The class of circle polygon, or spider, graphs is a broad class, containing as proper subsets all trees, cycles, interval, permutation, trapezoid, and of course circle graphs[Koe92, Koe90]. As well, both Koebe and Kratochvil and Kostochka[KK97] prove that chordal graphs form a proper subclass of spider graphs. Moreover, Koebe provides a polynomial recognition algorithm for spider graphs [Koe92].

The spider graph, in this thesis, is more convenient to use than the circle polygon graph due to the straightforward way to represent the graph. Moreover, as shown in the proof for Theorem 2.13, conversion between circle polygon and spider representations of a graph can be made in $O(n)$ time complexity.

Given a spider graph representing a graph G with n vertices, so that there are k endpoints on the circle, the circle can be arbitrarily cut, and the endpoints numbered clockwise from that point around the circle in increasing order. Then each vertex v is associated with a set C_i, where C_i consists of an ordered list of numbers between 1 and k, and each number occurs in exactly one C_i. This representation of the spider graph will be assumed for the remainder of this thesis.

2.4.3 A Brief Discursion into the Realm of the Sphere Graph

Consider, briefly, the sphere graph, which is the intersection graph of planes contained within a sphere. In this graph class two planes intersect if and only if they do so within the sphere.

A proper intersection graph is an intersection graph none of whose objects are completely contained one inside another. The planes can, without loss of generality, be required to be unique, which means that the sphere graph is actually a proper intersection graph of planes in the sphere. Similarly, the intersecting chords are unique in the circle graph, and the circle graph is a proper intersection graph of chords in a circle.

Definition 2.14 (Sphere Graph) *A graph G=(V,E) is defined to be a sphere graph if and only if there exists a one-to-one mapping from the vertices in V to unqiue planes inscribed in a sphere such that two planes P_i and P_j intersect if and only if their respective vertices v_i and v_j are adjacent.*

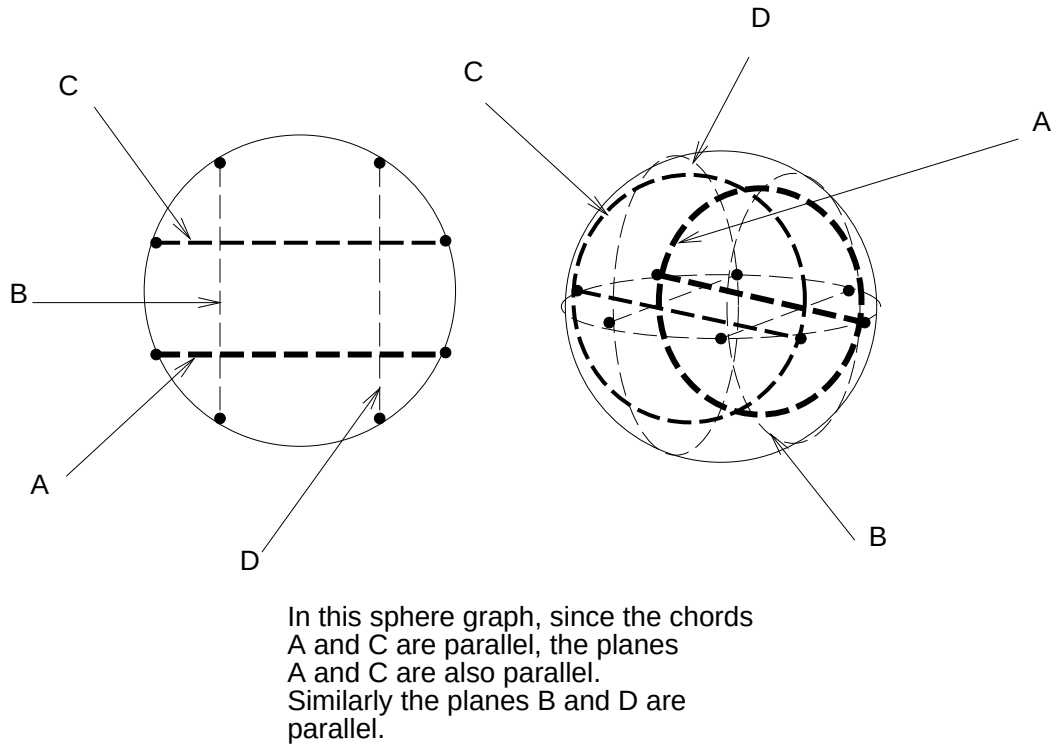


Figure 2.6: Inscribing a circle graph on a sphere graph.

Theorem 2.15 *The class of circle graphs is a proper subclass of the class of sphere graphs.*

Obviously the class of circle graphs is contained in the class of sphere graphs, since any circle graph can be converted to a sphere graph. This conversion consists of a three step process. First, the sphere is cut by one plane, whose intersection with the sphere will form a circle. The chords are then inscribed in this circle. Lastly each chord C_i is converted to a plane which cuts the circle plane along C_i and forms a 90° angle with the circle plane.

Since two inscribed planes P_i and P_j will intersect within the sphere if and only if their respective chords C_i and C_j intersect, the two representations represent the same underlying graph.

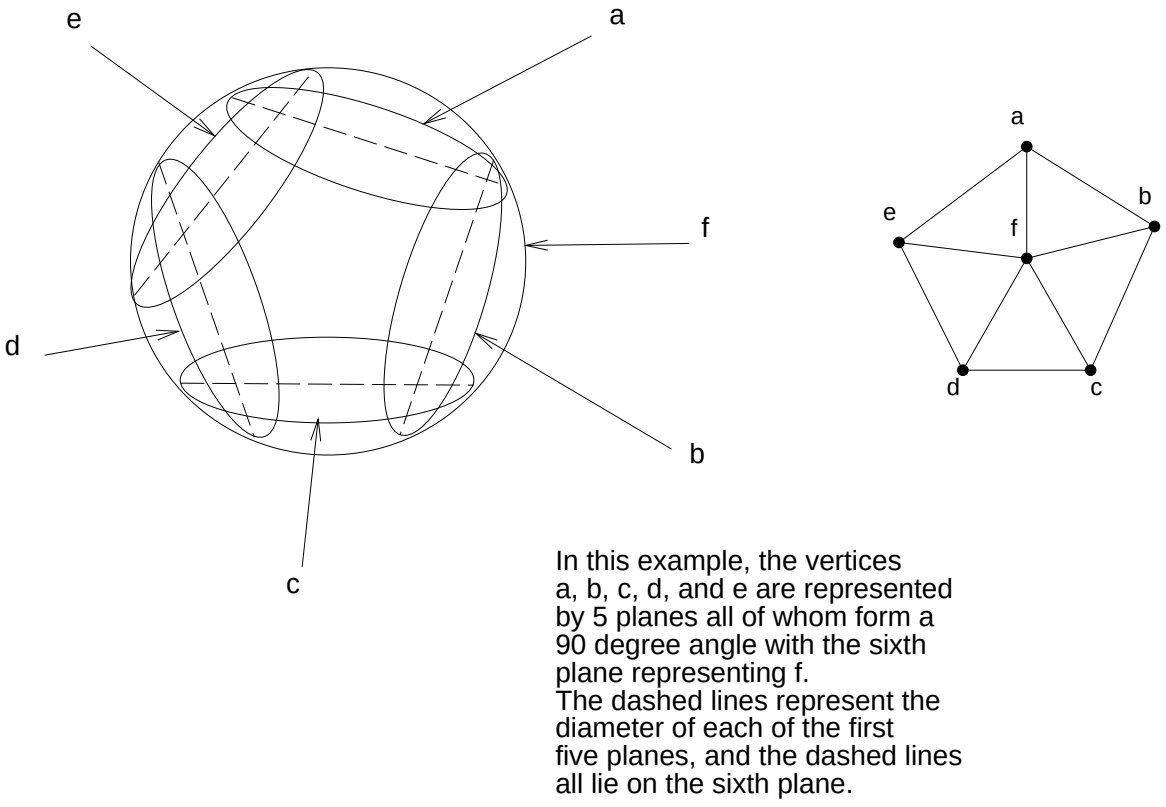


Figure 2.7: A sphere graph is not always a circle graph.

The circle and sphere graph classes are not equal, since there exist graphs which can be represented on the sphere but not the circle. One such graph consists of a cycle of length five with a sixth vertex that is adjacent to each of the cycle's vertices.

This graph cannot be represented as a circle graph[GSH89], but can be represented as a sphere graph. This can be done most simply by first constructing the sphere representation of the five cycle. Each of the five planes will then intersect the circle plane. Therefore, if the sixth vertex is mapped to the circle plane, it will be adjacent to each of the other five vertices.

Both the sphere and circle polygon graph classes contain the circle graph class as a proper subset. It is not clear at this point that the two class graphs are equivalent, and in fact I would conjecture that they are not.

A literature survey produced no references for sphere graphs, however, which suggests that no research has been done into the class of sphere graphs. As such, this class might prove interesting to explore further, in the future. A first question to address would be the question of whether this class of graphs is equivalent to any other class of graphs.

Chapter 3

Subtrees of a Tree

In Chapter 2 the intersection, containment, and overlap graphs of intervals on a line were introduced and discussed. This chapter will introduce the intersection, containment, and overlap graphs of connected subtrees of a connected tree. In each of the three graph classes, each vertex in the graph is represented by a connected subtree of the tree. It is important to note that, for the intersection and overlap models in particular, it is important to insist that the the subtrees must be connected. If the subtrees are not required to be connected, the class of graphs that can be represented by either model is equal to the class of all graphs.

Each of the three graph classes will be presented, and in the case of the subtree intersection and subtree containment graph, the equivalent graph classes will also be introduced. As well, the complexity of recognizing these graph classes will be discussed.

This chapter is fairly short, but, with Chapter 2, provides all the definitions required for Chapter 4.

3.1 The Subtree Intersection Graph Model: The Chordal Graph

Definition 3.1 (Subtree Intersection Graph) *[Gav74] A graph* $G = (V_G, E_G), |V_G| = n_G$ *is a subtree intersection graph if and only if the following holds:*
There exists a tree $T = (V_T, E_T)$ *and a collection of* n_G *connected subtrees* $T_i, 1 \leq i \leq n_G$, *so that* $\phi_I(T_i, T_j) = 1 \iff (v_i, v_j) \in E_G$, *for all* $v_i, v_j \in V_G$ *(see Figure 3.1).*

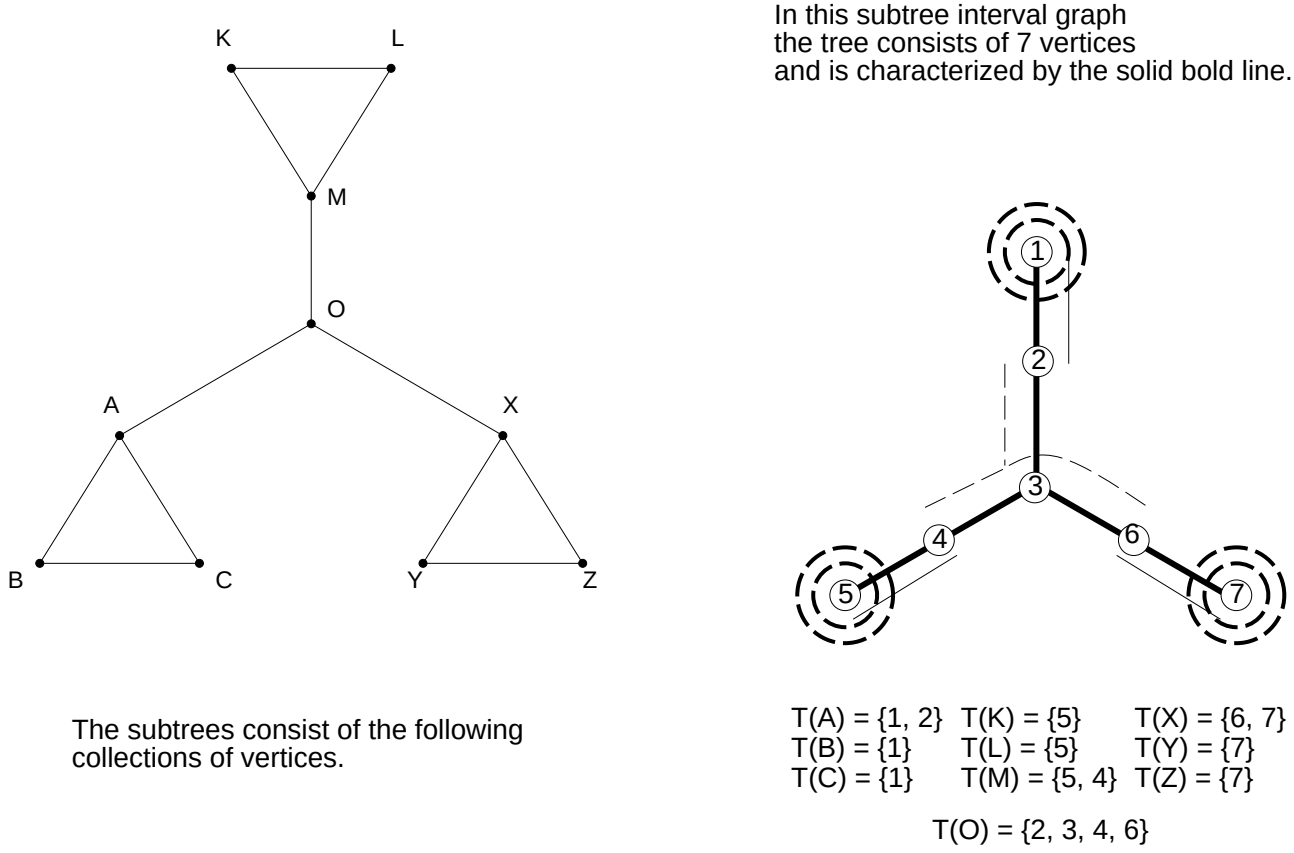


Figure 3.1: A subtree intersection graph

The first of the three graph classes modelled using subtrees of a tree is the subtree intersection graph. Gavril provides the following definition of chordal graphs, and then proceeds to prove Theorem 3.3.

Definition 3.2 (Chordal Graph) *[Gav74] A graph $G = (V, E)$ is a chordal graph if G does not contain any chordless cycles of length > 3.*

Theorem 3.3 *[Gav74] The classes of chordal and subtree intersection graphs are equivalent.*

Chordal graphs have been studied extensively. Rose, Tarjan, and Lueker [RTL76] have shown that chordal graphs can be recognized in $O(m + n)$ time.

The ϕ_I function for subtrees of a tree is defined as follows:

$$\phi_I(T_i, T_j) = \begin{cases} 1 & \text{if } T_i \cap T_j \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

3.2 The Subtree Containment Graph Model: The Comparability Graph

Definition 3.4 (Subtree Containment Graph) *A graph $G = (V_G, E_G), |V_G| = n_G$ is a subtree containment graph if and only if the following holds:*
There exists a tree $T = (V_T, E_T)$ and a collection of n_G connected subtrees $T_i, 1 \leq i \leq n_G$, so that $\phi_C(T_i, T_j) = 1 \iff (v_i, v_j) \in E_G$, for all $v_i, v_j \in V_G$.

Definition 3.5 (Comparability Graph) *An undirected graph $G = (V, E)$ is a comparability graph if it can be transitively oriented so that $x \rightarrow y$ and $y \rightarrow z$ implies $x \rightarrow z$ for all $x, y, z \in V$.*

Theorem 3.6 *[GS89] The classes of subtree containment graphs and comparability graphs are equivalent.*

Golumbic and Scheinerman discuss the class of subtree containment graphs and proves Theorem 3.6. McConnell and Spinrad [MS94] present an algorithm to recognize comparability graphs in $O(n^{2.38})$ time.

Comparability graphs, in turn, are also widely studied, and can be used for such applications as implication reduction, where a given set of implications is reduced as far as possible. Each implication is a directed edge of the graph; the graph is then modified to a comparability graph by adding the requisite number of directed edges so that the graph meets the definition's standards. At that point the shortest path between any two vertices will represent the most fully reduced implication.

3.3 The Subtree Overlap Graph Model

Lastly, consider the definition of the subtree overlap graph.

Definition 3.7 (Subtree Overlap Graph) *[Nov94] A graph $G = (V_G, E_G), |V_G| = n_G$ is a subtree overlap graph if and only if the following holds:*
There exists a tree $T = (V_T, E_T)$ and a collection of n_G connected subtrees $T_i, 1 \leq i \leq n_G$, so that $\phi_O(T_i, T_j) = 1 \iff (v_i, v_j) \in E_G$, $v_i, v_j \in V_G$ (see Figure 3.2).

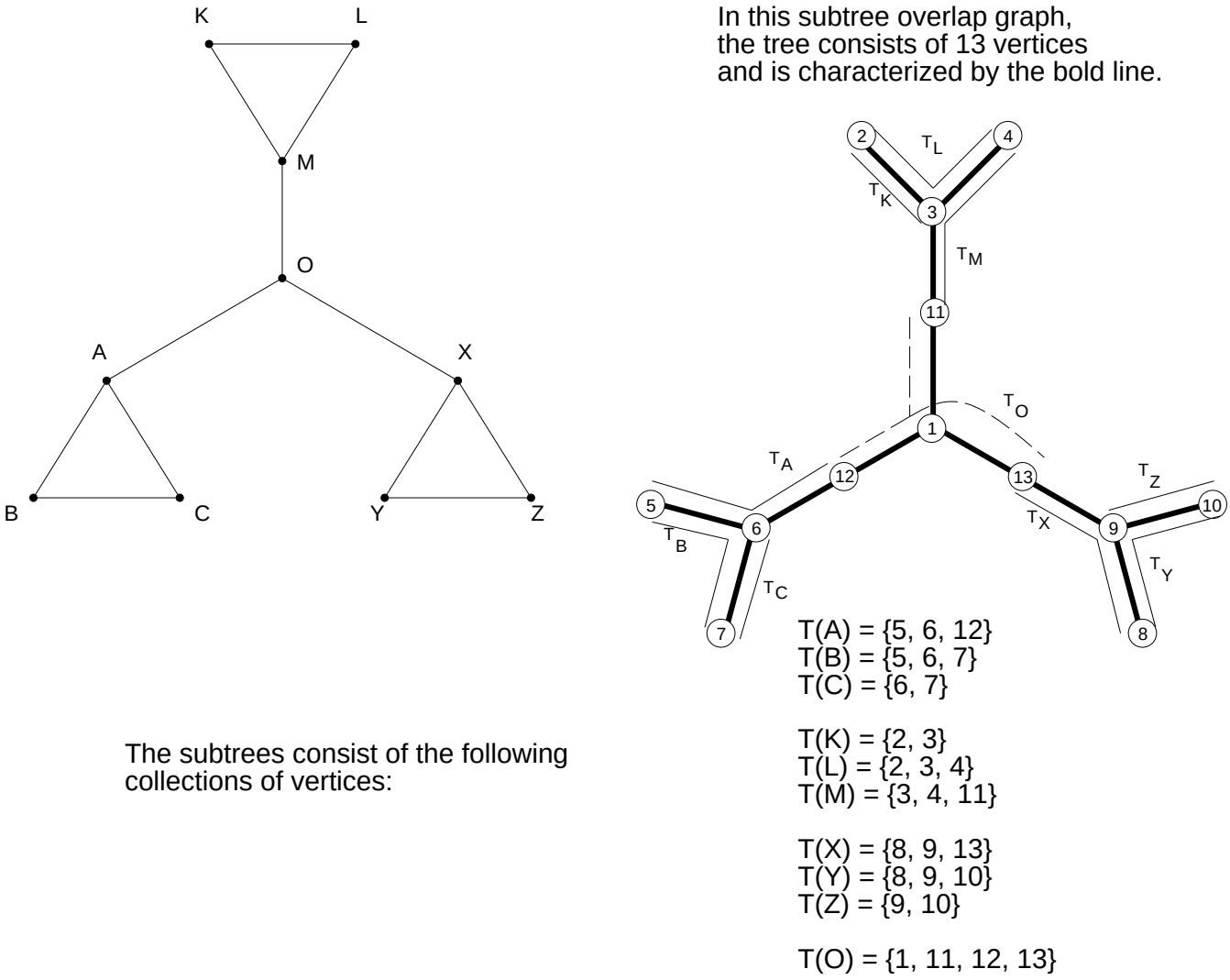


Figure 3.2: A subtree overlap graph

The definition is given in terms of the function ϕ_O which can be defined, for subtrees, as follows:

$$\phi_O(T_i, T_j) = \begin{cases} & \text{if } T_i \cap T_j \neq \emptyset \\ 1 & \text{and } T_i \cup T_j \neq T_i \\ & \text{and } T_i \cup T_j \neq T_j \\ 0 & \text{otherwise} \end{cases}$$

However, this definition of ϕ_O is somewhat unwieldy - though precise - and can be simplified to a search for three vertices a, b, and c that match the following requirements. If these vertices don't exist, then there is no overlap between the two subtrees in question.

$$\phi_O(T_i, T_j) = \begin{cases} & \exists a, b, c \in V^T \text{ such that} \\ 1 & a, b \in T_i, c \notin T_i \\ & \text{and } a \notin T_j, b, c \in T_j \\ 0 & \text{otherwise} \end{cases}$$

Novillo did some work on the structure of subtree overlap graphs, partially proving the containment of circle and chordal graphs in the class. No proof of equivalence with some known graph class has been established yet, and no recognition algorithm has been presented.

We will attempt to address both these questions, and others, in the following chapters. In particular, we are concerned with the hierarchy of graph classes, the possible recognition of the subtree overlap graph, and the algorithmic complexity of some common problems with respect to the subtree overlap graph class.

Chapter 4

Hierarchy of Classes

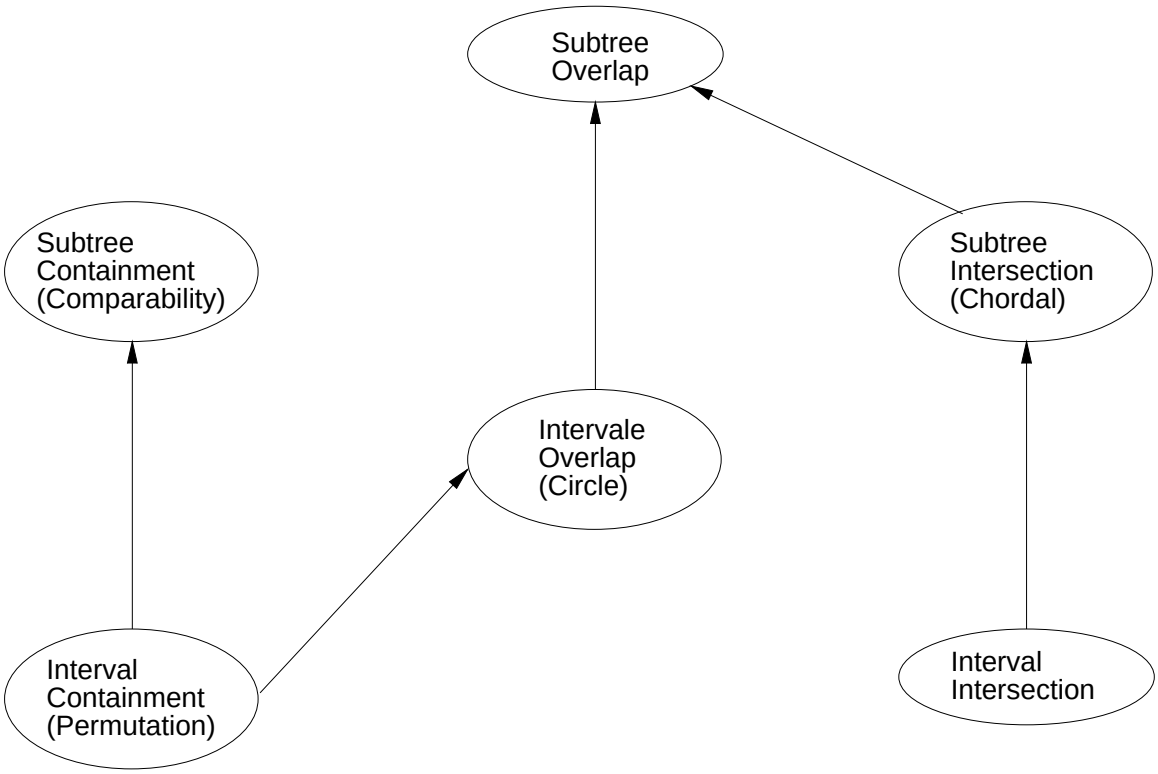


Figure 4.1: A simple hierarchy of the mentioned graph classes

Figure 4.1 shows the relationships between graphs modelled using intervals on a line and graphs modelled using subtrees of a tree, as discussed in Chapters 2 and 3. Each directed arrow between two graph classes denotes a proper subclass relationship. If an arrow leads from class A to class B, then graph class A is a proper subclass of graph class B. This means that every graph in A is a member of B, but B contains graphs that are not members of A. If no such arrow links two subclasses,

then neither class of graphs is a proper subclass of the other. Since none of the graph classes displayed in this hierarchy are equivalent, that means that two unconnected graph classes have no containment relationship.

The relationship between interval containment (permutation) graphs and interval overlap (circle) graphs is discussed in Section 2.3. Golumbic provides counter examples to show the lack of relation between interval containment (permutation) and interval intersection graphs[Gol80]. The relationship between subtree overlap and subtree intersection (chordal) graphs is proven in Theorem 4.5 (pg. 33), and the lack of relationships between the subtree containment (comparability) and the other subtree graphs is discussed in Section 4.3.2.

In general, every class of graphs modelled using intervals is a subclass of the class of graphs modelled using subtrees of a tree and the same relationship. The intervals on the line are equivalent to subpaths of a path, and hence subtrees of a tree. Moreover, each of these classes of graphs modelled by intervals on a line are well known to be proper subclasses of the related classes of graphs modelled using subtrees of a tree.

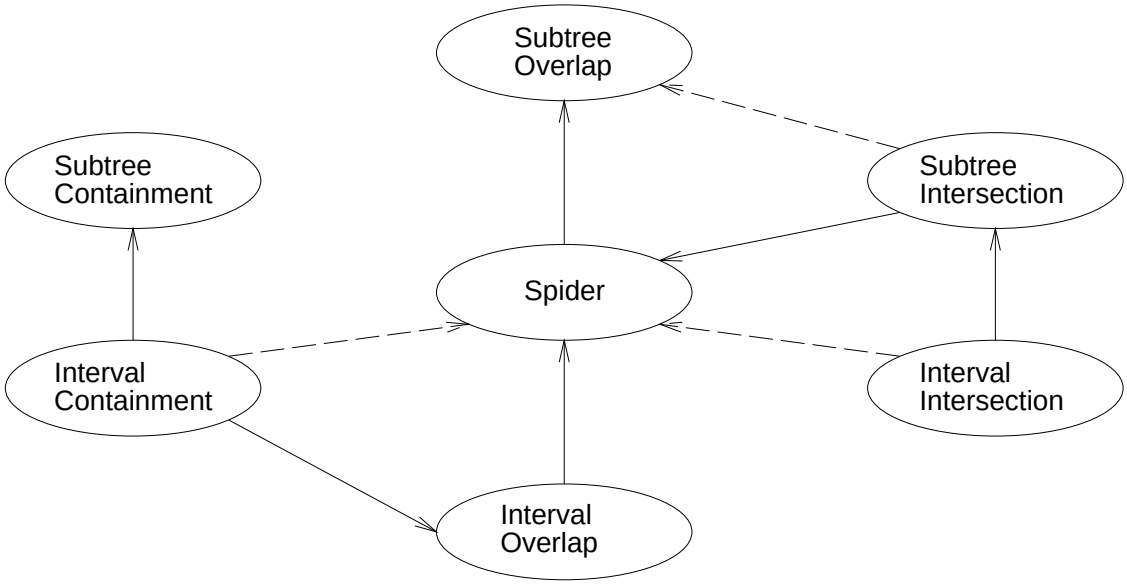


Figure 4.2: Placing the spider graph in the hierarchy of the graph classes

The hierarchy of Figure 4.1 can be refined however, by adding in the class of spider graphs, yielding the expanded diagram in Figure 4.2. The dashed lines are holdovers from the previous diagram which become unnecessary as the containment relationship is transitive.

Theorem 4.4 proves that the class of spider graphs is a proper subclass of the class of subtree overlap graphs. Moreover, the class of spider graphs includes - as proper subclasses - the classes of subtree intersection (chordal) and interval intersection graphs, and the classes of interval overlap (circle) and interval containment (permutation) graphs [Koe90, Koe92, KK97].

4.1 The Class of Spider Graphs

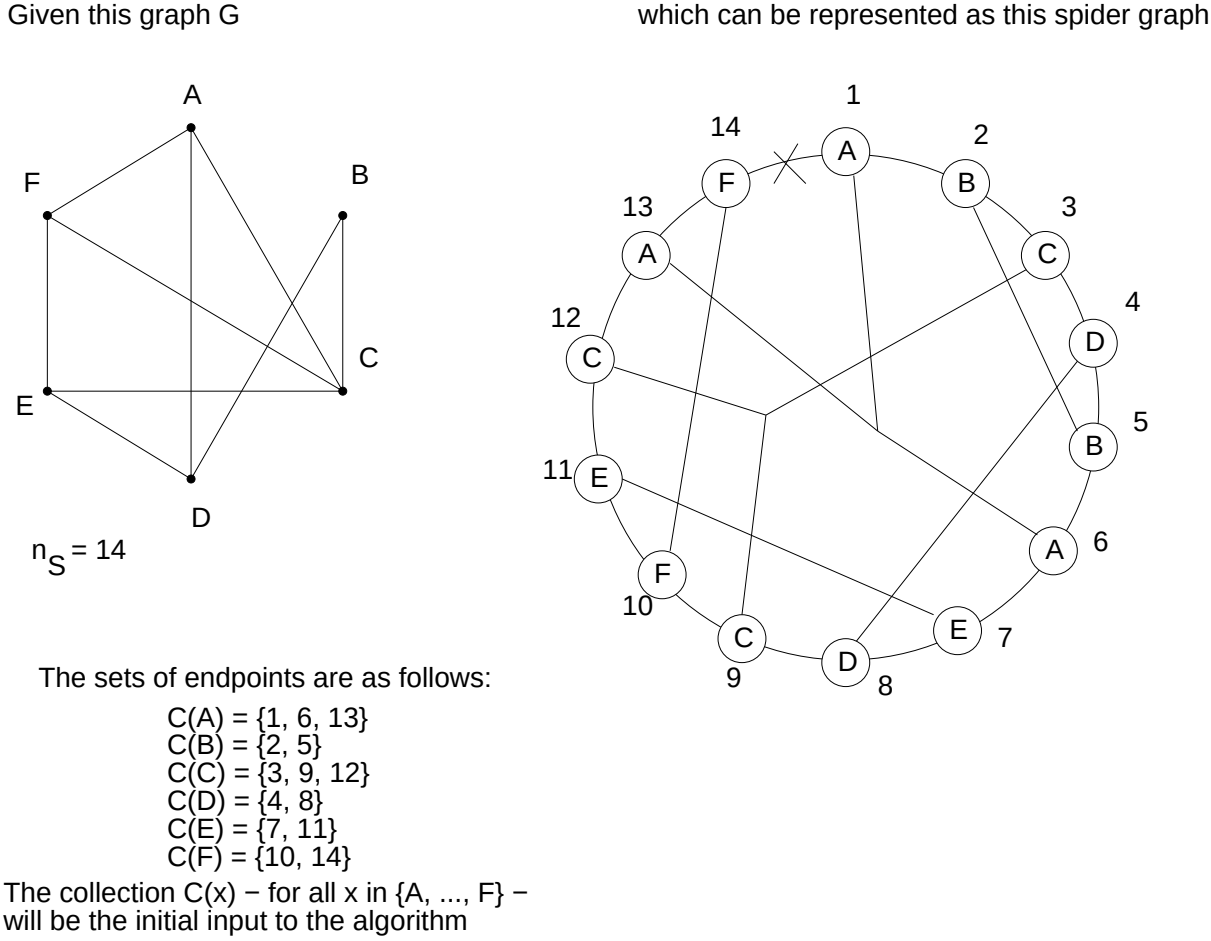


Figure 4.3: Initial input of a spider graph.

The class of spider graphs is neither a class of graphs modelled using intervals, nor a class of graphs modelled using subtrees of a tree explicitly. The class is a generalization of the class of circle graphs, and it will now be proven that the class of spider graphs is a proper subclass of the class of subtree overlap graphs.

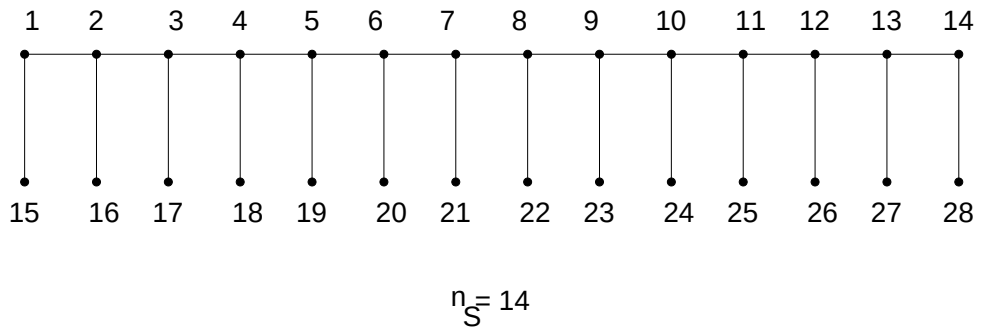


Figure 4.4: Construction of the underlying tree

Theorem 4.1 *Every spider graph is a subtree overlap graph.*

Proof:

Given a graph $G = (V^G, E^G), |V^G| = n^G$, so that G is a spider graph $S(G) = (\{C_i\}_{i=1}^{n^G})$, where each C_i corresponds to v_i in the original graph, and consists of a set $\{c_{i1}, c_{i2}, \ldots, c_{ik_i}\}$ of sorted endpoints inscribed in the circle. Let $n_S = \sum_{i=1}^{n^G} |C_i|$. Again, we can without loss of generality insist that each endpoint on the circle is contained in only one C_i, and hence is unique.

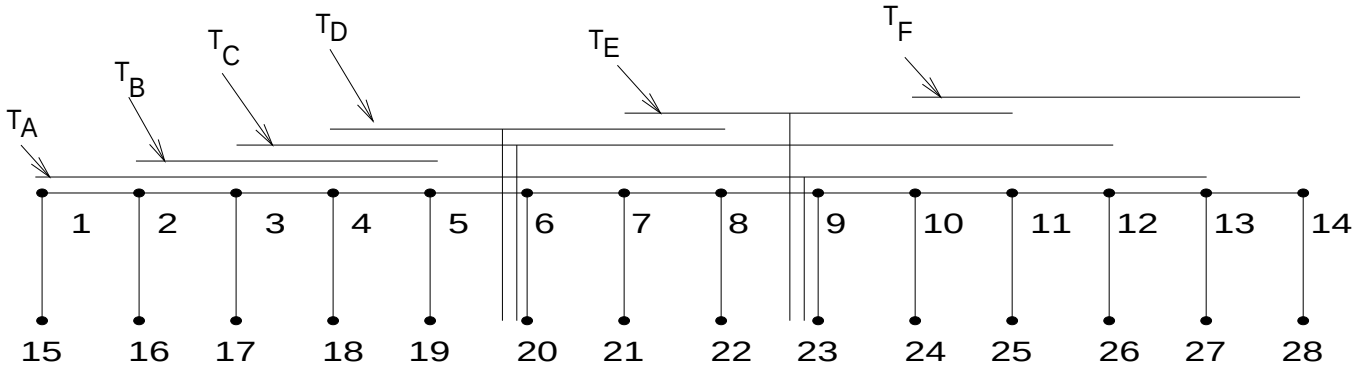
Given this structure, an arbitrary endpoint can be chosen and labeled as 1, where each next clockwise endpoint is labeled $2, 3, \ldots, n_S$. Then each C_i is first sorted so that $c_{ia} < c_{ib} \forall a < b$ before the set $\{C_i\}$ is sorted so that $\forall i, j, i < j \Rightarrow c_{i1} < c_{j1}$, resulting in input as in Figure 4.3. This input is sorted in lexicographical order both within each collection of endpoints, and within the set of collections of endpoints.

To show that G is a subtree overlap graph, we need to construct a tree $T = (V^T, E^T)$ and a set of n^G subtrees T_i, so that each T_i represents v_i, and $\phi_O(T_i, T_j) = 1 \iff (v_i, v_j) \in E^G$.

Each T_i will consist of a set of vertices of T, so that the graph induced by T_i on T is a connected tree. For each subtree we will also store a set of interior endpoints $I_i = \{c_{i2}, \ldots, c_{i(k_i-1)}\}$, where $I_i = \emptyset$ if $k_i = 2$.

ALGORITHM:

- Construct a tree $T = (V^T, E^T)$ consisting of a path of length $n_S - 1$ whose vertices are numbered $1, 2, ..., n_S$, so that each vertex $v_i^T \in V^T, i = 1 \ldots n_S$ has one neighbour $v_{i+n_S}^T$ which is adjacent only to v_i, as for example in Figure 4.4.



The tree T representing the original graph consists of 28 vertices (12 of which we can actually prune away.)

The subtrees representing the vertices of the graph consist of the following collections of vertices:

T(A) = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 23}
T(B) = {2, 3, 4, 5}
T(C) = {3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 20}
T(D) = {4, 5, 6, 7, 8, 20}
T(E) = {7, 8, 9, 10, 11, 23}
T(F) = {10, 11, 12, 13, 14}

Which matches up correctly with the original graph G:

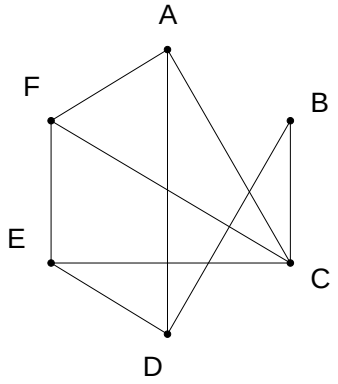


Figure 4.5: Converting a spider graph to a subtree overlap graph.

- For $i = 1 \ldots n^G$ do

  1. Set T_i = the path from c_{i1} to c_{ik_i}
  2. Set $I_i = C_i - \{c_{i1}, c_{ik_i}\}$
  3. For $j = 1 \ldots i-1$ do
     (a) For each point x in I_i, if $x \in T_j$ then $T_j = T_j \cup \{v^T_{x+n_S}\}$
     (b) For each point y in I_j, if $y \in T_i$ then $T_i = T_i \cup \{v^T_{y+n_S}\}$

After completion of this algorithm, T and T_i, $1 \leq i \leq n$, form a subtree overlap model representing G, as in Figure 4.5.

We can prove the accuracy of this algorithm using induction, where we want to show that for any $i \leq n^G$ the subtrees $T_j, j \leq i$ are a correct representation of the subgraph induced on G by the set of vertices $\{v^G_j | 1 \leq j \leq i\}$.

The base case, $i = 1$, is trivially true.

Now, assume this hypothesis holds for i. We want to show that the hypothesis still holds after we add the subtree T_{i+1}.

Having followed the algorithm to construct T_{i+1}, and modified all $T_j, j \leq i$ as required, consider each of the i vertices in turn.

For each vertex $j, 1 \leq j \leq i$ there are two possibilities:

1. $\phi_O(T_j, T_{i+1}) = 0$, which is to say that one of the following holds:
   (a) $T_j \cap T_{i+1} = \emptyset$
   (b) $T_j \subset T_{i+1}$
   (c) $T_{i+1} \subset T_j$
2. $\phi_O(T_j, T_{i+1}) = 1$

Then, for each j, we want to prove that condition 2 holds if and only if $(v_j^G, v_{i+1}^G) \in E^G$, or, alternatively, that $(v_j^G, v_{i+1}^G) \notin E^G$ if and only if condition 1 holds. Since the two are mutually exclusive, it is enough to prove the first *if and only if* statement, reflecting condition 2, which we will prove using the following two lemmas.

Lemma 4.2 $(v_j^G, v_{i+1}^G) \in E^G$ *implies* $\phi_O(T_j, T_{i+1}) = 1$

Assume $(v_j^G, v_{i+1}^G) \in E^G$. Then by the definition of the spider graph there exist $a, b \in C_j$ and $x, y \in C_{i+1}$ so that $a < x < b < y$, or $x < a < y < b$. We can ignore the case where $x < a < y < b$ since if this is true there also exist other a, b, x, y so that $a < x < b < y$ or $c_{(i+1)1} < c_{j1}$ which would contradict the inductive nature of the algorithm, given that $j < i + 1$.

In fact, we can let $a = c_{j1}$ and $x = c_{(i+1)1}$. We know, then, that $a < x$ since the C_i are sorted by the first element and $j < i + 1$. Then we only need to find a b and a y so that $a < x < b < y$. We can also let $y = c_{(i+1)k_{i+1}}$ without loss of generality, since this is the maximum possible for y in any case. Lastly, let b be the smallest c_{jx} so that $1 < x \leq k_j$ and $b > x$. Such a b will exist, given the assumption and the definition of the spider graph.

Then to show $\phi_O(T_j, T_{i+1}) = 1$, we need to establish the following three points:

1. T_j and T_{i+1} have at least one vertex $x^T \in V^T$ in common.

2. There exists a vertex $y^T \in V^T$ such that $y^T \in T_j$ but $y^T \notin T_{i+1}$

3. There exists a vertex $z^T \in V^T$ such that $z^T \notin T_j$ but $z^T \in T_{i+1}$

Consider first point 1; since T_j and T_{i+1} will minimally have the path from v_x^T to v_b^T in common after the algorithm is applied, they certainly have at least one vertex in common.

Similarly, for point 2, since $a < x$, this guarantees that by the algorithm $v_a^T \in T_j$ but $v_a^T \notin T_{i+1}$ since a is smaller than the left most (and hence smallest) endpoint in C_{i+1}.

Point 3 leads to two possibilities. If $y > c_{jk_j}$, which is to say the largest endpoint of C_{i+1} exceeds the largest endpoint of C_j, then $v_y^T \in T_{i+1}$ by the algorithm, but $v_y^T \notin T_j$ and so $z^T = v_y^T$ satisfies the condition. If $y < c_{jk_j}$ then it follows that b, which is less than y, must be an interior endpoint of v_j and hence $b \in I_j$. Then, by the algorithm, $v_{b+n_S}^T \in T_{i+1}$ but $v_{b+n_S}^T \notin T_j$ and $z^T = v_{b+n_S}^T$ satisfies the condition.

Since all three points are satisfied, we can conclude that $\phi_O(T_j, T_{i+1}) = 1$. □

Lemma 4.3 *$\phi_O(T_j, T_{i+1}) = 1$ implies $(v_j^G, v_{i+1}^G) \in E^G$, $j < i + 1$.*

Assume $\phi_O(T_j, T_{i+1}) = 1$. Then we want to show that this implies the existence of $a, b \in C_j$ and $x, y \in C_{i+1}$ such that $a < x < b < y$. If such a, b, x, and y exist, then $(v_j^G, v_{i+1}^G) \in E^G$ by the definition of the spider graph.

Let $a = min\{v^T; v^T \in T_j\}$. Since T_j includes the path $c_{j1}, \ldots, c_{jk_j}$, and c_{j1} is guaranteed by the algorithm to be the minimum vertex in T_j, this means $a = c_{j1}$. Similarly, let $x = min\{v^T; v^T \in T_{i+1}\}$, which implies that $x = c_{(i+1)1}$. Moreover, since the construction of the subtrees works from smaller to larger, the minimum vertex in T_j is guaranteed to be smaller than the minimum vertex in T_{i+1} since $j < i + 1$, and therefore $a < x$.

Let $y = max\{v^T; v^T \in T_{i+1}, v^T \leq n_S\}$, which is to say let y be the largest vertex in T_{i+1} that is not one of the leaves. Then $y = c_{(i+1)k_{i+1}}$, and $x < y$ since each collection of endpoints C_q representing the vertex $v_q^G \in V^G$ contains at least two distinct endpoints which have been sorted.

It remains only to find a $b \in C_j$ such that $x < b < y$. There are two possible cases to consider:

1. $max\{v^T; v^T \in T_j, v^T \leq n_S\} < y$
2. $max\{v^T; v^T \in T_j, v^T \leq n_S\} > y$

Each subtree consists of a path of length ≥ 1 which travels from the smallest to the largest endpoint, and possibly includes a collection of leaves. Since each endpoint is unique, we know that the third case, $max\{v^T; v^T \in T_j, v^T \leq n_S\} = y$ will never occur.

So let us first consider case 1. In this case, let $b = max\{v^T; v^T \in T_j, v^T \leq n_S\} = c_{jk_j}$. Then $b < y$. But we know $b > x$, otherwise $\phi_O(T_j, T_{i+1}) \neq 1$, which would contradict the assumption.

Let us then consider the other case, case 2. In this case, since $\phi_O(T_j, T_{i+1}) = 1$ and using the above algorithm, T_j must contain a vertex v_b^T such that the following hold:

- $b > x$
- $v_{b+n_S}^T \in T_{i+1}$
- $v_{b+n_S}^T \notin T_j$
- $v_b^T \in T_{i+1}$ and $v_b^T \in T_j$

At least one such b exists since otherwise T_{i+1} would be fully contained in T_j which would mean $\phi_O(T_j, T_{i+1}) \neq 1$. Given the way the algorithm works, $b \in I_j$, and hence $b \in C_j$. Moreover, $b < y$ or v_b^T would not be in T_{i+1}. Similarly, $b > x$, and $x < b < y$.

To sum up, we can choose four endpoints a, b, x, and y so that $a < x < b < y$, where $a, b \in C_j$, and $x, y \in C_{i+1}$. But, by the definition of the spider graph, this means that C_j and C_{i+1} intersect, and hence $(v_j, v_{i+1}) \in E^G$. □

Therefore, by Lemma 4.2 and Lemma 4.3, the induction proof holds, and therefore Theorem 4.1 is true.

□

Therefore the class of spider graphs is certainly contained in the class of subtree overlap graphs, begging only the question of whether the two classes are equal or the class of subtree overlap graphs is larger. The following theorem shows that while every spider graph is a subtree overlap graph, not every subtree overlap graph is a spider graph.

Theorem 4.4 *The class of spider graphs is a proper subset of the class of subtree overlap graphs.*

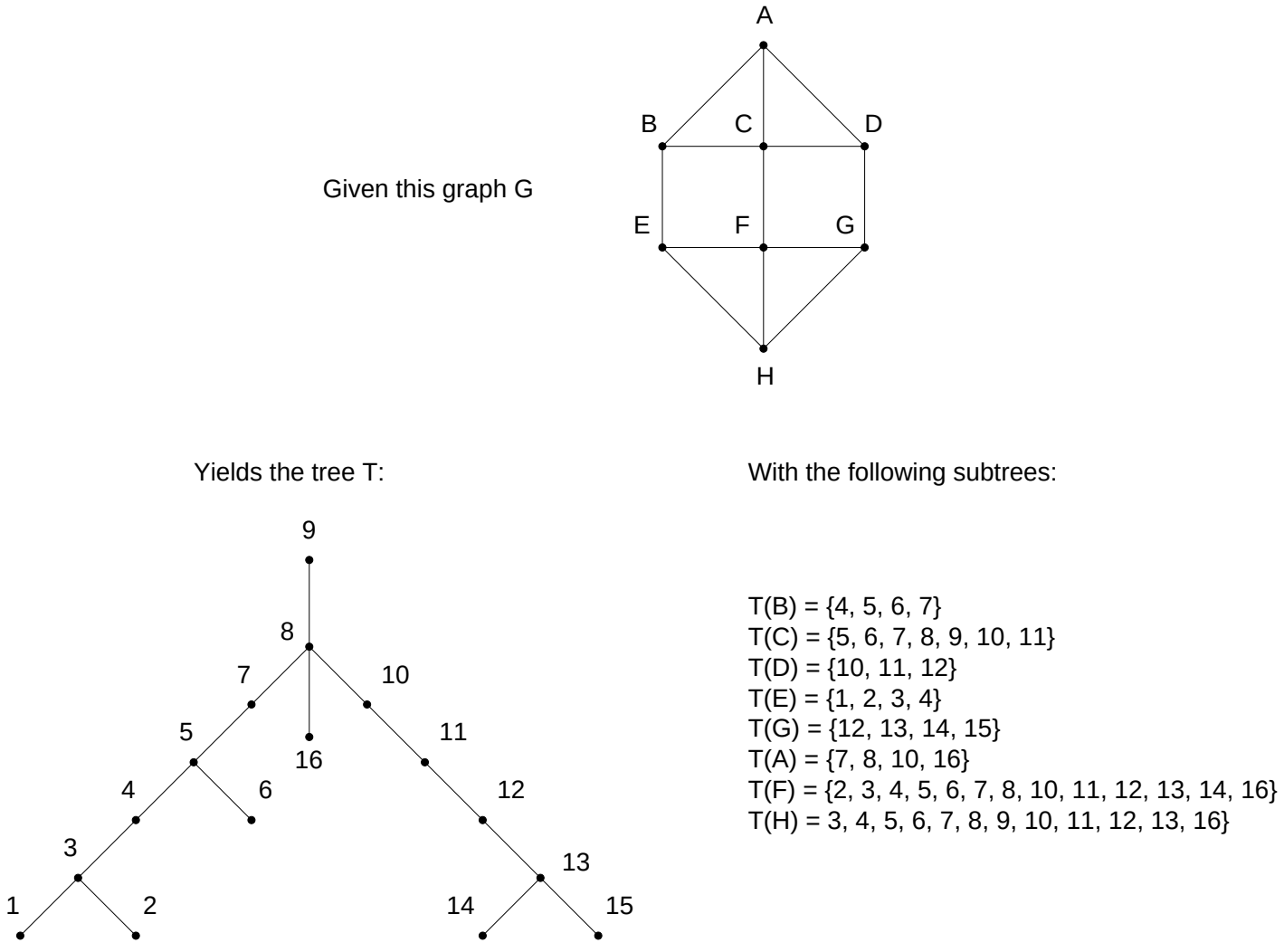


Figure 4.6: Not every subtree overlap graph is a spider graph.

The graph G shown in Figure 4.6 is not a spider graph [Koe92], but as can be seen from the tree T and its collection of subtrees, G can be represented as a subtree

overlap graph. Therefore there exists a graph G that is a subtree overlap graph, and not a spider graph, which proves the theorem. □

4.2 The Class of Subtree Intersection Graphs

This section considers the class of subtree intersection graphs and the relationship between this class and the class of subtree overlap graphs. As mentioned previously, both Koebe and Kostochka and Kratochvil [Koe92, KK97] prove that chordal graphs form a proper subclass of spider graphs, and hence immediately form a proper subclass of subtree overlap graphs.

However, it is interesting to consider this alternative proof, which takes advantage merely of the variation between the definitions of the ϕ_O and ϕ_I functions, and the ability of the model of a subtree intersection or overlap graph to be modified as needed.

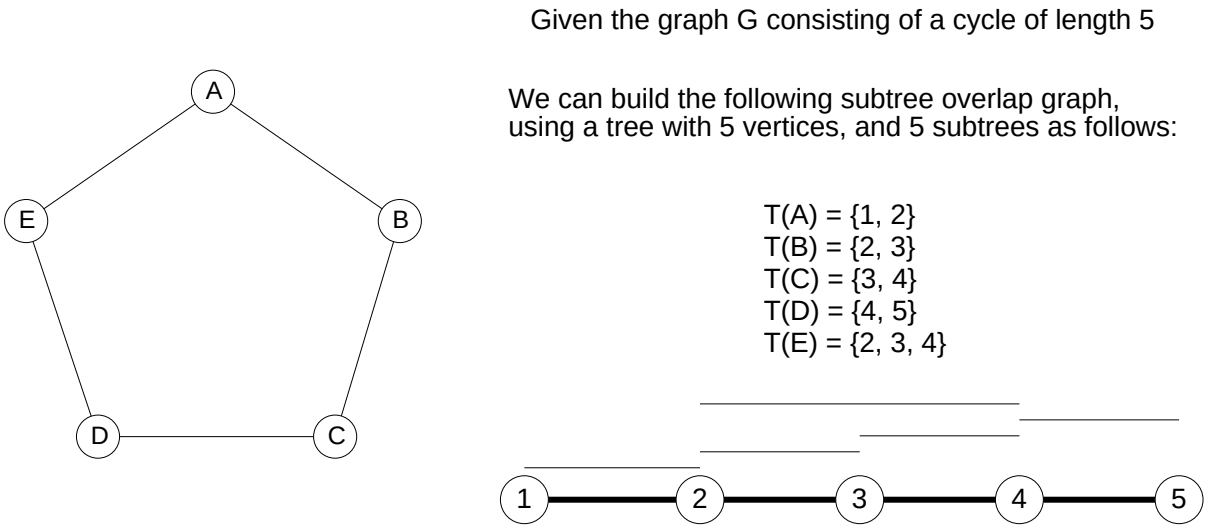


Figure 4.7: A subtree overlap graph which is not a subtree intersection graph

Theorem 4.5 *The class of subtree intersection graphs is a proper subset of the class of subtree overlap graphs.*

Proof:

Gavril, in [Gav74], proves that every subtree intersection graph is a proper subtree intersection graph, which is to say that the underlying tree and collection of subtrees

of any subtree intersection graph can be transformed so that no subtree is contained in any other subtree. Therefore, every subtree intersection graph is a subtree overlap graph.

Consider the graph G consisting of a chordless 5 cycle C_5. We know that G is not a subtree intersection graph, for G is not a chordal graph, and the classes of subtree intersection and chordal graphs are equivalent [Gol80, Gav74]. C_5 is a circle graph and hence a subtree overlap graph. So as Figure 4.7 shows, G is a subtree overlap graph.□

Gavril takes advantage of the fact that the subtrees used to model the vertices of the graph are dynamic and can be modified as needed as long as the edge relationships are maintained. Where intervals on the line are narrowly circumscribed and can be extended only to the left or right, the tree and the subtrees can be modified by adding extra branches, or extra leaves, or even by adding entire new trees to the model. Similarly, as mentioned later, it is possible in some cases to prune the tree, without modifying the edge dependencies created by the subtrees.

4.3 Graphs That Are Not Subtree Overlap Graphs

The previous sections concentrated on those graph classes contained in the class of subtree overlap graphs. A similarly interesting question is that of exclusion; which graphs are not subtree overlap graphs.

4.3.1 The Class of Graphs That Are Not Subtree Overlap Graphs

Novillo[Nov94] discusses the class of all graphs which are not subtree overlap graphs, and provides the following interesting theorem.

Theorem 4.6 *[Nov94] A graph G is not a subtree overlap graph if G contains two induced subgraphs G_1 and G_2 such that*

1. *G_1 is a cycle C_k with $k \geq 5$*

2. *G_2 is a connected graph with $|G_2| \geq k$*

3. *Every vertex v_i in G_1 is adjacent to a different vertex v_x in G_2 and v_x is not adjacent to any vertex other than v_i in G_1*

Based on Theorem 4.6, Figure 4.8 is not a subtree overlap graph. However, Theorem 4.6, as Novillo notes, is not complete, and more work remains to be done on graphs that are not subtree overlap graphs.

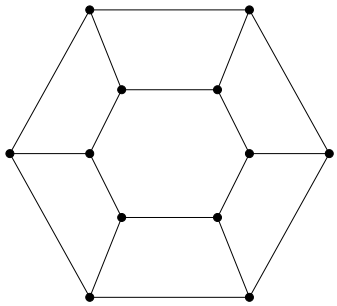


Figure 4.8: This graph is not a subtree overlap graph

4.3.2 The Class of Subtree Containment or Comparability Graphs

The class of comparability graphs and the class of subtree overlap graphs share some graphs - such as for instance the cycles C_{2k} of even size - but neither class is contained in the other. Similarly, the class of comparability graphs intersects the class of chordal graphs, but neither class is fully contained in the other. Both of these statements will be proven simply, using counter examples.

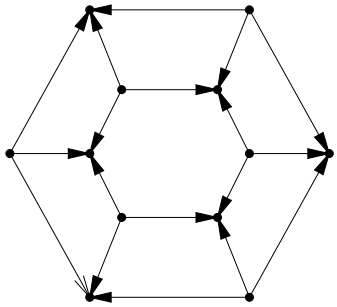


Figure 4.9: Transitively oriented version of Figure 4.8.

Lemma 4.7 *The class of comparability graphs is not a subclass of the class of subtree overlap graphs.*

Proof: Consider Figure 4.9, which is a transitively oriented version of Figure 4.8. This graph is therefore a comparability graph, but is not a subtree overlap graph by Theorem 4.6.□

Lemma 4.8 *The class of subtree overlap graphs is not a subclass of the class of comparability graphs.*

Proof: Consider the cycle C_5. This cycle is not a comparability graph since it can't be transitively oriented. However, as shown in Figure 4.7 on page 33, C_5 is a subtree overlap graph.□

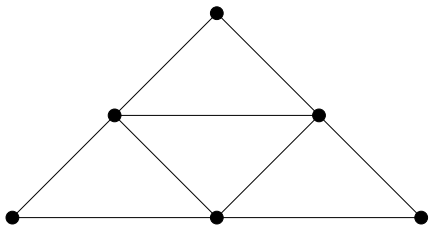


Figure 4.10: Chordal but not a comparability graph

The relationship between chordal and comparability graphs is well-known, and discussed by Golumbic[Gol80]. In short, C_4 is a comparability but not a chordal graph, and Figure 4.10 is a chordal, but not a comparability graph.

Chapter 5

The Structure of the Subtree Overlap Graph

The previous Chapter shows that certain classes of interval and subtree graphs are subclasses of the subtree overlap graph class. Each of these subclasses is recognizable in polynomial time, which begs the question of the difficulty of recognizing the subtree overlap graph.

Complexity of Recognition Problem	*Graph Class*	*Subclass of class of Subtree Overlap Graphs*
$O(n+m)$ [BL76]	Interval	Y
$O(m+n)$ [MS94]	Permutation	Y
$O(n^2)$ [Spi94]	Circle	Y
Polynomial [Koe92]	Spider	Y
$O(m+n)$ [RTL76]	Chordal	Y
$O(n^{2.38})$ [MS94]	Comparability	N
Unknown	Subtree Overlap	–

Table 5.1: Complexity of recognizing the different graph classes.

This hierarchy of subclasses does not immediately lead to a characterization of the subtree overlap graph class, naturally. Such a characterization has not been developed yet, and will be the topic of future research.

This chapter is concerned with the structure of the subtree overlap graph and its associated model. When dealing with a model, the representation of the model in memory needs to be addressed, and is discussed in Section 5.1. Section 5.2 discusses the size of the model used to model a subtree overlap graph and proves an upper bound on the size of the model.

5.1 Representing the Subtree Overlap Graph

Chapter 7 discusses several computational problems and their computational complexity for the class of subtree overlap graphs. Therefore a schema is needed to represent the subtree overlap graph in memory.

The circle graph is frequently represented in memory as n pairs of integers $C_i = \{c_{i1}, c_{i2}\}$ where $1 \leq i \leq n$, $1 \leq c_{i1} < c_{i2} \leq 2n$, and $c_{ia} = c_{jb} \iff ia = jb$. This presentation is produced in Spinrad's recognition algorithm for circle graphs[Spi94], or can be built given a circle and a set of chords by arbitrarily choosing one endpoint of one chord and assigning it the number 1, and then moving around the circle in clockwise fashion, numbering each next endpoint in increasing order.

Similarly, the spider graph can be represented as sets C_i of points inscribed on the circle, as Koebe does[Koe92], where the points are again labeled in increasing integers starting from one arbitrarily selected point and moving clockwise, and labeled $1, 2, \ldots k$. Here $k = \sum_{i=1}^{n} |C_i|$.

In both the circle and spider graph cases, however, the underlying structure on which the endpoints lie is known to be the circle. The circle is a static structure and knowing the relative ordering of the endpoints as they are inscribed in the circle is sufficient to reconstruct the graph. The underlying structure of the subtree overlap graph, on the other hand, is a tree. And, unlike the structure of the circle, the structure of the tree is dynamic as the tree can be modified by adding more vertices to the tree and possibly to the subtrees, in any way which preserves the edge relationship between subtrees. Therefore two pieces of data must be recorded in any schema: the structure of the underlying tree, and the portion of the underlying tree covered by each subtree.

In this thesis a subtree overlap graph will be represented as a tree $T = (V^T, E^T)$ which is stored as a standard graph, including vertices and edges, and as a set $\{T_i\}$ of subsets of V^T, where each subtree T_i is the subgraph of T induced by T_i. One example of such a represention is Figure 4.6 on page 32.

With this schema, it is required that each subtree be connected. Moreover, the evaluation of $\phi_O(T_i, T_j)$ now becomes a search within the two sets of vertices in T_i and T_j for a triplet (a, b, c) such that $a, b \in T_i, c \notin T_i$ and $a \notin T_j, b, c \in T_j$. If at least

one such triplet exists then the two subtrees intersect, but neither is contained in the other. If no such triplet exists then either the subtrees fail to intersect and there doesn't exist any vertex which lies in both, or one subtree is contained in the other. Such a search can be made in $O(|V^T|)$ time. One of the first questions addressed is therefore the maximum size of this underlying tree.

5.2 Limiting the Number of Vertices in the Underlying Tree

This section is concerned with the size of the tree used to model a subtree overlap graph. Specifically, an algorithm will be presented to reduce any subtree overlap model to a minimal subtree overlap model, and an upper bound on the size of this resulting minimal subtree overlap model will be proven. But first, naturally, the concept of a minimal subtree overlap model needs to be defined.

Definition 5.1 (Spare Vertex) *Given a subtree overlap model consisting of a tree $T = (V^T, E^T)$ and a set of subtrees $\{T_i\}$, a vertex $v \in V^T$ is a spare vertex if for every subtree T_i containing v the following holds:*
For every subtree $T_j, i \neq j$ such that $\phi_O(T_i, T_j) = 1$

- *if $v \in T_j$ then there exists a $w \in V^T, w \neq v$ such that $w \in T_i$ and $w \in T_j$*
- *if $v \notin T_j$ then there exists a $w \in V^T, w \neq v$ such that $w \in T_i$ and $w \notin T_j$*

Definition 5.2 (Minimal Subtree Overlap Model) *A subtree overlap model consisting of a tree $T = (V^T, E^T)$ and a set of subtrees $\{T_i\}$ is minimal if the model does not contain any spare vertices.*

In Figure 5.1, the subtree overlap model is not minimal. In this model, every vertex except 3 is spare. Note that 8 is a spare vertex since it is not contained in any subtree. Removing all the spare vertices in one swoop from the model would result in a model with no spare vertices, but also one which no longer accurately represented the graph. On the other hand, if each vertex is examined in turn to see whether it is still spare and, if so, immediately removed from the model, then certain vertices will cease to be spare before they can be removed. For instance, given Figure 5.1 and

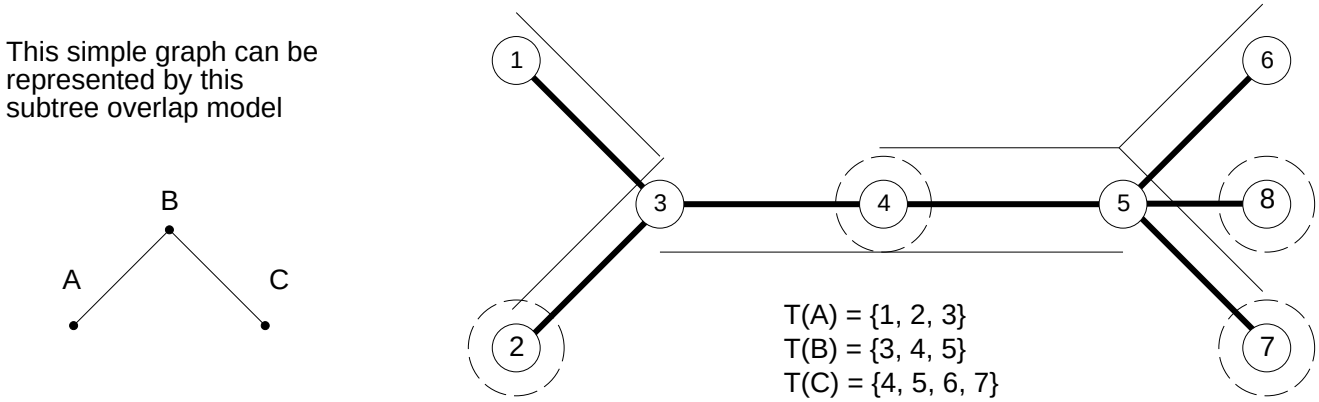


Figure 5.1: A non-minimal subtree overlap model.

removing spare vertices, one after another; Vertex 8 can be removed first; it is not contained in any subtree, and hence irrelevant. If vertex 7 is removed, then vertex 6 ceases to be spare and is not removed. Vertex 4 is still spare, and can be removed next, but if 4 is removed, 5 ceases to be spare. Vertex 2 is also still spare, so is removed next. When 2 is removed, 1 ceases to be spare, and so is not removed either. The final minimal model which results after vertices are examined in this particular order is shown in Figure 5.2. For the purpose of this algorithm it is sufficient that the vertices are examined iteratively in some arbitrary order, and it should be noted that the resulting model is minimal, but not necessarily minimum.

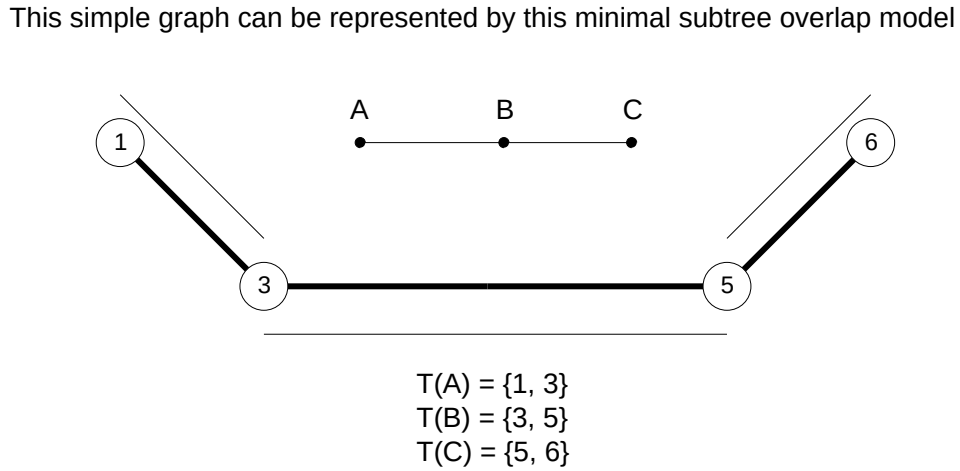


Figure 5.2: The subtree overlap model is now minimal.

Essentially the algorithm to reduce a subtree overlap model consists of one step: examine each vertex v of the model in turn to see if v is a spare vertex in the current

model and, if so, remove this v from the model. Removing a spare vertex from the model will never create new spare vertices - any vertex that wasn't spare before will not become spare - and the removal may result in several other vertices ceasing to be spare vertices, as happens in the above example when vertices $7, 4$, and 2 are each removed in turn from the model. This algorithm can be coded as follows:

1. For vertex $v \in V^T$

   if v is a spare vertex then

   (a) Set $N = \{x; x \in V^T, (v, x) \in E^T\}$, so $N = \{n_1, n_2, \ldots, n_k\}$ is the set of all neighbours of v, where k is the number of neighbours of v.

   (b) Set $E^T = E^T - \{(v, n_i); 1 \leq i \leq k\}$

   (c) Set $E^T = E^T + \{(n_1, n_i); 1 < i \leq k\}$

   (d) For all T_i such that $v \in T_i$, $T_i = T_i - \{v\}$

   (e) Set $V^T = V^T - \{v\}$

Lemma 5.3 (Correctness of Algorithm) *Given a subtree overlap graph $G = (V^G, E^G)$ represented by a subtree overlap model consisting of a tree $T = (V^T, E^T)$ and a collection of subtrees $\{T_i\}_{i=1}^{n^G}$, the model resulting after the algorithm is executed is minimal and still correctly represents the graph G.*

Proof: The resulting subtree overlap model will be minimal since the spare vertices are removed in step 1. The spare vertices are removed iteratively, which is to say that each vertex v_i in the model is examined in turn and if - at that point in the algorithm - v_i is still a spare vertex, it is then removed. Moreover, for all $v_j, j < i$, if v_j is still a vertex of the model, v_j is guaranteed not to be a spare vertex, since otherwise v_j would have been removed from the model. Therefore, when the algorithm has examined all the vertices in the model, no spare vertices will remain.

It remains only to show that the subtree overlap model still accurately represents G after each iteration of this step is completed and a spare vertex v is removed. For each iteration where v is not spare, nothing happens, and the model remains accurate. Therefore we can assume that v is a spare vertex, and therefore the removal of v will

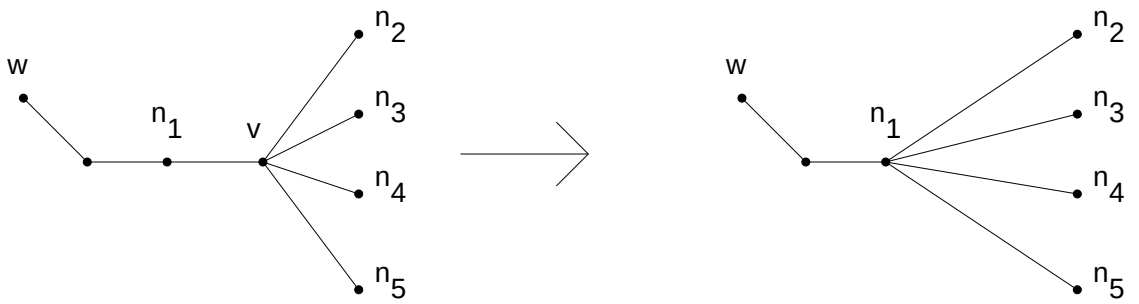


Figure 5.3: Removing a spare vertex from the model.

not affect any edge relationships between subtrees. It remains only to show that after this step the tree and all affected subtrees remain connected.

Step 1 is illustrated in Figure 5.3. The first step in the algorithm is to contract the edge between this v and one of its neighbours. The tree remains connected after this edge contraction, but every subtree containing v is now unconnected, since v is no longer adjacent to any vertex in the graph. However, simply removing v from every subtree which contains v - which is the next step - will produce subtrees that are once more connected.

It remains only to show that the subtrees still accurately model the graph G. Therefore consider any two subtrees T_i and T_j. It is easy to see that if neither subtree contains v then the edge relationships will remain undisturbed. Therefore we can assume without loss of generality that $v \in T_i$. Then there are four possible cases:

1. $\phi_O(T_i, T_j) = 1$ and $v \in T_j$
2. $\phi_O(T_i, T_j) = 1$ and $v \notin T_j$
3. $\phi_O(T_i, T_j) = 0$ and $v \in T_j$
4. $\phi_O(T_i, T_j) = 0$ and $v \notin T_j$

In the first two cases, the definition of a spare vertex will guarantee that there exists some other vertex $w \in V^T$ which will satisfy the edge relationship. In the third case, where both subtrees include v, either T_i is contained in T_j or vice versa. Regardless, v is removed from both subtrees, and the containment of one subtree within the other remains. In the last case, where T_j does not contain v, either T_i and

T_j do not intersect and the edge relationship after v is removed remains unchanged, or T_j is contained in T_i and will remain that way when v is removed.

Therefore the subtree overlap model will still represent G after every iteration of each of the two steps, and the final subtree overlap model will correctly represent G. □

Lemma 5.4 (Complexity of Algorithm) *The algorithm runs in* $O((n^T)^2(n^G)^2))$ *time.*

Proof:

Step 1 will iterate n^T times, once for each vertex in the tree T. Each vertex is examined to see if it is - at this point in the algorithm - still a spare vertex. This examination requires the checking of each pair of subtrees; a pair of subtrees can be checked in at most n^T steps, and $(n^G)^2$ pairs need to be checked. Therefore the complexity of this step and this algorithm is $O((n^T)^2(n^G)^2)$.□

In short, it is possible, given a subtree overlap model, to reduce this model to a minimal form in polynomial time. This is important because it is possible to prove an upper bound on the size of a minimal subtree overlap model, as the following theorem shows.

Theorem 5.5 *Given a subtree overlap graph* $G = (V^G, E^G)$ *with a minimal subtree overlap model consisting of a tree* $T = (V^T, E^T)$ *and a set of subtrees* $\{T_i\}$, $|V^T| \leq 3m$, *where* $m = |E^G|$.

Proof: Consider a subtree overlap graph G, with m edges, and a minimal subtree overlap model consisting of a tree $T = (V^T, E^T)$ and a set of subtrees $\{T_i\}$ which accurately represents G.

For each edge $e_i = (v_x, v_y) \in E^G$ find one triple $p_i = (a_i, b_i, c_i)$ so that $a, b \in T_x, c \notin T_x$ and $a \notin T_y, b, c \in T_y$. At least one such triple must exist for every edge, for otherwise $\phi_O(T_x, T_y) = 0$, contradicting the assumption that this model represents G. Let P be the set of all vertices contained in all p_i. It follows that $|P| \leq 3m$, since exactly m triples were constructed, and each triple contains 3 vertices. $|P| < 3m$ if duplication occurs between two or more triples.

It remains only to show that $P = V^T$.

Assume that $P \neq V^T$. $P \subseteq V^T$, and therefore there must exist at least one $v \in V^T$ such that $v \notin P$. Then v is not a part of any triple. If v is not contained in any subtree, then v is a spare vertex, contradicting the assumption that the subtree overlap model is minimal. Therefore v must be contained in at least one subtree. But each edge relationship between a subtree T_i containing v and any other subtree can be represented using some other vertex $w, w \neq v$ in the tree, since v not in P. Therefore v is a spare vertex, again contradicting the assumption that the subtree overlap model is minimal.

Therefore $P = V^T$. $\square$

It is possible, then, to reduce any subtree overlap model to a minimal form; and, moreover, there exists an upper bound on the size of the minimal model. The algorithm will not yield a minimum subtree overlap model in all cases, where a minimum subtree overlap model is a model of minimum size. However, the key point is that a bound can be put on the size of the model in terms of the number of edges of the original graph. This means that the size of any subtree in the model is similarly bounded, for no subtree can exceed the tree in size. This is particularly important in Section 6.2, when proving that the Maximum Independent Set problem is polynomially solvable for subtree overlap graphs.

Chapter 6

The Maximum Independent Set Problem

Gavril, in [Gav73], presents an algorithm to find the maximum independent set in a circle graph. This section generalizes the algorithm Gavril presents to provide a solution for any pair of overlap/intersection graph classes, as long as certain conditions are met, and then proves that the pair of subtree overlap and subtree intersection graph classes meets those conditions, yielding a polynomial algorithm.

Given a graph $G = (V, E)$, an *independent set* consists of a subset $H \subset V$ such that for all $v, w \in H$ $(v, w) \notin E$). Then an induced subgraph of G induced on the vertices in H would contain absolutely no edges. A *maximum independent set* is an independent set of G whose cardinality is not exceeded by that of any other independent set of G.

A pair of overlap/intersection graph classes refers to the overlap graph class and intersection graph class of objects of type Z. The two graph classes are related in that the definition of Z remains constant for both graph classes. An example of one such pair are the subtree overlap and subtree intersection graph classes.

6.1 Algorithm for any Pair of Overlap/Intersection Graphs

Note that this algorithm is very similar to the one that Gavril presents, but is a generalization to all pairs of intersection/overlap graphs of a shared class of objects Z.

Firstly, to introduce the notation. Let $\alpha(G)$ denote the size of a maximum independent set for a graph G. Moreover, if G is a weighted graph, $\alpha_w(G)$ will denote the sum of all the weights of all the vertices of a maximum weight independent set.

Now, if G is an overlap graph of objects Z, then every vertex in G is mapped to some object of type Z, and C is the collection of objects that represent the vertices of G. If G is a subtree overlap graph, then C is the set of subtrees $\{T_i\}$. Then let $\tilde{G}$ represent the intersection graph of this collection C of objects of type Z. Then if G is a subtree overlap graph, $\tilde{G}$ is the subtree intersection graph whose tree T and collection of subtrees $\{T_i\}$ is identical to that of G.

Note that every object $e \in C$ denotes a vertex v in both G and $\tilde{G}$, and that two vertices are adjacent in G if their objects overlap (which is to say, they intersect but neither contains the other) but that those same two vertices are adjacent in $\tilde{G}$ if their objects intersect. Therefore $\tilde{G}$ will always contain all the edges of G, but may also contain extra edges.

Now given an overlap graph $G = (V, E), |V| = n$ with a collection of objects $C = \{c_i\}_{i=1}^{n}$, where c_i represents $v_i \in V$, let $U_i = \{c_j; c_j \subset c_i\}$ be the set of all objects contained in the object c_i. Let G_i be the overlap subgraph of G induced by U_i.

Then assume that a weight $WT(i) = \alpha(G_i) + 1$ is assigned to every vertex $v \in \tilde{G}$. Now assuming $\{v_1, v_2, \ldots, v_r\}$ is a maximum weight independent set of $\tilde{G}$ and $D_j, 1 \leq j \leq r$ is a maximum independent set of G_j, consider the following lemma.

Note that Lemma 6.1 shows that finding the maximum independent set of an overlap graph G is equivalent to finding the maximum weighted independent set of the graph $\tilde{G}$, if each vertex $v \in V$ is assigned a weight $WT(v)$ and a set D_v so that $WT(v) = \alpha(G_v) + 1$ and D_v is a maximum independent set of G_v.

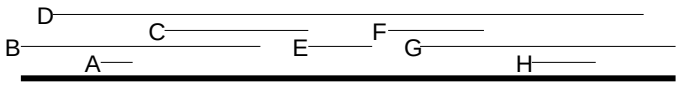


Figure 6.1: An illustration of Lemma 6.1

Consider the example in Figure 6.1. In this figure, for instance, $U_D = \{A, C, E, F, H\}$. Then, for the overlap graph represented by this model, every independent set containing the vertex D, could also contain vertices from U_D forming an independent set. For instance, any independent set containing D could also contain A, C, and F. Then

D_D is an independent set of U_D whose weight is maximized, and in this example D_D could be $\{A, C, F\}$.

Lemma 6.1 *$\alpha(G) = \alpha_w(\tilde{G})$ and $\bigcup_{j=1}^{r}(D_j \cup \{v_j\})$ is a maximum independent set of G.*

Proof:

Let J be a maximum independent set of G.

Let $A = \{x; x \in J, c_x \subset c_y \rightarrow y \notin J\}$. Assume $A = \{u_1, u_2, \ldots, u_t\}$.

Then for every $1 \leq i \leq t$, the set $J_{u_i} = \{v_y; v_y \in J \text{ and } c_y \subset c_{u_i}\}$ is an independent set of G_{u_i}. Let C be a maximum independent set of G_{u_i}, and assume $|J_{u_i}| < |C|$.

Then certainly $(J - J_{u_i}) \cup C$ is an independent set of G, and one which has more vertices than J, which is a contradiction. Therefore $|J_{u_i}| = |C|$ and for every $1 \leq i \leq t$ J_{u_i} is a maximum independent subset of G_{u_i}.

Note that for every $i \neq j$, where $v_i, v_j \in J$, one of three possibilities hold: $c_i \subset c_j$, or $c_j \subset c_i$, or $c_i \cap c_j = \emptyset$. However, since A contains only those v_i whose objects are not contained in the objects representing any v_j, it follows that for every pair of vertices $v_i, v_j \in A$, $c_i \cap c_j = \emptyset$. Therefore A is an independent set of $\tilde{G}$.

Moreover $WT(u_i)$ represents the size of the independent set consisting of the vertex u_i and the maximum independent set of the subgraph G_{u_i} of G. This subgraph G_{u_i} consists only of those vertices v_j such that $c_j \subset c_{u_i}$, which means that each of these v_j are not adjacent to u_i in G.

Therefore

$$\alpha(G) = \sum_{i=1}^{t} WT(u_i) \leq \alpha_w(\tilde{G})$$

where each $WT(u_i)$ represents the size of the maximum independent set of the subgraph G_{u_i} which consists of the vertex u_i and all vertices u_j such that $c_{u_j} \subset c_{u_i}$.

Now let $B = \{x_1, x_2, \ldots, x_r\}$ be a maximum weight independent set of $\tilde{G}$. Then each D_j is a maximum independent set of G_{x_j} and $D_j \cup \{x_j\}$ is an independent set of G. Since $c_{x_i} \cap c_{x_j} = \emptyset$ for all $1 \leq i < j \leq r$, $\bigcup_{j=1}^{r}(D_j \cup \{x_j\})$ is an independent set of G. Therefore

$$\alpha_w(\tilde{G}) = \sum_{i=1}^{r} WT(x_i) = |\cup_{j=1}^{r}(D_j \cup \{x_j\})| \leq \alpha(G)$$

Therefore $\alpha(G) = \alpha_w(\tilde{G})$ and $\cup_{j=1}^{r}(D_j \cup \{x_j\})$ is a maximum independent set of G. $\square$

Consider the following algorithm to find the maximum independent set of a graph G, which has been modified from Gavril's algorithm as presented in [Gav73]:

1. Build a containment/intersection $n \times n$ matrix M such that

$$M_{i,j} = \begin{cases} 0 & c_i \text{ and } c_j \text{ do not intersect} \\ 1 & c_i \text{ and } c_j \text{ intersect but } c_j \text{ does not contain } c_i \\ 2 & c_i \text{ is contained in } c_j \end{cases}$$

2. Define a family of subsets of C (the collection of objects) in the following way:

   $A_1 = \{c_i;$ there does not exist $c_j \in C$ such that $c_j \subset c_i\}$

   $A_2 = \{c_i; c_i \notin A_1$ and if $c_j \subset c_i$ then $c_j \in A_1\}$

   $\vdots$

   $A_k = \{c_i; c_i \notin \bigcup_{j=1}^{k-1} A_j$ and if $c_j \subset c_i$ then $c_j \in \bigcup_{j=1}^{k-1} A_j\}$

   $\bigcup_{i=1}^{k} A_i = C$

3. For each vertex v_i such that $c_i \in A_1$, set $WT(v_i) = 1$ and $D_{v_i} = \emptyset$.

4. For $x = 2 \ldots k$ do:

   (a) For each vertex v_i such that $c_i \in A_x$ do the following:

      i. Build G_{v_i} which is the overlap subgraph of all vertices contained in c_i

      ii. Assign to each corresponding vertex $u \in \tilde{G}_{v_i}$ the weight $WT(u)$.

      iii. Find a maximum weight independent set $\{w_1, w_2, \ldots, w_t\}$ in $\tilde{G}_{v_i}$

      iv. Set $WT(v_i) = \alpha(G_{v_i}) + 1 = \alpha_w(\tilde{G}_{v_i}) + 1$

      v. Set $D_{v_i} = \bigcup_{j=1}^{t}(D_{w_j} \cup \{w_j\})$

5. For every vertex v in G, attach the weight $WT(v)$ to the corresponding vertex in $\tilde{G}$.

6. Find a maximum weight independent set $\{w_1, w_2, \ldots, w_r\}$ of $\tilde{G}$.

7. Compute the set $B = \bigcup_{j=1}^{r} (D_{w_j} \cup \{w_j\})$ which is a maximum independent set of G.

The proof of this algorithm is inductive, and follows after the following observations. Firstly $c_i \in A_x$ if and only if c_i contains at least one object $c_j \in A_{x-1}$ and for every object c_j contained in c_i, $c_j \in \bigcup_{j=1}^{x-1} A_j$.

Secondly, if $c_i, c_j \in A_x$ then c_i and c_j overlap or are disjoint, but neither is contained in the other. Moreover, if $x \neq y$, then $A_x \cap A_y = \emptyset$, and every c_i is a member of exactly one A_x.

Thirdly, once the matrix M has been constructed, the intersection and containment status of two objects c_i and c_j can be determined in constant time by looking up $M_{i,j}$ and $M_{j,i}$. Clearly, if $M_{i,j} = M_{j,i} = 0$, the two objects do not intersect. Similarly, if $M_{i,j} = M_{j,i} = 1$, the two objects do overlap, but if $M_{i,j} = 1$ and $M_{j,i} = 2$ then c_j is contained in c_i. It can be assumed without loss of generality that no two objects will be identical, so that the case of $M_{i,j} = M_{j,i} = 2$ will never occur. If the matrix is constructed correctly the case where $M_{i,j} = 0$ and $M_{j,i} = 1$ should never occur, since that would indicate that c_i intersects c_j, but c_j has no contact with c_i, which would be a contradiction. Naturally, it must be possible to construct this matrix in polynomial time for a given graph G modelled using overlapping objects of type Z.

Fourthly, the maximum weight independent set problem must be polynomially solvable for any graph $\tilde{G}$ modelled using intersecting objects of type Z to complete Steps 4 and 6.

Theorem 6.2 *This algorithm will produce a maximum independent set for an overlap graph G of objects of type Z in $O(max(f(n,m), n(g(n,m)))$ time if the containment/intersection matrix M can be built in time $O(f(n,m))$, and the maximum weight independent set problem is solved for intersection graphs of objects of type Z in time $O(g(n,m))$.*

Proof: Clearly, in the base case, for every $v \in A_1$ $WT(v) = 1$ and $D_v = \emptyset$.

Now assume by induction on x that all the vertices v_i where $c_i \in \bigcup_{j=1}^{x-1} A_j$ are attached a weight $WT(v_i) = \alpha(G_{v_i}) + 1$ and a maximum independent set D_{v_i} of G_{v_i}.

Consider a vertex u with a corresponding object c such that $c \in A_x$. For every vertex v_i of G_u, $c_i \in \bigcup_{j=1}^{x-1} A_j$, and thus every vertex v_i in G_u has already been assigned both a $WT(v_i)$ and a D_{v_i}. Assigning the same values for each $v_i \in G_u$ to the corresponding vertices of $\tilde{G}_u$ results in the weighted object graph $\tilde{G}_u$, whose maximum weighted independent set $\{w_1, w_2, \ldots, w_t\}$ can be found in polynomial time $O(g(n,m))$.

Applying Lemma 6.1, $\alpha(G_u) = \alpha_w(\tilde{G}_u)$ and $\bigcup_{j=1}^{t}(D_{w_j} \cup \{w_j\})$ is a maximum independent set of G_u. So defining $WT(u) = \alpha(G_u) + 1$ and $D_u = \bigcup_{j=1}^{t}(D_{w_j} \cup \{w_j\})$ effectively attaches the weight $WT(u)$ and a maximum independent set D_u of G_u to every vertex u of G. Applying Lemma 6.1 again, it follows that B is a maximum independent set of G. Therefore this algorithm will produce a maximum independent set for G.

The complexity of this algorithm is then $O(max(f(n,m), n(g(n,m))))$, where $n = |V|$. Step 1 requires $O(f(n,m))$ complexity. Once the matrix M has been constructed, Step 2 can be completed in $O(n^2)$ time. Steps 3 and 4 will require $O(g(n,m))$ time to compute the weight and maximum independent set for each vertex and will visit each of the n vertices once, resulting in a complexity of $O(n(g(n,m)))$. Step 5 can be completed in $O(n)$ steps, and step 6 will then require $O(g(n,m))$ time. Lastly, step 7 can be computed in $O(n)$ time. Since each of the steps follows linearly, the time complexity is the larger of $O(f(n,m))$ and $O(n(g(n,m))$.□

6.2 The Subtree Overlap Graph

Consider a pair of subtree overlap and intersection graphs G and $\tilde{G}$, where G is a subtree overlap graph with a minimal subtree overlap model. As proven in Theorem 6.2, if the matrix M in the algorithm can be built in polynomial time and the maximum weight independent set problem is solved for the class of subtree intersection graphs, then the maximum independent set can be found in polynomial time for any subtree overlap graph.

Building the matrix will require $O(mn^2) < O(n^4)$ complexity, where $m = |E|$. To build the matrix n^2 pairs of subtrees need to be examined. By Theorem 5.5, each subtree will contain at most $3m$ vertices if G is modelled by a minimal subtree

overlap model, and so comparing any two subtrees will require $O(m)$ complexity, since the information stored for each subtree T_i consists of the vertices contained in T_i, and testing for intersection between two subtrees T_i and T_j requires only the finding of a triple (a, b, c) where a, b, and c are vertices in the underlying tree so that $a, b \in T_i, a \notin T_j$ and $b, c \in T_j, c \notin T_i$. If there is no b, there is no intersection. If there is no a, but there is a b, then T_i is contained in T_j. Similarly, if there is no c but there is a b, then T_j is contained in T_i. If all three vertices can be found, then T_i and T_j intersect without containment.

To address the second requirement, Gavril presents, in [Gav72] a polynomial algorithm to find the maximum weighted independent set of a chordal graph in $O(n^2)$ time, where $n = |V|$. Since the chordal and subtree intersection graphs are equivalent, this means that the maximum weighted independent set of a subtree intersection graph can be computed in $O(n^2)$ time.

Thus, by Theorem 6.2, the complexity of the maximum independent set for G is $O(max(mn^2, n^3)) = O(mn^2)$. This leads to the following theorem:

Theorem 6.3 *The Maximum Independent Set Problem for subtree overlap graphs is solvable in $O(mn^2)$ time if a minimal subtree overlap model for the graph is given.*

6.3 Overlap Graphs and NP-completeness

A polynomial solution for the maximum independent set problem may not exist for every type of overlap graph. In fact if the Independent Set problem is NP-complete for the related proper intersection graph, then the problem is NP-complete for the overlap graph, since the proper intersection graph class is always contained in the overlap graph class. Asano, in [Asa88], presents the following theorem.

Theorem 6.4 *[Asa88] The Maximum Independent Set problem is NP-complete for intersection graphs of objects of type M if M is:*

1. *a class of unit squares*
2. *a class of unit circles*
3. *a class of line segments in the plane*

4. *a class if horizontal and vertical line segments in the plane, any two of which may overlap*

5. *a class of horizontal, vertical, and 45 degree line segments in the plane with no two of the same type touching.*

The theorem describes several different classes of intersection graphs. Classes 3 and 4 are not proper classes of intersection graphs, since the containment of one object inside another is not precluded. Class 5, in contrast, is a proper intersection graph since no two line segments of the same type are allowed to touch, let alone contain one inside the other. Therefore the maximum independent set problem is NP-complete for the class of overlap graphs of objects of type 5.

The two classes of unit squares and unit circles are not immediately proper, since one square could be overlay another exactly, if the two squares are identical. However, for purposes of solving the maximum independent set problem this guarantees that at most one of these identical squares, or circles, will be chosen, and that moreover the choice of which identical square, or circle, is chosen is irrelevant. Therefore, the intersection graph can be preprocessed to remove all but one copy of identical squares, or circles, and the remaining graph will be a proper intersection graph. Then, since the Maximum Independent Set problem is NP complete for these two classes of intersection graphs, the problem will also be NP-complete for the related classes of overlap graphs.

Similarly, Rim and Nakajima show, in [RN95] that the Maximum Independent Set problem is NP-complete for intersection graphs of rectangles in the plane whose sides are parallel to the axes. From this it follows then that the Maximum Independent Set problem is also NP-complete for the overlap graph of the same rectangles, and they show that that does indeed follow.

On a slightly different note, Kashiwabara, Masuda, Nakajima, and Fujisawa [KMNF91] apply the algorithm of Theorem 6.2 to the specific case of the class of graphs modelled by overlapping circular arcs, showing that the complexity of the maximum independent set problem for this class of graphs is $O(n^2)$.

Chapter 7

Algorithmic Complexities

The maximum independent problem is polynomially solvable for the class subtree overlap graphs. But what of other well known graph problems? This chapter is concerned with the question of the complexity of various well known graphs problems, including the Hamiltonian Cycle, vertex colouring, several variants of domination problems, and the question of graph isomorphism.

7.1 The Hamiltonian Cycle Problem

Given a graph $G = (V, E)$, G contains a Hamiltonian path if it is possible to follow a path through the graph that passes through every vertex exactly once. G contains a Hamiltonian Cycle if the graph contains a Hamiltonian path whose start and endpoints are adjacent.

Damaschke, in [Dam89], shows that both problems are NP-complete for circle graphs. Since, by Theorem 4.1, the class of circle graphs is a proper subset of the class of subtree overlap graphs, it follows that the Hamiltonian Cycle and Path Problems are NP-complete for subtree overlap graphs too.□

7.2 Domination Problems

For a graph $G = (V, E)$, a subset $S \subseteq V$ of the vertices of G is a dominating set if every vertex $v \in V$ that is not in S is adjacent to a vertex in S. In general, domination problems are concerned with finding a minimum cardinality S. Moreover, domination problems can be varied by placing restrictions on the structure of the subgraph G_S

induced on G by the set S.

Possible restrictions include the requirement that S is connected so that there is a path from every vertex in S to every other vertex in S, or that S is a clique which means that every pair of vertices in S is adjacent. Similarly S can be a total set, which means G_S has no isolated vertices, or S can be an independent set.

7.2.1 Domination and the Circle Graph

Keil[Kei93] proves that the problems of finding a minimum dominating set, minimum total dominating set, and minimum connected dominating set are all NP-complete for circle graphs. Since Theorem 4.1 shows that the class of circle graphs is a proper subset of the class of subtree overlap graphs, it follows that these problems are NP-complete for subtree overlap graphs too.

However, Keil presents an $O(nm)$ algorithm to solve the problem of finding a minimum dominating clique for the circle graph, and notes that the problem of finding the minimum independent dominating set for a circle graph is still open.

7.2.2 Domination and the Chordal Graph

Theorem 4.5 shows that the class of subtree intersection graphs, which is equivalent to the class of chordal graphs, is a proper subset of the class of subtree overlap graphs.

Booth and Johnson[BJ82] show that the problem of finding the minimum dominating set is NP-complete for chordal graphs, which again means that the problem is NP-complete for subtree overlap graphs, too. Keil also mentions in [Kei93] that the minimum cardinality dominating clique problem is NP-complete for chordal graphs, and hence NP-complete for subtree overlap graphs.

Yen and Lee[YL96] consider the problem of the perfect dominating set. A perfect dominating set D of a graph $G = (V, E)$ is a subset of V such that every vertex not in D is adjacent to exactly one vertex in D. As before, the structure of the graph G_D induced in G by the set D can be further restricted so that G_D is a connected, total, or independent graph.

They prove that the general problem of finding a minimum perfect dominating set, as well as finding minimum perfect total and independent dominating sets, are all NP-complete for chordal graphs.

7.3 The Graph Isomorphism Problem

It is possible, as shown in Lemma 5.5 to put a constraint on the size of the underlying tree for a subtree overlap graph. However, the problem of determining whether two graphs are isomorphic is isomorphism-complete for subtree overlap graphs.

Booth and Leuker show that the problem of graph isomorphism is isomorphism-complete for chordal graphs [BL76]. Since by Theorem 4.5 chordal graphs are a proper subset of subtree overlap graphs, the problem of isomorphism is isomorphism-complete for subtree overlap graphs.

7.4 Colouring a Subtree Overlap Graph

Colouring a graph $G = (V, E)$ means that each vertex in the graph is assigned some colour so that no two adjacent vertices are assigned the same colour. In general the problem of colouring the graph with a minimum number of colours is NP-complete, although for some simple graph classes, including the permutation graph class as mentioned in the introduction, polynomial algorithms exist.

Unger, in [Ung88], shows that the problem of colouring a circle graph with k colours is NP-complete for $k \geq 4$, although he provides an $O(nlogn)$ algorithm for 3-colouring circle graphs [Ung92]. Therefore, since by Theorem 4.1 the class of subtree overlap graphs contains the family of circle graphs, the k-colouring problem is NP-complete for $k \geq 4$, and is at least $O(nlogn)$ for $k = 3$.

Chapter 8

Conclusion

In studying the subtree overlap graph, several results are obtained. Some of these results are specific to subtree overlap graphs, whilst the algorithmic solution of the maximum independent set problem provides a more general solution for all overlap graphs. The results obtained, then, can be divided into the following two sections.

8.1 Subtree Overlap Graphs

In studying the subtree overlap graph, two main results are proven. Firstly, the place of the subtree overlap graph class in the hierarchy of graph classes is further refined by proving that the class of spider graphs is a proper subclass of the class of subtree overlap graphs. This leads to a better understanding of the scope of this class of subtree overlap graphs.

Secondly, an upper bound on the size of a subtree overlap model is proven to be of order $O(m)$ if the model is minimal. An algorithm is provided to reduce any subtree overlap model to a minimal - but not necessarily minimum - subtree overlap model. This bound is then used to show that an intersection/containment matrix M can be built in $O(mn^2)$ time. M can be used to reduce the testing for intersection, containment, and overlap thenceforth to $O(1)$ rather than $O(m)$ time.

Thirdly, a survey was completed of well known graph problems as to their computational complexity. Since the place of the subtree overlap graph class in the hierarchy of classes is known, most problems which are NP-complete for a subclass of the subtree overlap graph class are also NP-complete for the class of subtree overlap graphs itself.

One exception to this 'inheritance of difficulty' rule is the problem of recognition; there exist classes of graphs who are recognizable in polynomial time but who contain a subclass of graphs that is not recognizable in polynomial time. One such pair of graphs are the classes of circle and k-polygon graphs[ES].

8.2 The Maximum Independent Set Problem

A far more general result is proven for the maximum independent set problem. An algorithm is presented which will find the maximum independent set of any overlap model if the following two conditions hold:

1. An intersection/containment matrix can be built in polynomial time
2. The maximum weight independent set problem is polynomially solvable for the intersection graph obtained from the same model.

A brief survey is made of overlap graph classes for which the maximum independent set problem is either polynomially solvable using this algorithm, or NP-complete.

Chapter 9

Future Research

Several avenues of exploration remain.

Work also needs to be done on the recognition algorithm for the subtree overlap graph; it is good to know that there is an upper bound on the subtree overlap model, and that - given the model - we can find a maximum independent set for a subtree overlap graph. However, these result will become more useful once it is possible to recognize a subtree overlap graph and construct its model.

The maximum independent set algorithm can be also be expanded. As long as the weight of any vertex in the graph is greater than the sum of the weights of the maximum independent set of vertices contained within the object representing this vertex, the algorithm is guaranteed to produce a maximal independent set. By manipulating the weights, it may be possible to generate all maximal independent sets, or even the minimum independent dominating set.

Similarly, the minimum independent dominating set problem remains open for circle graphs, and hence for spider graphs and subtree overlap graphs, as does finding an approximate minimum colouring. An upper bound has been established on the number of colours needed to colour a spider graph, albeit sub optimally but in polynomial time, but no such bound has been established for the subtree overlap graph as yet.

A definition of the new class of sphere graphs is given, and it is shown that the class of circle graphs is a proper subclass of the class of sphere graphs. Exploring this class further proved to lie beyond the scope of the thesis, but does offer interesting avenues of exploration in the future.

Bibliography

[Asa88] T. Asano. Difficulty of the maximum independent set problem on intersection graphs of geometric objects. In Y. Alavi, G. Chartrand, O.R. Oellerman, and A.J. Schwenk, editors, *Proceedings of the Sixth Quadrennial International Conference on the Theory and Application of Graphs*, volume 1. Western Michigan University, 1988.

[BJ82] Kellogg S. Booth and J. Howard Johnson. Dominating sets in chordal graphs. *SIAM J. Comput.*, 11(1):191–199, 1982.

[BL76] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and planarity graphs using pq-tree algorithms. *J. Comput. Syst. Sci*, 13:335–379, 1976.

[Dam89] Peter Damaschke. The Hamiltonian circuit problem for circle graphs is NP-complete. *Inform. Process. Lett.*, 32(1):1–2, 1989.

[DM41] B. Dushnik and E. W. Miller. Partially ordered sets. *American Journal of Mathematics*, 63:600–610, 1941.

[ES] E. S. Elmallah and Lorna Stewart. Polygon graph recognition. Private Communication.

[FMW97] Stefan Felsner, Rudolf Müller, and Lorenz Wernisch. Trapezoid graphs and generalizations, geometry and algorithms. *Discrete Appl. Math.*, 74(1):13–32, 1997.

[Gav72] F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *Siam J. Comput.*, 1:180–187, 1972.

[Gav73] F. Gavril. Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3:261–263, 1973.

[Gav74] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory Ser. B*, 16:47–56, 1974.

[Gol80] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press Inc, 1980.

[GS89] Martin Charles Golumbic and Edward R. Scheinerman. Containment graphs, posets, and related classes of graphs. In *Combinatorial Mathematics: Proceedings of the Third International Conference (New York, 1985)*, volume 555 of *Ann. New York Acad. Sci.*, pages 192–204, New York, 1989. New York Acad. Sci.

[GSH89] Csaba P. Gabor, Kenneth J. Supowit, and Wen-Lian Hsu. Recognizing circle graphs in polynomial time. *J. Assoc. Comput. Mach.*, 36(3):435–473, 1989.

[Kei93] J. Mark Keil. The complexity of domination problems in circle graphs. *Discrete Appl. Math.*, 42:51–63, 1993.

[KK97] Alexandr Kostochka and Jan Kratochvil. Covering and coloring polygon-circle graphs. *Discrete Math.*, 163:299–305, 1997.

[KMNF91] Toshinobu Kashiwabara, Sumio Masuda, Kazuo Nakajima, and Toshio Fujisawa. Polynomial time algorithms on circular-arc overlap graphs. *Networks*, 21(2):195–203, 1991.

[Koe90] M. Koebe. Colouring of spider graphs. In *Topics in Combinatorics and Graph Theory*, pages 435–442. Physica-Verlag Heidelberg, 1990.

[Koe92] M. Koebe. On a new class of intersection graphs. In *Combinatorics Graphs Complexity: Proceedings of the 4th Czechoslovak Symposium on Combinatorics*, pages 141–143. North Holland, 1992.

[MS94] Ross M. McConnell and Jeremy P. Spinrad. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (Arlington, VA, 1994)*, pages 536–545, New York, 1994. ACM.

[Nov94] Diego A. Novillo. Overlap graphs of subtrees in a tree. Project for Cmput 506, 5 April 1994.

[Ola91] Stephan Olariu. An optimal greedy heuristic to color interval graphs. *Inform. Process. Lett.*, 37:21–25, 1991.

[RN95] Chong S. Rim and Kazuo Nakajima. On rectangle intersection and overlap graphs. *IEEE Trans. Circuits Systems I Fund. Theory Appl.*, 42(9):549–553, 1995.

[RTL76] Donald J. Rose, R. Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5(2):266–283, 1976.

[Spi94] Jeremy Spinrad. Recognition of circle graphs. *Journal of Algorithms*, 16:264–282, 1994.

[Ung88] Walter Unger. On the k-colouring of circle graphs. In R. Cori and M. Wirsing, editors, *STACS 88: Fifth annual symposium on the theoretical aspects of computer science, Bordeaux, France, February 11-13, 1988 Proceedings*, volume 294 of *Lecture Notes in Computer Science*, pages 61–72. Springer-Verlag, Berlin, 1988.

[Ung92] Walter Unger. The complexity of colouring circle graphs. In A. Finkel and M Jantzen, editors, *STACS'92. Proceedings.*, volume 577 of *Lecture Notes in Computer Science*, pages 389–400. Springer, 1992.

[YL96] Chain-Chin Yen and R. C. T. Lee. The weighted perfect domination problem and its variants. *Discrete Appl. Math.*, 66(2):147–160, 1996.

The independence number of a graph G is the maximum cardinality of an independent set in G. The girth of a graph is the length of its smallest cycle, while the chordality of a graph is the length of its largest chordless cycle. A sub-division of an edge uv consists in replacing the edge with a new vertex adjacent to u and v. For two graphs G and H, we denote by G + H the disjoint union of G and H. In particular, nG is the disjoint union of n copies of G. As usual, Pn, Cn and Kn denote the chordless path, the chordless.Â Theorem 1. Let M be a set of graphs and X the class of M -free planar graphs of degree at most 3. If Îº(M) < âˆž, then the maximum independent set prob-lem is NP-hard in the class X. Proof. [2] also considered the Maximum Clique problem in D-DI graphs, and showed that this problem is fixed parameter tractable with respect to D. Jiang [17] studied the problem of Maximum Independent Set in D-DI graphs. He presented a simple 32 -approximation algorithm for 2-DI graphs, and a (5D 6 + O(log d))-approximation algorithm or D-DI graphs.Â In this paper we focus mainly on three classical graph-theoretic optimization problems: Maximum Independent Set, Minimum Dominating Set, and Minimum Vertex Cover. We present an O(DnD)-time algorithm for Maximum Independent Set and Minimum Vertex Cover in D-DI graphs with n 2 vertices, and give an O(D2 nO(D))-time algorithm for Minimum Dominating Set. While taking the Algorithms: Design and Analysis II class, one of the questions asks about the maximum-weight independent set problem for a path graph. shown below is a (blurry) screenshot of the problem statement, and the corresponding lecture videos are on YouTube: https://www.youtube.com/watch?v=0awkct8SkxA. https://www.youtube.com/watch?v=pLOkbHGRsv0. https://www.youtube.com/watch?v=Im_zjFkZDCY. This problem can be elegantly solved by dynamic programming, with literally one line of code. a[i] = max(a[i - 1], a[i - 2] + w[i]). The question is as follows: Which of the following is true for o