THE MAXIMUM K-COLORABLE SUBGRAPH PROBLEM

by

Giri Narasimhan

Computer Sciences Technical Report #864

July 1989

The Maximum K-Colorable Subgraph Problem

by

Giri Narasimhan

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

Doctor of Philosophy
(Computer Sciences)

at the
UNIVERSITY OF WISCONSIN-MADISON
1989

Abstract

THE MAXIMUM K-COLORABLE SUBGRAPH PROBLEM

Giri Narasimhan

Under the supervision of Assistant Professor Rachel Manber

For a given graph, the *Maximum k-Colorable Subgraph Problem* is the problem of determining the largest set of vertices of the graph that can be colored using k distinct colors such that adjacent vertices, if colored, are assigned different colors. This problem is NP-hard for general graphs. We examine several approaches to deal with the hardness of the maximum k-colorable subgraph problem. One of our main results is that *algebraic* techniques can be used to efficiently compute bounds for the size of the maximum k-colorable subgraph in a graph.

As a first approach, we designed polynomial-time algorithms for the maximum 2-colorable subgraph problem on various special classes of graphs like interval graphs, circular-arc graphs, and tolerance graphs.

As a second approach, we considered approximation algorithms for the maximum k-colorable subgraph problem. We showed that no good approximation algorithm is possible for general graphs, and we analysed the naive greedy approximation algorithm on some special classes of graphs.

As a last approach, we used algebraic techniques to design algorithms that find upper bounds for the size of the maximum k-colorable subgraph. This approach was motivated by work due to László Lovász. Lovász proved a *sandwich theorem* by defining an efficiently computable function of the graph whose value, for any graph, always lies between two interesting, but hard to compute, graph parameters. We proved a generalization of Lovász's "sandwich" theorem by defining another efficiently computable graph function whose value always lies between two other interesting graph parameters that are hard to compute. One of the bounding parameters is the size of the maximum k-colorable subgraph in a graph, thus providing efficiently computable bounds for this graph parameter.

.....

Table of Contents

Table of Figures

Acknowledgements

I wish to thank the following people:

My advisor Rachel Manber, who has been a source of encouragement, support, and constructive criticism. She is responsible for teaching me how to 'write', and for teaching me to ask a lot of questions.

Anne Condon, Richard Brualdi, Eric Bach, Michael Ferris, Debby Joseph, and Yannis Ioannidis for being on my examining committees. Special thanks are due to Anne for carefully reading my thesis, and pointing out numerous mistakes in the earlier drafts.

Kirk Pruhs for umpteen fruitful discussions, and for making the office a pleasant as well as stimulating atmosphere. With his innumerable brain teasers, he showed me not to trust my intuition.

Sam Bent and Udi Manber for being excellent teachers.

Paddy and Radiya for being such good friends. Their warmth, and affection will remain with me forever. It would be impossible for me to list out all the ways in which they have helped me.

Jorg for making 'workout' so enjoyable.

Sridhar for many movies, bicycle rides, and slide shows.

Meera for trying to teach me music and dancing.

Raghu, Muralikrishna, Judy, Prasad, Prabhu, Prashun, Martha, Randy, Uma, Karen, Mary, and Madan for many good times.

Narendran, Marty, Jon, and Joao for bearing with me at the office.

Madison has been a wonderful place to spend the graduate school years. This is mainly due to the numerous friendships I made here, many of which I will remember and cherish forever.

This thesis would not have been possible without the constant love and support of my parents, and my brother Ravi. Their encouraging and loving letters have constantly fueled my motivation and spurred me on during these years in Madison. To them I dedicate this thesis.

Chapter 1

Introduction

Graph theory has been an important branch of Combinatorics, and has been intensely researched for more than a hundred years. Graphs help model various situations that involve objects and the interconnections between them. For instance, we could model a map of all the highways in the United States with the cities as the objects and the highways between them as the interconnections. A graph consists of a set of vertices and a set of edges connecting pairs of vertices.

Computer scientists and mathematicians have examined graphs from an algorithmic point of view so that various questions concerning the modeled situations could be answered efficiently. For the road map example, a typical question could be to find a route to drive from Madison to San Fransisco that passes through Memphis and Las Vegas. The study of graphs has resulted in a rich variety of algorithms. Although some of the fundamental graph algorithms are quite old, many interesting and efficient algorithms for numerous problems have been found relatively recently. Algorithms for the stable marriage problem, the matching problem, the minimum spanning tree problem, the network flow problem, and the shortest path problems are notable examples of efficient graph algorithms.

Most of the graph problems can be simply stated. But many of these problems involve searching through a vast number of potential solutions to find an answer, and simply do not seem to be amenable to solution by efficient algorithms. It is most likely that any algorithm for these hard problems suffers a *combinatorial explosion*. These hard problems include problems for which an efficient solution would be very useful because of their numerous practical applications. The notion of hardness was formalized by Cook [Cook71] and Karp [Karp72] by defining a class of problems known as *NP-hard* problems. Informally, these are problems that are considered to be the hardest among the class of problems that can be solved in non-deterministic polynomial time. The *graph partitioning problem*, is an example of an NP-hard problem. It is the problem of partitioning the set of vertices of a given graph into two sets such that the number of edges connecting a vertex in one partition and a vertex in the other partition is minimized. It is commonly believed that the class of problems that can be solved in deterministic polynomial time (P) is a strict subset of the class of problems that can be solved in non-deterministic polynomial time (NP). Hence, it is unlikely that any polynomial-time algorithms can be found either for the graph partitioning problem, or for any NP-hard problem.

In Section 1.1, we discuss four ways of dealing with a problem that is known to be hard. In Section 1.2, we introduce an NP-hard problem known as the maximum k-colorable subgraph problem. Three out of the four approaches mentioned in Section 1.1 were tried on this NP-hard problem. Our results using the various approaches are briefly discussed in Section 1.3. After summarizing our results, we end this chapter by giving a brief overview of how the thesis is organized.

To facilitate the discussion here, we introduce a few basic definitions and a little notation. For most part, we define new terms as and when required. Occasionally, however, the reader may be referred to the *Appendix*, which is provided at the end of this report. This is resorted to, only to prevent any disruptions in the flow of the material and to improve readability. Let $G = (V, E)$ be a *simple graph* (i.e., a finite, undirected, loopless graph without multiple edges), with vertex set V of size n, and edge set E of size m. An *independent set* is a set of vertices in a graph none of which are adjacent to each other. The *stability number* of G, denoted by $\alpha(G)$, is the size of the maximum independent set in G. A *clique* is a set of vertices in a graph each of which is connected to the other by an edge. The *clique number* of G, denoted by $\omega(G)$, is the size of the largest clique in the graph. The *chromatic number* of G, denoted by $\chi(G)$, is defined as the minimum number of colors needed to color the vertices of G in such a way that adjacent vertices are assigned distinct colors. It is NP-hard to compute each of the graph parameters mentioned above, namely, $\omega(G)$, $\alpha(G)$, and $\chi(G)$.

1.1. Coping with NP-Hardness

Many NP-hard problems need to be solved in real-life applications on a daily basis. Hence, it is necessary to devise techniques to cope with the apparent intractability of NP-hard problems. There are several ways of dealing with NP-hard problems. We discuss four different approaches of coping with NP-hardness. One possible approach, which we call the *special classes approach*, can be used in problems where something more is known about its possible inputs. It is possible that the input graphs to a certain graph problem also satisfy some special properties. In a lot of practical applications it is known that inputs are from some special classes of graphs. In such cases, even though a problem may have been proved to be hard when the input is any arbitrary graph, the problem may have polynomial-time algorithms when the input is restricted to special classes of graphs. In such cases, one could look for solutions for the hard problems on the concerned special class of graphs by exploiting some of the structural properties of that class of graphs.

For example, certain register allocation problems for optimizing compilers, which optimize iterative programs, can be reduced to the maximum independent set problem with the constraint that the input graphs are from a special class of graphs known as the *circular-arc graphs*, a class of graphs that we define later in this chapter. The *maximum independent set problem* (abbreviated as the MIS problem) is the problem of finding the largest subset of vertices that induce an independent set. The maximum independent set problem is an NP-hard on general graphs. However, the maximum independent set problem can be solved in polynomial time when the input is a circular-arc graph [Ga73]. In the road maps example, if it is known that the highways do not intersect except at cities, then the input can be modeled as a *planar graph* (see *Appendix* for definition) since the graph can be embedded on a plane surface. Many NP-hard problems are known to have efficient solutions on planar graphs. The problem of finding the maximum clique in a

graph is NP-hard for general graphs, but has efficient algorithms on planar graphs. In fact, by Kuratowski's theorem (see [Har69]) it is known that in a planar graph there can be no cliques of size greater than 4.

A second possible approach, which we call the *approximation approach*, is to design algorithms that may not always find the optimal solution, but may find the optimal solution with high probability, or may find a solution that is reasonably close to the optimal. Such algorithms are called *heuristic algorithms*. Furthermore, if, for any input, it is possible to prove a bound on how good the solution found is going to be in comparison to the optimal solution, then such an algorithm is called an *approximation algorithm*.

The second approach can also be useful in cases where the first special classes approach fails to be of help. In other words, it may turn out that some NP-hard problem is NP-hard even if the inputs are restricted to a special class of graphs. In such cases, it is quite likely that we might get better approximation algorithms for this class of graphs, in the sense that we could design approximation algorithms, that find approximate solutions that are closer to the optimal than the solutions found for general graphs. Finding a maximum independent set remains NP-hard even for the class of planar graphs. While no good approximation algorithm for this problem is known for general graphs, when the inputs are restricted to planar graphs, there exists an algorithm that finds an independent set that is at least a fifth of the size of a maximum independent set [CNS81, F84]. Thus, specific structural properties can be exploited to design better approximation algorithms on special classes of graphs.

Designing fast approximation algorithms could also be useful for problems where the only polynomial-time algorithms (which find the exact solution) known have a very high time complexity. This is true in cases where the algorithms use the ellipsoid method. The Ellipsoid Method (see for example [Sho77], [JN76], [GLS81]) is a technique used for convex function minimization. The precision required by the ellipsoid method is very large. Hence polynomial-time algorithms using this method may not be very practical. In the interest of practicality, it makes sense to try to design approximation algorithms for such problems.

The third possible approach, which we call the *estimation approach*, to deal with inherent hardness of a problem is to find good estimates or bounds for the function that is being optimized - either lower bounds or upper bounds. This is a reasonable approach in cases where the NP-hard problem is either a maximization or a minimization problem. Even if the problem is a decision problem, often it has a corresponding optimization version that is NP-hard. One reason to find bounds is that it could reduce the search space, and consequently could lead to improvements in the time complexity of algorithms that do exhaustive search. Here is a trivial example : if we had an exponential time algorithm to find the largest clique in a graph, and if we can quickly find reasonable upper and lower bounds, then we could restrict our algorithms to look for solutions whose sizes lie within these bounds. It is interesting to note that designing approximation algorithms also help in finding estimates for the function to be optimized since proving a bound for the worst possible ratio of the size of the optimal solution to that of the approximate solution found by the approximation algorithm could provide bounds on the function to be optimized.

A fourth possible approach, which we call the *average case approach*, to deal with a NP-hard problem is to design algorithms that work well on the average. A problem may be hard because of a small proportion of difficult instances of the problem. In such cases, it is possible to look for algorithms that run rapidly most of the time (that is, they have a small average running time). These algorithms may perform very badly on some pathological problem instances, but may be efficient on most instances of the problem. Analysis of such algorithms usually assume that each instance of the problem occurs with a certain associated probability. The study of *random graphs* (see Bollobas [Bol85]) is an example of this approach, where the assumption is that inputs for graph problems follow a certain probability distribution. We did not try this approach on the maximum k-colorable subgraph problem.

1.2. The Maximum k-Colorable Subgraph Problem

In this thesis we consider an NP-hard problem known as the Maximum k-Colorable Subgraph Problem (abbreviated as the MkCS problem), and discuss results of the three approaches that we tried on this problem. The *maximum* k-*colorable subgraph problem* can be described as follows: given a graph G and k colors $1, 2, \ldots, k$, determine the maximum number of vertices that can be colored with k colors with the condition that if adjacent vertices are colored, then they are colored with distinct colors. It is a simple corollary of the result by Lewis and Yannakakis [LY80] that this problem is NP-hard for all values of k between 1 and n, the number of vertices. It is clear that the vertices that are colored with the same color cannot be adjacent to each other and hence form an independent set. Hence the MkCS problem can also be thought of as the problem of finding the maximum induced k-partite subgraph in the graph.

The MkCS problem can be viewed as part of a spectrum of problems of the same general type. At one end of the spectrum is the *maximum independent set problem*, which can be viewed as the MkCS problem with $k = 1$. The MkCS problem with $k = 2$, also known as the *maximum induced bipartite subgraph problem*, lies in this spectrum too. Another interesting problem in this spectrum of problems is the MkCS problem when k is a fixed constant. Finally, at the other end of this spectrum of problems lies the MkCS problem for general k. The *graph coloring problem* also lies in this spectrum. Given a graph G, and an integer k, the graph coloring problem is the problem of determining if the entire graph G can be colored with k colors. Note that if we can solve the MkCS problem efficiently we can also solve the general coloring problem efficiently. This is because, if we want to determine if a given graph can be colored with k colors, we could instead compute the maximum k-colorable subgraph in the graph, and check whether this subgraph is the entire graph. On the other hand, if there is an efficient solution for the general coloring problem, it does not imply an efficient solution for the MkCS problem.

Having put the problem in perspective, we will now motivate the MkCS problem further by describing a problem from VLSI that can be reduced to the MkCS problem, and which motivated some of our research

in the initial stages. This problem is called the *Via Minimization Problem* [MS84]. The problem can be reduced to the MkCS problem, for the case $k = 2$, where the inputs are not any general graph, but are restricted to a class of graphs called the *circle graphs* [Go80]. We will also describe the class of circle graphs shortly.

The via minimization problem is described as follows. We are given a routing region M that is bounded by a continuous closed curve C. A routing region is the area on a VLSI chip where the connections between terminals are laid out. If C is traversed in the clockwise direction, the region M lies to the right of the curve C. We shall assume that M is a rectilinear polygon. There are a set of terminals T that lie on the bounding curve C. Specified pairs of terminals need to be connected by placing wire segments inside the routing region M. In a more general setting, we might need to connect sets of terminals, instead of just pairs of terminals. Each terminal is assumed to be involved in exactly one pairing. This routing region can consist of several layers. If the routing region M has only one layer, then we may not be able to make all the necessary connections between the terminals, since we have the restriction that none of the connecting wire segments can intersect. In most current technologies, however, two layers are available for routing. Hence we shall assume that M has two layers, and that all the terminals are available in both layers. This means that a particular connection between a pair of terminals can be made in either of the two layers as long as it does not intersect any other connection made in the same layer. Even with two layers, it may not be possible to make all the necessary connections. For example, if we need to make three connections such that all three must intersect each other, then two layers are not enough to connect them. After making the connections on the two layers, the leftover connections can be made using *Vias*. A via is an area where both layers are electrically interconnected. Hence, when a connection cannot be made on either of the layers without crossing other connections, they can still be connected by making part of the connection on one layer, then using a via, after which, the rest of the connection takes place on the other layer. If n connections were required and if m of these are placed on the two layers, then Marek-Sadowska [MS84] showed that the rest of the connections can be made using $(n - m)$ vias. However, vias affect the performance of the circuit in a negative way. The *Via Minimization Problem* is the problem of minimizing the number of vias used to achieve the necessary connections. Equivalently, the problem is to maximize the sum of the number of connections placed on the two layers.

Before showing how to transform the via minimization problem into a problem on the class of circle graphs, we need to define circle graphs. The class of circle graphs can be defined in terms of the concept of intersection graphs. A graph G is called an *intersection graph* if there is a one-to-one correspondence between the vertex set V and a family of non-empty sets, $\mathcal{F}$, such that two vertices of the graph are adjacent iff their corresponding sets intersect. The set $\mathcal{F}$ forms the *intersection model* for the graph. Every graph is an intersection graph for some family of sets [Mar45]. *Circle graphs* are the intersection graphs of a family of chords of a circle (Here intersection is in the sense of geometric line intersection). The associated diagram showing the circle and the family of chords is called the *circle diagram.*

Going back to the via minimization problem, we think of the routing region as being circular, and the terminals as points on the bounding circle. The connections to be made can be thought of as chords of the bounding circle. What we have is a circle diagram and it is clear that if two of the chords in the circle diagram intersect then the corresponding connecting wire segments must also intersect and the corresponding connections have to be put on different layers. The circle diagram has a corresponding circle graph associated with it. Furthermore, the problem of minimizing the number of vias is equivalent to finding the largest 2-colorable subgraph (or the largest induced bipartite subgraph) in the corresponding circle graph. It is worth mentioning that if advancements in technology were to allow more than two layers for the routing region, the corresponding via minimization problem would reduce to the MkCS problem, where k is the number of layers in the routing region.

It was recently shown [SL87] that the via minimization problem is NP-hard.

1.3. Our Results

In the previous section we described an NP-hard problem. Based on the discussions from the earlier sections, we have studied several of the approaches to deal with the apparent inherent hardness of the MkCS problem. We now discuss some of the approaches that we took with regard to this problem and mention the results that are presented in the later chapters of this thesis.

1.3.1. Algorithms on Special Classes of Graphs

The first approach that we discussed was to design algorithms on special classes of graphs. The MkCS problem has practical applications where the inputs are restricted to special classes of graphs. We have several new results using the first approach (the special classes approach). As a first step in this direction, and due to our interest in the via minimization problem, we looked at the problem of finding the largest 2-colorable subgraph in a circle graph. Since the solution to this problem was elusive, we looked at classes of graphs that were related to the class of circle graphs with the hope that the solution on related classes of graphs would help us solve it on circle graphs.

We studied the classes of *interval graphs*, *circular-arc graphs*, and *tolerance graphs*. Below we define these classes, provide some historical background and applications for these classes, and examine the relationships between them.

An *interval graph* is the intersection graph of a family of intervals on a linearly ordered set. Interval graphs were first studied in the context of genetic applications where the problem was reduced to that of recognizing interval graphs [Go80]. Many problems in scheduling applications have also been reduced to

problems on interval graphs [Go80]. For example, suppose we have a set of courses $C = \{c_i\}$ that are offered by a university, and we know the time slots T_i when course c_i is to be held, and we would like to assign classrooms for these courses in such a way that if two courses have overlapping time slots then different classrooms are assigned for them. It is easily seen that an assignment of classrooms for the courses in C is equivalent to the problem of coloring an interval graph. Gilmore and Hoffman [GH64] proved that an interval graph is also a *chordal graph*, that is, it does not have a chordless 4-cycle (see *Appendix* for detailed definition). They also gave an interesting characterization of an interval graph based on an ordering of all their maximal cliques [GH64]. Later, Booth and Lueker [BL76] designed a linear time recognition algorithm for interval graphs. Their algorithm also obtains the interval representation of the interval graph as a by-product. It was shown [GLL82] that a simple greedy algorithm can compute the largest independent set in an interval graph. We show that the greedy algorithm can also be extended to solve the maximum induced bipartite subgraph problem on interval graphs. Like the maximum independent set algorithm, our algorithm has a time complexity of $\mathcal{O}(n \log n)$. Yannakakis and Gavril [YG87] showed that, in fact, the greedy algorithm can be applied to solve the MkCS problem on interval graphs.

We next looked at a generalization of the class of interval graphs known as the class of circular-arc graphs. A *Circular-Arc Graph* is the intersection graph of a family of intervals that lie on a circle (i.e., they are circular arcs). Coloring problems for circular-arc graphs have applications in optimizing compilers where the number of registers used needs to be minimized for iterative program segments. This and other applications of circular-arc graphs are mentioned in Golumbic [Go80]. Circular-arc graphs are natural generalizations of interval graphs, and the class of circular-arc graphs properly contains the class of interval graphs. Tucker gave a characterization of circular-arc graphs in terms of the pattern of ones in the adjacency matrix of such a graph [Tuc71], and later showed an efficient test for circular-arc graph recognition [Tuc80]. Tucker's $\mathcal{O}(n^3)$ algorithm also obtained the circular-arc representation of a circular-arc graph. Gavril [Ga74] gave solutions for a whole range of problems on circular-arc graphs. He gave polynomial-time algorithms for the maximum independent set problem, the maximum clique problem, and the minimum clique cover. These algorithms were not very efficient, and they were subsequently improved by Gupta, Lee, and Leung [GLL82], and later by Masuda and Nakajima [MN88]. All these algorithms assume that the circular-arc representation of the graph is also provided as input. Golumbic and Hammer [GH86] gave efficient algorithms when no circular-arc representation is provided as input to these algorithms. Garey, Johnson, Miller, and Papadimitriou [GJMP80] showed that although the maximum independent set problem and the maximum clique problem can be solved in polynomial time on circular-arc graphs, the coloring problem is NP-hard on this class of graphs. We give efficient algorithms to compute the maximum induced bipartite subgraph in a circular-arc graph. We solved the problem of finding the largest 2-colorable subgraph (or induced bipartite subgraph) in a circular-arc graph. This algorithm can be considered to be an extension of the algorithm to find the largest independent set in a circular-arc graph [GLL82].

It may not be immediately clear how the class of circle graphs is related to the class of interval and circular-arc graphs. These relations will become apparent when we state an equivalent definition of circle graphs. Consider a set of intervals on a linearly ordered set. Two intervals are said to *overlap* iff the two intervals have a non-empty intersection but neither is contained in the other. If there is a one-to-one correspondence between the vertices in V and the intervals in I such that two vertices are adjacent iff the corresponding intervals overlap, then such a graph is called an *overlap graph.*

The circle graphs are equivalent to the class of overlap graphs [Go80]. Given a circle with chords, choose a point p on the circle that is not an end point of a chord. Open out the circle at p into a straight line. Now each of the chords looks like an interval on the straight line. Furthermore, two chords intersect iff their corresponding intervals overlap according to the definition above. Hence the overlap graph corresponding to the set of intervals obtained by opening out the circle at p is identical to the circle graph corresponding to the initial set of chords.

This shows that the class of overlap graphs is closely related to both the class of interval graphs and the class of circular-arc graphs. The first application for circle graphs was for the problem of sorting permutations using stacks ([Go80], [EI71], [Tar72]). Sorting a permutation with a minimum number of stacks is equivalent to finding the chromatic number of a circle graph. Gavril [Ga73] demonstrated polynomial-time algorithms to compute the largest independent set and the largest clique in a given circle graph, while Garey, Johnson, Miller, and Papadimitriou [GJMP80] proved that coloring is an NP-hard problem on circle graphs. The recognition problem for circle graphs was an open problem for a long time, before a polynomial-time algorithm was recently demonstrated simultaneously by Supowit, Gabor, and Hsu [GHS85], and by Bouchet [Bou87]. Both the algorithms have a time complexity of $\mathcal{O}(nm)$. Our attempts to solve the via minimization problem were unsuccessful. Naclerio, Masuda, and Nakajima [NMN87] solved an easier problem by showing that if the positions of the interconnections for the via minimization problem is fixed, then the problem of minimizing the number of vias can be solved in polynomial time. It was not until recently that Sarrafzadeh and Lee [SL87] proved the via minimization problem to be an NP-hard problem. Their reduction was from a problem known as the Planar SAT problem. Thus they showed that it is NP-hard to compute the maximum induced bipartite subgraph in a circle graph.

The next class of graphs we looked at is known as the class of *tolerance graphs.* This is another class of graphs that generalizes the class of interval graphs. A tolerance graph is similar to an interval graph in the sense that each vertex of a tolerance graph corresponds to an interval on a linearly ordered set, just like a vertex in an interval graph. Each interval is further associated with a number called its tolerance. Two intervals are said to intersect iff the amount of their intersection is at least as large as the smaller of the tolerances of the two intervals. Hence the class of tolerance graphs generalizes the class of interval graphs in the sense that it generalizes the notion of intersection of intervals by allowing some amount of tolerance on their intersections. Tolerance graphs were introduced by Golumbic and Monma [GM82]. Golumbic, Monma, and Trotter [GMT84] proved several properties about tolerance graphs. The recognition problem

for tolerance graphs remains an open problem. The algorithmic aspects of tolerance have not been studied at all. We designed efficient algorithms for various problems on tolerance graphs including the problem of finding the largest independent set, finding the largest bipartite subgraph, and finding the maximum k-colorable subgraph for fixed k. We also give an efficient algorithm to compute the largest clique in a tolerance graph, which can be thought of as computing the maximum independent set in the complement of a tolerance graph.

Dagan, Golumbic, and Pinter [DGP86] introduced a new class of graphs called the trapezoidal graphs, which has applications in VLSI layout problems. These are intersection graphs of a set of trapezoids. They [DGP86] also solved the coloring problem for these graphs. As it turns out the algorithms we used for tolerance graphs can also be applied to the class of trapezoidal graphs with minor modifications. This gives us algorithms to solve the problem of determining the largest independent set, and the largest k-colorable subgraph for any k between 2 and n.

We note that all the special classes of graphs that we studied, and for which we tried to solve the MkCS problem have polynomial-time algorithms to compute the largest independent set. The classes of graphs were chosen with this intention because the independent set problem is very closely related to the MkCS problem.

The classes of graphs mentioned above are quite closely related. One way to think about them is as a set of classes of graphs related to the class of interval graphs. The class of interval graphs is a subclass of the class of circular-arc graphs since the intervals on a straight line can be thought of as lying on a circle where all of the arcs lie within just some fraction of the circle. Overlap graphs are also quite similar to interval graphs. The difference is that, in interval graphs edges correspond to intersecting intervals, whereas, in overlap graphs edges correspond to overlapping intervals. The class of tolerance graphs as well as the class of trapezoidal graphs properly contain the class of interval graphs.

1.3.2. Approximation Algorithms

The second approach that we discussed was to design heuristic or approximation algorithms. To this end, we prove that, as in the case of the maximum independent set problem, it is unlikely that any "good" approximation algorithm can be designed for the MkCS problem for general graphs. Of course, this does not prevent the possibility of "good" approximation algorithms existing for special classes of graphs.

Since the MkCS problem on circle graphs was proved to be NP-hard, we looked at approximation algorithms for this case. We show that the naive or the greedy approximation algorithm for the MkCS problem performs quite well on circle graphs, and guarantees a solution which is at least half the size of the optimal solution. It is worth noting that this approximation algorithm guarantees the bound for all graphs,

but runs in polynomial time only for those classes of graphs for which the maximum independent set problem is polynomial-time computable.

Another problem for which we have an approximation algorithm is the coloring problem on tolerance graphs. The number of colors used by our algorithm may be away from the optimal number of colors by at most 1 color.

1.3.3. The Estimation Approach Using Algebraic Methods

We now describe some of our results using the third approach or the estimation approach.

Lovász showed the existence of a polynomial-time computable function, whose value always lies between the size of the largest clique and the chromatic number [Lov86]. This function $\vartheta(G)$ also gives a polynomial-time computable bound for the chromatic number and the clique number of a graph. The proof of the existence of this function uses powerful linear algebraic techniques and uses the ellipsoid algorithm [GLS81] for its computation. It may also be noted that computing Lovász's function for the complement of a given graph gives upper bounds for the size of the largest independent set in the original graph. Lovász's theorem is called a *sandwich theorem* since a polynomial-time computable function is sandwiched between two NP-hard graph invariants.

Inspired by Lovász's sandwich theorem, we obtain polynomial-time computable functions that estimate certain interesting graph parameters. We prove the existence of polynomial-time computable functions, denoted by $\vartheta_k(G)$, $1 \leq k \leq n$, which are sandwiched between two graph parameters $\omega_k(G)$ and $\chi_k(G)$. The parameter $\omega_k(G)$ is the size of the largest induced subgraph that can be covered with k cliques. The parameter $\chi_k(G)$ is a generalization of the chromatic number. We shall describe these parameters in more detail later. Since our results give upper bounds for the parameter $\omega_k(G)$, and since the cliques in a graph correspond to the independent sets in its complement, it is clear that computing our function for the complement of a given graph, gives polynomial-time computable bounds for the size of the maximum k-colorable subgraph in the original graph.

We would like to make special mention of the intimate link between graph theory and linear algebra, which has sparked off a whole new direction of research - the area of Algebraic Graph Theory ([B74], [CDS80]). This area examines graph properties by studying matrices related to a graph, most notably the *adjacency matrix* of a graph. Here we define the *adjacency matrix* of a graph with n vertices to be the n by n matrix with the ijth entry equal to 1 iff vertices i and j share an edge or $i = j$, and with all other entries being zero. Studying this matrix as an algebraic entity reveals interesting information about the graph.

A lot of research has focused on the *spectra of graphs*, which is the set of eigenvalues of the adjacency matrix of a graph. In fact, Lovász's function ($\vartheta(G)$) as well as our functions ($\vartheta_k(G)$) are related to the eigenvalues of a matrix, which is closely related to the adjacency matrix of the graph. One of the first

non-trivial results on the spectra of graphs was due to Wilf [Wil67], who showed that the largest eigenvalue of the adjacency matrix is at least as large as the chromatic number of a graph, and consequently at least as large as the size of the largest clique in the graph. An interesting by-product of our research on the use of algebraic techniques is that we generalize Wilf's result by proving that the sum of the k largest eigenvalues of the adjacency matrix of the graph is at least as large as the size of the largest induced subgraph that can be covered with k cliques. Equivalently, the sum of the k largest eigenvalues of the adjacency matrix of the complement of the graph is at least as large as the size of the maximum k-colorable subgraph in a graph. Besides giving a simple bound on the size of $\omega_k(G)$, and the size of the maximum k-colorable subgraph, this result is interesting in its own right, since it puts the inequality in Wilf's result in a more general setting.

Algebraic methods are also very useful in the design of algorithms. An example of an algorithm using algebraic methods is Lovász's use of the sandwich theorem to design an algorithm [Lov86] for finding a maximum independent set in a special class of graphs called the *perfect graphs*. A *perfect graph* is a graph for which the chromatic number is equal to the clique number for every induced subgraph of the graph. This example shows that finding good polynomial-time computable estimates of graph parameters can help design algorithms, if the inequalities are satisfied by equality for some interesting cases. This example also shows that designing sandwich theorems can be a powerful technique in designing algorithms.

In spite of the fact that there exists a polynomial-time algorithm for the maximum independent set problem, it is known that the MkCS problem on perfect graphs is NP-hard [YG87]. Hence it is unlikely that our sandwich theorems would help to compute the size of the largest k-colorable subgraph in a graph. However, our inequalities do lead to an algorithm to find the largest k-colorable subgraph in a subclass of perfect graphs. In a later chapter we construct an example of a graph for which our sandwich theorem is useful. This graph can be generalized to show that our sandwich theorem is useful for a family of graphs.

We conclude this section with a discussion on the class of perfect graphs. The class of *perfect graphs* has been of interest to graph theorists for a long time. It is obvious that the chromatic number of a graph is at least as large as the size of the largest clique, since every vertex in the clique must have distinct colors in any coloring of the graph. This generated interest in graphs for which these two parameters, namely, the chromatic number and the clique number (or the size of the largest clique) are equal. Berge defined a *perfect graph* to be a graph for which the chromatic number and the clique number are equal for every induced subgraph [Go80]. In 1972, Lovász [Lov72] proved the *Perfect Graph Theorem*, at least a decade after it was conjectured by Berge [Ber61]. The theorem states that the complement of a perfect graph is also perfect. The consequence of this theorem is that, for perfect graphs, the stability number or the size of the largest independent set and the clique cover number are also equal for every induced subgraph [Go80]. Berge made another conjecture [Ber62], which remains open, and which has become famous as the *Strong Perfect Graph Conjecture*. The conjecture states that a graph is perfect iff it does not have an induced subgraph isomorphic to an odd cycle (also called an *odd hole*) of size at least 5, or the complement of an odd cycle (also called an *odd antihole*) of size at least 5. The conjecture has been proved to be true for a

number of special classes of graphs [Go80]. Numerous alternate equivalent forms of the conjecture have been formulated [Go80]. The *Strong Perfect Graph Conjecture* remains one of the most fascinating open problems for the graph theorists. The strong perfect graph conjecture, if proved, would give another characterization of the class of perfect graphs. With the lack of a working characterization of the class of perfect graphs, it is not surprising that finding an efficient algorithm for recognizing perfect graphs is an open problem. Lovász proved that the recognition problem is in the class Co-NP [Lov83]. The class Co-NP is the class of problems whose complement can be computed in non-deterministic polynomial time. Hence, imperfectness can be tested in non-deterministic polynomial time. It is not even known whether testing for perfectness can be done in non-deterministic polynomial time.

Lovász proved that the largest clique in a perfect graph can be computed in polynomial time [Lov86]. This follows trivially from his sandwich theorem, since for a perfect graph, the size of the largest clique is equal to the chromatic number. Furthermore, since perfect graphs are closed under complementation [Lov72], Lovász also showed that there are polynomial-time algorithms to compute the largest independent set, the chromatic number, and the clique cover number in a perfect graph. Lovász's algorithms for perfect graphs was one of the first instances of algorithms that successfully exploited results from algebraic graph theory.

We discuss briefly how the class of perfect graphs is related to some of the classes we discussed earlier. Many of the classes we discussed earlier are subclasses of perfect graphs. Interval graphs are perfect [Go80]. The same is true of tolerance graphs [GMT84]. Circular-arc graphs are not perfect, but it is known that for circular-arc graphs, the size of a largest independent set is at most one less than the cardinality of the smallest clique cover. In that sense, they may be considered to be slightly imperfect [Ga74]. No similar result is known about circle graphs (Golumbic [Go80] calls them 'Not So Perfect Graphs').

We did not follow the fourth approach (or the average case approach), which was to design algorithms that work well on the average. We will discuss this as a potential direction for future research.

1.4. Summary

Summarizing the theme of our work, the aim of our study has been to gain more insight into the intractability of "hard" problems. We study the tractability of a few hard problems on many special classes of graphs. Our most interesting results are the consequence of applying techniques from linear algebra to study and estimate graph parameters, and to design algorithms. The special classes of graphs that we focused attention on were: *(1) Interval Graphs, (2) Circular-Arc Graphs, (3) Circle Graphs (or Overlap Graphs), (4) Tolerance Graphs, (5) Trapezoidal Graphs*, and *(6) Perfect Graphs.*

The conclusion is that, when the input to a hard problem is restricted to be from a special class of graphs, then it is possible that either efficient algorithms could be designed for them or better approximate solutions could be found.

D. S. Johnson [Joh85] made a survey of 11 interesting problems and their complexity on 30 special classes of graphs. Part of that summary of known results and open problems, which pertain to our report, are tabulated in the *Appendix*. A number of open problems have been mentioned in that survey and have partly motivated some of our work.

1.5. Outline of the Thesis

The thesis is organized as follows. In Chapter 2, we present algorithms for the maximum bipartite subgraph problem on interval graphs and circular-arc graphs. Chapter 3 is devoted to algorithms on tolerance graphs and trapezoidal graphs. We present polynomial-time algorithms for the maximum independent set problem, maximum bipartite subgraph problem, and the MkCS problem on both tolerance graphs and trapezoidal graphs. We also show an efficient algorithm for the maximum clique problem on tolerance graphs, and an approximation algorithm for coloring tolerance graphs. In Chapter 4, we discuss new estimation techniques for the size of the largest k-colorable subgraph in a general graph. We discuss how this gives a polynomial-time algorithm for a very special subclass of perfect graphs. This chapter also gives a generalization of Wilf's inequality, and Lovász's sandwich theorem. Finally, in Chapter 5 we discuss approximation algorithms, and show that the greedy approximation algorithm performs fairly well on classes of graphs for which the largest independent set can be computed in polynomial time. We conclude with a set of open problems and directions for future research in Chapter 6.

Chapter 2

Algorithms on Interval and Circular-Arc Graphs

2.1. Introduction

As defined earlier, a graph $G = (V, E)$ is an *Interval Graph* if it is the intersection graph of a family of intervals on a linearly ordered set. A graph is a *Circular-Arc Graph* if it is the intersection graph of a family of intervals that lie on a circle (circular arcs). These two classes of graphs have found numerous practical applications in areas ranging from scheduling and data storage to archeology and genetics [Go80].

In this chapter we give efficient algorithms for the Maximum Induced Bipartite Subgraph (MIBS) Problem on both interval graphs and circular-arc graphs. These algorithms run in times $\mathcal{O}(n \log n)$ and $\mathcal{O}(n^3)$ respectively. The algorithms assume that their inputs are intersection models for the given graphs. An intersection model for interval graphs can be computed in $\mathcal{O}(n+m)$ time [BL76]. An intersection model for circular-arc graphs can be found in $\mathcal{O}(n^3)$ [Tuc80].

It is known that many problems that are known to be hard on general graphs are solvable in polynomial time on interval graphs. These include the maximum independent set problem [GLL82], and the general coloring problem. However, for circular-arc graphs the general coloring problem is NP-Complete, while the other problems mentioned above are known to be solvable in polynomial time [Ga74] [GJMP80]. Sections 2.2 and 2.3 contain polynomial-time algorithms for the MIBS problem on interval graphs and on circular-arc graphs respectively. In Section 2.4 we discuss the time complexity of the algorithm for the MIBS problem given in Section 2.3. Finally, in Section 2.5, we discuss the complexity of the MkCS problem on interval and circular-arc graph.

2.2. Interval Graphs

Let $I = \{u_i = (a_i, b_i) : i = 1, \ldots, n\}$ be an intersection model for the interval graph $G(I) = (V, E)$. That is, there is a one-to-one correspondence between the intervals in I and the vertices of G such that $[v_i, v_j] \in E$ if and only if the corresponding intervals u_i and u_j have a nonempty intersection. We assume without loss of generality that all intervals in I are open and share no end point.

Interval graphs are *chordal*. That is, if there exists a cycle of length at least 4, then there exists an edge connecting two vertices that are not consecutive on the cycle. In other words, every cycle of length greater than 3 has a chord. It follows that every bipartite interval graph is a forest. Moreover, since every induced subgraph of an interval graph is an interval graph, every induced bipartite subgraph of an interval graph is acyclic.

Algorithm $\mathcal{A}$

Input: A set of intervals I given as a set of ordered pairs of end points.

Output: $S \subseteq I$ such that the subgraph $G(S)$ induced by the intervals in S is the largest induced bipartite subgraph in $G(I)$.

1. Sort the n right end points of the n intervals.
2. Let y_0 be the interval with the leftmost right end point.
3. $S \longleftarrow \{y_0\}$.
4. $I \longleftarrow I - \{y_0\}$.
5. **while** I is nonempty **do**
   6. Let y_1 be the interval in I with the leftmost right end point.
   7. Add y_1 to S, and remove it from I.
   8. If y_0 intersects y_1 then remove from I all intervals that intersect both y_0 and y_1.
   9. $y_0 \longleftarrow y_1$.

   endwhile

Figure 2.1: Algorithm for the MIBS problem on interval graphs

Algorithm $\mathcal{A}$ is shown below in Figure 2.1. It finds a maximum induced bipartite subgraph of a given interval graph. It assumes that both the graph and the interval representation are provided as input. It first sorts the n right end points of the n given intervals and then scans the intervals in that order (from left to right). The scanning process repeatedly chooses the interval with the leftmost right end point to be in the solution. The chosen interval is then deleted along with all the intervals that form triangles with the intervals that have been chosen to be in the solution.

Since the scanning process can be done in linear time, the time complexity of algorithm $\mathcal{A}$ is $\mathcal{O}(n \log n)$, as it is dominated by the time complexity of sorting, If the intervals were previously sorted by the right (or left) end points, then this algorithm would take only $\mathcal{O}(n)$ time. Clearly, this algorithm is simply the greedy algorithm. However, it is non-trivial to show that this algorithm produces the maximum induced bipartite subgraph in an interval graph.

Theorem 1: Algorithm $\mathcal{A}$ (shown in Figure 2.1) computes the maximum induced bipartite subgraph of an interval graph in time $\mathcal{O}(n \log n)$.

Theorem 1 will be proved by induction on the number of intervals in I. The basis of the induction is proved by showing that there exists an optimal solution containing the two intervals with the leftmost right end points. This is proved in Lemmas 2 and 3.

Lemma 2: Let I be a family of intervals on a linearly ordered set, and let y_0 be the interval in I with the leftmost right end point. Then there exists a solution to the MIBS problem for $G(I)$ that contains y_0.

Proof: Let J be any solution to the MIBS problem. Suppose J does not include y_0. Let y_0' be the interval in J with the leftmost right end point. The interval y_0' must intersect y_0, or else $J \cup \{y_0\}$ would be a solution of size larger than the size of J. We claim that $J' = J - \{y_0'\} \cup \{y_0\}$ is a solution to the MIBS problem for $G(I)$. It suffices to show that $G(J')$ is bipartite. Let v_0 and v_0' be the vertices corresponding to intervals y_0 and y_0' respectively. Suppose v is a vertex in $G(J)$ and v and v_0' lie in the same bipartition, that is $(v, v_0') \notin E$, then the interval corresponding to v must lie wholly to the right of y_0' and consequently must lie wholly to the right of y_0. It follows that $(v, v_0) \notin E$, and since $G(J)$ is bipartite so is $G(J')$. ∎

Lemma 3: Let I be a family of intervals on a linearly ordered set and let y_0 and y_1 be the two intervals with the leftmost right end points in I. Then there exists a solution to the MIBS problem for $G(I)$ that contains both y_0 and y_1.

Proof: By Lemma 1 there exists a solution J' that includes y_0. Suppose $y_1 \notin J'$. Let V_1 and V_2 be a bipartition of the vertex set of $B(J')$. Let J_1' and J_2' be a partition of J' that corresponds to the vertex sets V_1 and V_2. We may assume that $y_0 \in J_1'$. We want to replace an appropriate element of J' by y_1 thus constructing a solution that contains both y_0 and y_1. Let y_2 be the interval with the leftmost right end point in J_2' and let $J_2 = J_2' - \{y_2\} \cup \{y_1\}$ and $J_1 = J_1'$. All the intervals in $J_2' - \{y_2\}$ lie entirely to the right of y_2 and hence must lie entirely to the right of y_1. It follows that the intervals in J_2 do not intersect each other and the intervals in $J = J_1 \cup J_2$ induce a bipartite subgraph of cardinality $|J'|$. Thus J is a solution to the MIBS problem for $G(I)$ that contains both y_0 and y_1. ∎

Proof of Theorem 1: What we need to show is that the output of algorithm $\mathcal{A}$ on input I is a solution to the MIBS problem for $G(I)$. We use induction on the number n of intervals in I. The claim is trivially true for $n = 1$. Assume that the algorithm works correctly on all sets with less than n intervals and let $|I| = n$.

Let S' be the solution found by algorithm $\mathcal{A}$ on input I. Then S' is clearly bipartite. Denote by y_0 and y_1 the two intervals in I with the leftmost right end points. If y_0 and y_1 intersect, we define Δ as the set of intervals that intersect both of them, otherwise $\Delta = \emptyset$. Clearly $y_0, y_1 \in S'$. Moreover, algorithm $\mathcal{A}$ will return the solution $S' - \{y_0\}$ on the input $I' = I - \{y_0\} - \Delta$. By the inductive assumption $S' - \{y_0\}$ is a correct solution for I'.

Let S be any solution on input I containing y_0 and y_1. By Lemma 2 such a solution exists. Next we show that $S - \{y_0\}$ is a solution on input I'. This would complete the proof of the Theorem since it implies that $|S| = |S'|$.

Clearly $S - \{y_0\}$ induces a bipartite subgraph in $G(I')$. Assume that $S - \{y_0\}$ is not a maximum size solution on input I'. Let T' be a solution on input I' such that $|T'| > |S - \{y_0\}|$. By Lemma 1, we can assume that $y_1 \in T'$. Let $T = T' \cup \{y_0\}$. We claim that the intervals in T correspond to a bipartite subgraph of $G(I)$. Indeed, if any triangle is formed in $G(T)$, it must have y_0 as one of its vertices. But, if an interval intersects y_0, it also intersects y_1, because the right end point of y_0 lies to the left of the right end point of y_1. Hence any triangles in $G(T)$ must be formed by the vertices corresponding to y_0, y_1 and one other interval. But all such intervals are in Δ, and $\Delta \cap I' = \emptyset$. It follows that T induces a bipartite subgraph on input I. The fact that $|T| > |S|$ contradicts the assumption that S is a solution on input I. ∎

We conclude this section with the following simple observation on the nature of algorithm $\mathcal{A}$. The rightmost right end point of a set of intervals S is called Right-end(S).

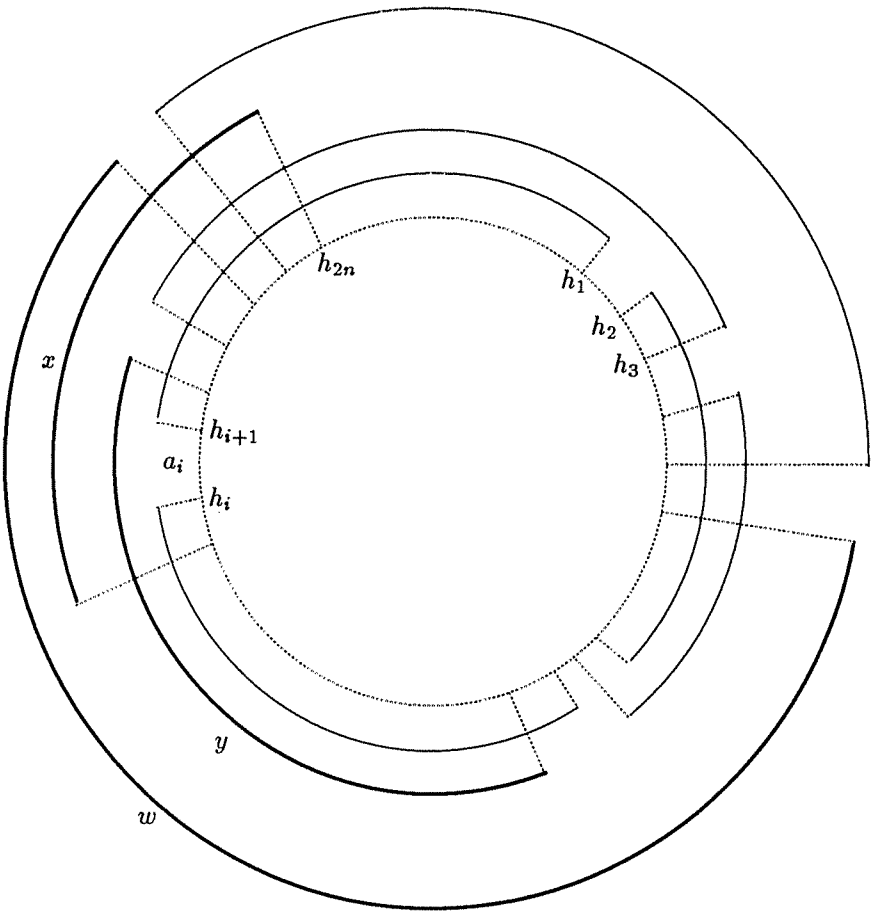
Lemma 4: Let I be a set of intervals on a straight line. When algorithm $\mathcal{A}$ is applied to input I it returns a solution S such that Right-end(S) is the least (i.e., leftmost) among all solutions.

Proof: Let $u_r = (x_r, y_r)$ be the interval in S with the rightmost right end point. Consider any solution S' on input I. Let $u'_r = (x'_r, y'_r)$ be the interval with the rightmost right end point in S'. Since algorithm $\mathcal{A}$ is correct, $|S| = |S'|$. Suppose $y'_r < y_r$. Now let R be the set of all intervals in I with right end points to the right of y'_r. Then $u_r \in R$ and $R \cap S' = \emptyset$. Also, let $T = R \cap S$. Since $u_r \in T$, we have $|T| \geq 1$. Clearly, S' is a solution on input $I - R$. However, on input $I - R$ algorithm $\mathcal{A}$ will find the solution $S - T$, which is of suboptimal cardinality ($< |S'|$), thus contradicting the correctness of algorithm $\mathcal{A}$. ∎

In an independent study, Yeh and LaPaugh [YL87] have developed a similar algorithm for the maximum bipartite subgraph problem on interval graphs. They mention a VLSI application for this problem known as the 2-*Track Assignment Problem*. If non-overlapping intervals can be placed on a "track", then the 2-track assignment problem, which is the problem of packing the maximum number of intervals on two tracks, is exactly the maximum bipartite subgraph problem.

2.3. Circular-arc Graphs

In this section we present an algorithm for solving the maximum induced bipartite subgraph problem for circular-arc graphs. The input for the algorithm is a collection $I = \{u_i = (x_i, y_i) : i = 1, \ldots, n\}$ of intervals or arcs on a circle (taken in the clockwise direction). The circular-arc graph corresponding to I is denoted by $G(I)$. Let $h_1, h_2, \ldots, h_{2n}, h_{2n+1} = h_1$, be an ordering of the end points of the circular arcs in the clockwise direction on the circle. Denote the interval (h_i, h_{i+1}) by a_i for $1 \leq i \leq 2n$. We call each of the a_i an *elementary segment*. If an elementary segment a_i is such that h_i is a right end point of some interval (or closing end point), and h_{i+1} is a left end point of another interval (or opening end point), then we call such an a_i a *critical segment*. Let W_i be the set of all arcs in I that contain interval a_i, and let $I_i = I - W_i$.



$a_i = (h_i, h_{i+1})$

$W_i = \{w, x, y\}$

Figure 2.2: An Example of a Circular-Arc Graph

In Figure 2.2, we show an example of a circular-arc graph. We also show a critical segment $a_i = (h_i, h_{i+1})$, with the corresponding set W_i shown as a set of thicker circular arcs.

Let S be a collection of circular arcs such that $G(S)$ is a solution to the MIBS Problem for $G(I)$. Hence $G(S)$ consists of two independent sets. Since the arcs corresponding to an independent set in $G(I)$ cannot cover the entire circle, there exists an interval a_{i_0} such that W_{i_0} has at most one interval from S. If there exists an interval a_{i_0} such that $W_{i_0} \cap S = \emptyset$, then finding a maximum bipartite subgraph for $G(I)$ is equivalent to finding one for the interval graph $G(I_{i_0})$. Otherwise, there exists an interval a_{i_0} such that $|W_{i_0} \cap S| = 1$.

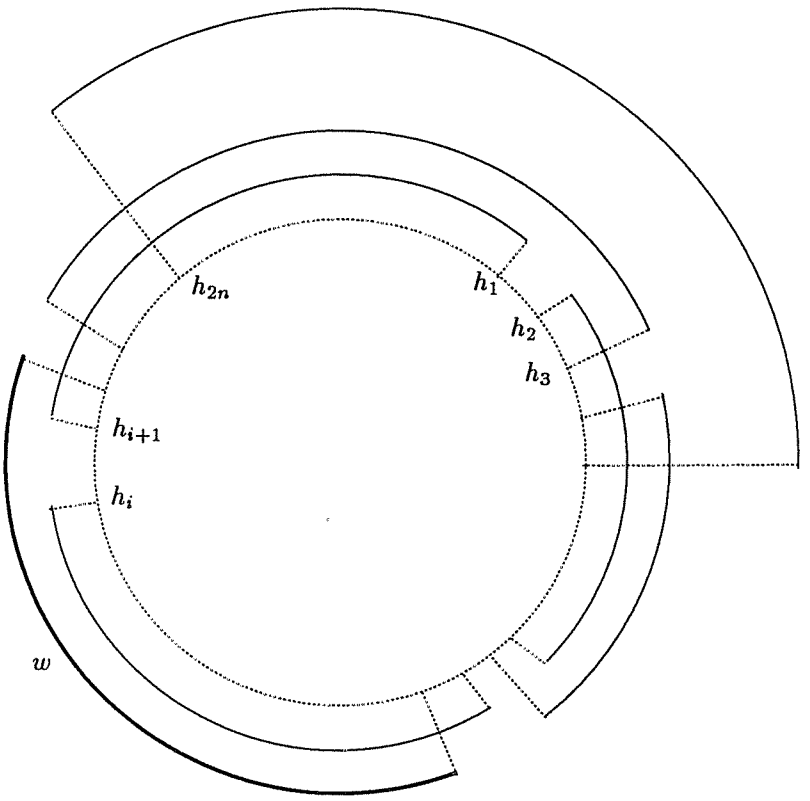


Figure 2.3: An elementary circular-arc graph with circular interval w

Let $W_{i_0} \cap S = \{w\}$. In that case, finding a maximum bipartite subgraph for $G(I)$ is equivalent to finding one for $G(J)$, where $J = I_{i_0} \cup \{w\}$, for some $w \in W_{i_0}$.

Now $G(J)$ is a special kind of circular-arc graph. We call such a graph an *Elementary Circular-Arc Graph*. More precisely, an elementary circular-arc graph is a pair [$G(J)$,w] such that $G(J)$ is a circular-arc graph, w is in J, and $G(J - \{w\})$ is an interval graph. The arc w is called here the *Circular Interval*. Figure 2.3 shows an example of an elementary circular-arc graph. If the interval w, which is marked with a thicker arc, is removed then we are left with a set of intervals that can be drawn on a straight line, hence forming a representation for an interval graph.

Thus the MIBS problem for circular-arc graphs can be reduced to a set of sub-problems, each of which is an MIBS problem on an interval graph or on an elementary circular-arc graph.

Algorithm $\mathcal{B}$ is shown below in Figure 2.4. It extends algorithm $\mathcal{A}$ to work on elementary circular-arc graphs. It takes as input an intersection model of an elementary circular-arc graph $[G(I), w]$ and finds a solution S of maximum cardinality such that $G(S)$ is an induced bipartite subgraph and $w \in S$.

Let $[G(J), w]$ be an elementary circular-arc graph. When the circular interval w is removed from J, we are left with a set of intervals that correspond to an interval graph. This implies that within interval w, there is an elementary interval $a = (y_1, y_2)$ such that no interval from $J = I - \{w\}$ intersects a. Hence all intervals in J lie in the interval (y_2, y_1). Without loss of generality, we assume that (y_2, y_1) lies on a straight line and we apply our usual notions of 'left' and 'right' to the intervals in J. When we make an 'ordered scan' of the intervals we mean a clockwise scan of the right end points of the intervals starting from y_2.

Algorithm $\mathcal{B}$

Input: A set of intervals, I and an interval w in I such that $[G(I), w]$ is an elementary circular-arc graph.

Output: A set of intervals $S \subseteq I$ with the property that S is the largest subset such that $G(S)$ is bipartite and $w \in S$.

1. Sort the intervals in $J = I - \{w\}$ according to the order in which the right end points appear in an ordered scan.
2. Let u_0 be the interval in J with the leftmost right end point.
3. $S \longleftarrow \{w, u_0\}$.
4. Remove w and u_0 from I and also all the intervals that form triangles with them.
5. **while** I is nonempty **do**
   6. Let v be the interval in I with the leftmost right end point.
   7. Put v in S and remove it from I.
   8. Remove from I all intervals that form odd cycles with intervals from S.

   endwhile
9. Return S as the solution.

Figure 2.4: Algorithm for the MIBS problem on elementary circular-arc graphs

In step 8 of algorithm $\mathcal{B}$, we remove all intervals in I that form odd cycles with intervals in S. The algorithm runs in time $\mathcal{O}(n \log n)$. Note that if the input is already sorted according to the right end points of the intervals then the algorithm will run in time $\mathcal{O}(n)$.

Lemma 5: Let I be the set of circular intervals corresponding to an elementary circular-arc graph $[G(I), w]$. Let u_0 be the interval in $J = I - \{w\}$ whose right end point is encountered first. Then among the solutions which include w, there exists a solution S' which includes u_0.

Proof: Let S be any solution for input I which includes w. Suppose $u_0 \notin S$. Denote the two bipartitions of $G(S)$ by V_1 and V_2 and denote their corresponding interval sets by J_1 and J_2. Without loss of generality we assume that $w \in J_1$. Let v be the interval in J_2 with the leftmost right end point. Let $S' = S - \{v\} \cup \{u_0\}$. If $x \in J_2 - \{v\}$ then x does not intersect v and hence x lies completely to the right of v. Consequently, it lies

completely to the right of u_0. This implies that $G(S')$ is bipartite and hence S' is a solution which includes both w and u_0. ∎

Lemma 6: Algorithm $\mathcal{B}$ correctly computes the maximum induced bipartite subgraph in an elementary circular-arc graph.

Proof: We give only a rough sketch of the proof since it is similar to the proof of correctness of algorithm $\mathcal{A}$ (Theorem 3). We use induction on the number of intervals in input J. Let S' be the solution returned by the algorithm and let S be any solution which includes the circular interval w and the interval with the leftmost right end point, u_0. Let Δ be the set of intervals in J which form triangles in $G(J)$ with w and u_0.

Following the lines of the proof of Theorem 3, we can show that $S' - \{u_0\}$ and $S - \{u_0\}$ are both solutions on input $J - \Delta - \{u_0\}$ and hence must be of the same size. This implies that S and S' are also of same size, thus proving the correctness of algorithm $\mathcal{B}$. ∎

Algorithm $\mathcal{B}$ solves the MIBS problem for a special kind of circular-arc graph. We now present algorithm $\mathcal{C}$ that will solve the MIBS problem for any circular-arc graph. Algorithm $\mathcal{C}$ is shown in Figure 2.5 below.

Algorithm $\mathcal{C}$

Input: A set of intervals, I, lying on a circle.

Output: $S \subseteq I$ such that the subgraph $G(S)$ is the largest induced bipartite subgraph in the circular-arc graph, $G(I)$.

1. **For** i, $1 \leq i \leq 2n$ **do**
   2. Run algorithm $\mathcal{A}$ on I_i to obtain solution S_{i0}.
   3. Let $W_i = \{w_{i1}, \ldots, w_{ik}\}$.
   4. **For each** $w_{ij} \in W_i$ run algorithm $\mathcal{B}$ on the elementary circular-arc graph $[G(I_i \cup \{w_{ij}\}), w_{ij}]$. Denote its output by S_{ij}.
   5. Pick S_i to be the largest of the solutions $S_{i0}, S_{i1}, \ldots, S_{ik}$.

   endfor
6. Return S, the largest of the solutions $S_1, \ldots, S_{2n}$.

Figure 2.5: Algorithm for the MIBS problem on circular-arc graphs

Theorem 7: Algorithm $\mathcal{C}$ correctly computes the maximum induced bipartite subgraph in any circular-arc graph.

Proof: As noted earlier, for every solution, S, to the MIBS problem for a circular-arc graph, there exists an m, $1 \leq m \leq 2n$ such that S has at most one interval from W_m.

If there exists a solution S and an integer m, $1 \leq m \leq 2n$ such that S contains no interval from W_m, then S is also a solution to $G(I_m)$. Hence, algorithm $\mathcal{C}$ will find a solution of size $|S|$ after executing step 2 in the m^{th} iteration. Otherwise, $\forall m$, $1 \leq m \leq 2n$ and for every solution S, $|S \cap W_m| \geq 1$. It follows that there exists an m, $1 \leq m \leq 2n$ such that $W_m \cap S = \{w_{mj}\}$, for some $w_{mj} \in W_m$. This implies that S is also a solution for $I_{mj} = I_m \cup \{w_{mj}\}$. Since I_m corresponds to an interval graph, the intervals in I_{mj} correspond to an elementary circular-arc graph with w_{mj} as its circular interval. Hence, algorithm $\mathcal{C}$ would correctly find a solution of size $|S|$ after the execution of step 4 during its m^{th} iteration. ∎

2.4. Worst-Case Time Complexity of the MIBS Algorithm on Circular-Arc Graphs

Each of the W_i can be of size $\mathcal{O}(n)$. Hence, there could be $\mathcal{O}(n)$ calls to algorithm $\mathcal{A}$ and $\mathcal{O}(n^2)$ calls to algorithm $\mathcal{B}$. Since we need to sort only once, the total time complexity is $\mathcal{O}(n^3)$.

In this section we show how to achieve a marginal improvement in the time-complexity of the algorithm for the MIBS problem on circular-arc graphs, which was presented in the previous section. A similar improvement was proved in [GLL82] for the maximum independent set algorithm, where they exploited a nice observation attributed to Gavril [Ga74]. Gavril had observed that every independent set leaves some elementary segment uncovered, and that, in fact, it leaves some critical segment uncovered. Thus by restricting the algorithm to look at all critical segments instead of all elementary segments, the worst-case time complexity of the maximum independent set algorithm presented in [GLL82] was $\mathcal{O}(kn+n\log n)$, where k is the number of critical segments in the circular representation of the graph.

We extend the observation by Gavril to the MIBS problem. In Section 2.3, we had argued that there must be at least one elementary segment a_i such that W_i, which is the set of intervals that cover this elementary segment, has at most one interval in a maximum bipartite subgraph of the graph. It is not hard to see that, in fact, there must be a critical segment satisfying the same property. Hence step 1 in algorithm $\mathcal{C}$ need not be performed for all i between 1 and $2n$. It only needs to be performed for those values of i for which a_i is a critical segment. Hence if there are k critical segments in the circular representation of the graph, the MIBS algorithm for circular-arc graphs has a worst-case time complexity of $\mathcal{O}(kn^2)$.

Yeh and LaPaugh [YL87] also have an algorithm for the maximum bipartite subgraph problem on circular-arc graphs. Their algorithm has a complexity of $\mathcal{O}(n\delta^2)$, where δ is the minimum number of edges covering any point on the circle.

2.5. Discussion

We conclude this chapter by commenting on the complexity of the MkCS problem on interval and circular-arc graphs. Yannakakis and Gavril [YG87] showed that, in fact, the greedy algorithm also finds the maximum k-colorable subgraph on interval graphs. Just like the algorithm that we presented for the MIBS problem on interval graphs, this algorithm also runs in linear time.

It remains an open problem to determine the complexity of the MkCS problem on the class of circular-arc graphs. It would be interesting to look at the slightly easier question of determining the complexity of the MkCS problem on circular-arc graphs when k is a fixed constant. In particular, it would be interesting to solve it for $k = 3$. This might provide insight into why the more general MkCS problem seems hard on this class of graphs, and why the algorithm for the MIBS problem on circular-arc graphs does not lend itself to an easy generalization to provide an algorithm for the MkCS problem on circular-arc graphs.

Chapter 3

Algorithms on Tolerance Graphs

3.1. Overview

Recently, Golumbic and Monma [GM82] introduced a new subclass of perfect graphs called tolerance graphs. The class of tolerance graphs generalizes the class of interval graphs by relaxing the notion of "intersection" of intervals that is used in the definition of interval graphs. As mentioned in the previous chapter, interval graphs have applications where the notion of overlap in time or space needs to be modeled. Like interval graphs, tolerance graphs have applications in scheduling problems where some amount of tolerance is allowed on the overlap in time or space. Specific examples can be found in [GMT84].

In this chapter we study some algorithmic aspects of the class of tolerance graphs. We present polynomial time algorithms to solve the maximum clique problem, and the maximum independent set problem on tolerance graphs. We briefly discuss how to generalize our algorithm to compute the maximum independent set (equivalently, the maximum 1-colorable subgraph) to solve the maximum bipartite (or 2-colorable) subgraph problem and the maximum k-colorable subgraph problem for constant k. We also present an approximation algorithm to find a coloring of a tolerance graph that uses at most one color more than the optimal number of colors needed to color the graph.

3.2. Tolerance Graphs

An undirected graph $G(V, E)$ is a *tolerance graph* if there exists a collection $\mathcal{I} = \{\overline{v} | v \in V\}$ of closed intervals on a line and a multiset $\mathcal{T} = \{t_v | v \in V\}$ such that

$$(x, y) \in E \Longleftrightarrow |\overline{x} \cap \overline{y}| \geq \min\{t_x, t_y\}.$$

Here $|\overline{x}|$ denotes the length of interval $\overline{x}$. The number t_v is called the *tolerance* of $\overline{v}$. We say that two intervals *conflict* if their intersection rises above a threshold equal to the minimum of the tolerances of the two intervals. Thus, a graph is a tolerance graph if there exists a pair $\langle \mathcal{I}, \mathcal{T} \rangle$ such that

$$(x, y) \in E \Longleftrightarrow \overline{x} \text{ and } \overline{y} \text{ conflict}$$

The pair $\langle \mathcal{I}, T \rangle$ is called a *tolerance representation* of G. For example, the simple cycle of length 4, C_4, is a tolerance graph. A tolerance representation for C_4 is given in Figure 3.1. We also note that C_4 is not an interval graph.

The class of tolerance graphs is similar to the class of interval graphs; the difference being that the notion of "intersection" of intervals is replaced by a notion of "conflict" of intervals. Golumbic and Monma [GM82] have shown that the class of tolerance graphs properly contains the classes of interval graphs and permutation graphs† [GM82]. They prove that if all the intervals in a tolerance representation have 0 tolerance, the resulting tolerance graph is also an interval graph. On the other hand, they also show that every permutation graph has a tolerance representation with the tolerances equal to the lengths of the intervals. It has also been shown that all tolerance graphs are perfect [GMT84].

An interval in a tolerance representation is *bounded* if its tolerance does not exceed its length, otherwise it is *unbounded*. A tolerance representation is *bounded* if all its intervals are bounded. A tolerance graph is a *bounded tolerance* graph if it admits a bounded tolerance representation. The tolerance representation for C_4 given in Figure 3.1, was not a bounded tolerance representation since the tolerance of interval $\overline{b}$ was 6, while its length was only 3. However, C_4 is a bounded tolerance graph since it admits a bounded tolerance representation as shown in Figure 3.2.

Golumbic and Monma [GM82] showed that every bounded tolerance graph is a co-comparability graph. Since we are going to make several references to co-comparability graphs, we define them here. A *co-comparability graph* is a graph whose complement is a comparability graph or a transitively orientable graph. A *comparability graph* or a *transitively orientable graph* is a graph whose edges can be assigned a direction in such a way that the resulting orientation is transitive. An orientation is transitive if it is true that whenever there is an oriented edge from x to y, and an oriented edge from y to z, there is an oriented edge from x to z. From the statements made earlier it is clear that interval graphs and permutation graphs are bounded tolerance graphs.

The algorithmic aspects of tolerance graphs have not been studied before. Since tolerance graphs are perfect, all the algorithms on perfect graphs would also work on tolerance graphs. In particular, there are polynomial time algorithms to compute the following four parameters [GLS81] :

- *Stability number* : size of the largest independent set,
- *Clique number* : size of the largest clique,
- *Chromatic number* : least number of independent sets to cover vertex set.
- *Clique Cover number* : fewest number of cliques to cover vertex set.

In fact, since for perfect graphs the chromatic number equals the clique number and the stability number equals the clique cover number, it suffices to compute only two of these parameters. The algorithms to compute these parameters for perfect graphs use the Ellipsoid method [GLS81] and hence are not very

† for definition see Appendix

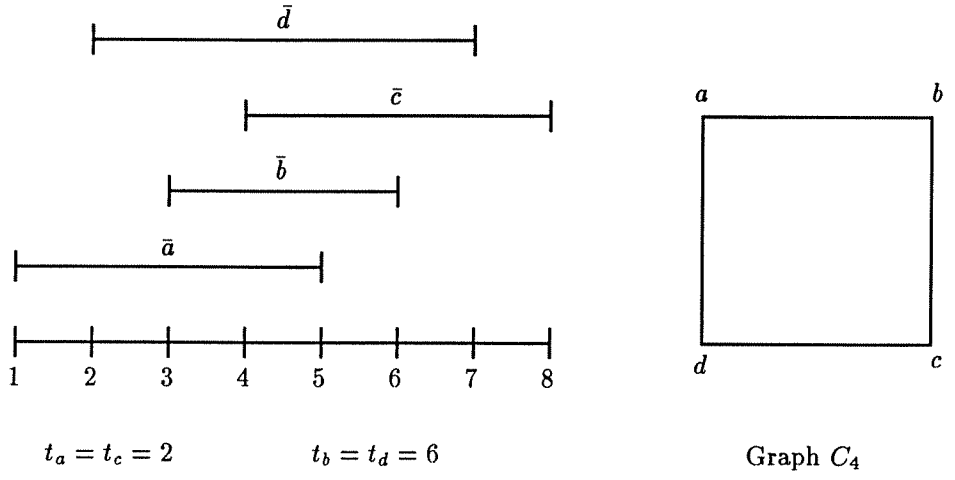


Figure 3.1: A tolerance graph C_4 and its tolerance representation

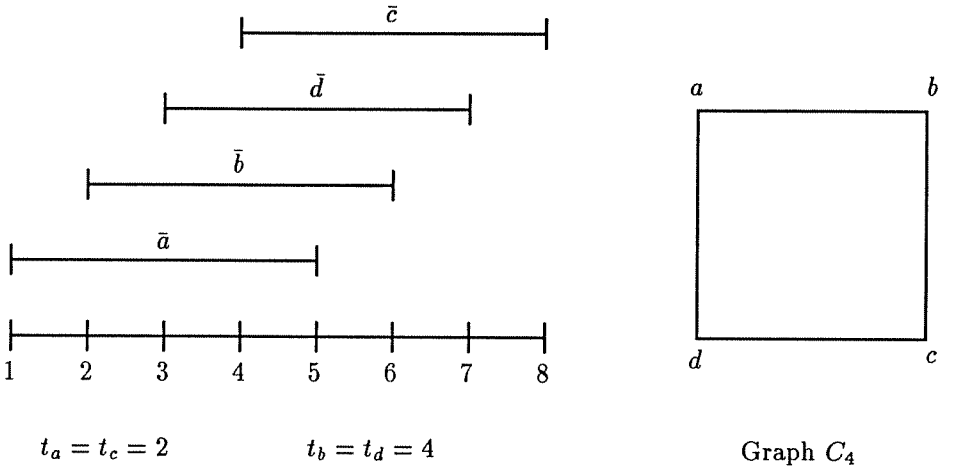


Figure 3.2: A bounded tolerance representation for graph C_4

efficient. For most known subclasses of perfect graphs there exist more efficient algorithms to determine the values of these parameters. Our aim is to design more efficient algorithms to compute the above parameters on tolerance graphs. Since bounded tolerance graphs are co-comparability graphs, all the known algorithms on co-comparability graphs will apply on the restricted subclass of bounded tolerance graphs. These include polynomial time algorithms for computing all the four parameters mentioned above. We present polynomial time algorithms to solve the maximum independent set problem and the maximum clique problem on general tolerance graphs.

Given a tolerance representation, its corresponding tolerance graph can be constructed in polynomial time. Determining the presence or absence of an edge between two vertices involves finding the length of the intersection of the two corresponding intervals and comparing it with the minimum of the two tolerance values, all of which can be performed in $\mathcal{O}(1)$ time. Since $\mathcal{O}(n^2)$ pairs of intervals need to be checked, the tolerance graph can be constructed from the tolerance representation in $\mathcal{O}(n^2)$ time. In contrast, the recognition problem for the class of tolerance graphs is yet unsolved. Even when the input graph is known to be a tolerance graph, it is not known how to obtain a tolerance representation for it. Moreover, given a tolerance graph it is not known how to decide in polynomial time whether it is a bounded tolerance graph. The tolerance representation for a tolerance graph is not unique, as can be seen from the example of the two representations for C_4 in figures 3.1 and 3.2. In view of these remarks, for all the algorithms presented in this chapter we assume that along with the input graph $G = (V, E)$, we are given a tolerance representation $\langle \mathcal{I}, \mathcal{T} \rangle$ of G. The interval corresponding to a vertex $v \in V$ is $\overline{v} = (l(v), r(v))$, where $l(v)$ and $r(v)$ represent the left and right end points of interval $\overline{v}$. Following [GMT84] we also assume that the tolerance representation has intervals such that each of the end points is distinct. Such a tolerance representation will be called a *regular* representation. In [GMT84], a simple way of making any tolerance representation regular is shown. We shall assume that for our algorithms the input consists of a tolerance graph and a regular tolerance representation for that graph.

In the next section we present a polynomial-time algorithm to compute a maximum independent set in a general tolerance graph.

3.3. Maximum Independent Set Algorithm

The intervals in a tolerance representation of a tolerance graph can be partitioned into two sets of intervals. One set $\overline{B} = \{\overline{b_1}, \ldots, \overline{b_p}\}$ consists of all bounded intervals, and the other set $\overline{U} = \{\overline{u_1}, \ldots, \overline{u_q}\}$ consists of all the unbounded intervals. Without loss of generality, we assume that $r(b_1) < r(b_2) < \ldots < r(b_p)$, and $r(u_1) < r(u_2) < \ldots < r(u_q)$. This partition induces a partition of the vertices into the two sets of vertices $B = \{b_1, \ldots, b_p\}$ and $U = \{u_1, \ldots, u_q\}$. We refer to vertices in B as *bounded vertices* and to vertices of U as *unbounded vertices.* Let the subgraphs induced by these sets be represented by G_B and

G_U respectively. As mentioned earlier G_B is a co-comparability graph. Since no two unbounded intervals intersect, G_U is an independent set. Our algorithm to find the maximum independent set in a tolerance graph G transforms the bounded part of G into a weighted directed graph whose weights are a function of the unbounded part of G.

Theorem 1: Given a tolerance graph $G = (V, E)$ and a regular tolerance representation $\langle \mathcal{I}, \mathcal{T} \rangle$ of G, there is an $\mathcal{O}(|B|^2 \log |U|)$ algorithm to find the largest independent set in G, where $|B|$ and $|U|$ are the numbers of bounded and unbounded vertices in G.

We first describe the algorithm to compute the stability number of a co-comparability graph. Then we explain why a bounded tolerance graph is a co-comparability graph. This provides us with a maximum independent set algorithm for G_B. After that we show how to extend the algorithm to develop an algorithm for the maximum independent set problem on general tolerance graphs.

The algorithm to compute the stability number of a co-comparability graph G computes the clique number of its transitively orientable complement $\overline{G}$. The computation of the clique number of a comparability graph is based on the fact that a clique in a comparability graph corresponds to a directed path in its transitive orientation [Go80]. Consequently, a maximum clique in a comparability graph corresponds to the longest path in its transitive orientation. Although the longest path problem is in general NP-complete, it can be computed in linear time for a digraph obtained as a transitive orientation of a comparability graph since this digraph is acyclic [Go80]. A transitive orientation of a comparability graph can be computed in $\mathcal{O}(\Delta|E|)$, where Δ is the maximum degree of a vertex in G [Go80]. Thus, the time complexity to determine the stability number of a co-comparability graph is $\mathcal{O}(|V|^2 + \Delta|E|)$. In fact, the algorithm can actually find the largest independent set in the given co-comparability graph since we can easily recover the list of vertices along the longest path in the transitive orientation of its complement.

Now we explain why a bounded tolerance graph G is a co-comparability graph, and how we can use the bounded tolerance representation of G to transitively orient the complement of G in linear time. Following [GMT84], we define the *right end point orientation* of $\overline{G}$ as follows. An edge (x, y) in $\overline{G}$ is oriented from vertex x to vertex y if in the given tolerance representation of G, interval $\overline{x}$ terminates before interval $\overline{y}$. It is not hard to see that a right end point orientation of a bounded tolerance graph is transitive [GMT84]. Hence transitive orientations of bounded tolerance graphs can be obtained faster than for general co-comparability graphs. This implies that the maximum independent set in a bounded tolerance graph can be found in $\mathcal{O}(|V|^2)$.

We extend the maximum independent set algorithm on bounded tolerance graphs to present a maximum independent set algorithm on general tolerance graphs. We reduce the problem of finding the maximum independent set in a tolerance graph G to that of finding the longest (heaviest) path in an acyclic weighted directed graph $H(G)$. The digraph $H(G)$ consists of the right end point orientation of the complement of the subgraph induced by the bounded vertices G_B, together with two additional vertices, a source vertex s and

a sink vertex t. Thus every bounded vertex is represented by a vertex in $H(G)$. The source is joined to all the vertices in $B \cup \{t\}$ and the edges are oriented from s. The sink is joined to all the vertices of B and the edges are oriented to t. One may think of s (resp. t) as representing an interval that starts and terminates before (resp. after) all the intervals in $\mathcal{I}$ and whose tolerance equals its length. It is convenient to think of $H(G)$ as the simply the right end point orientation of the complement of $G_{B'}$, where B' is obtained from B by adding two vertices s and t, or equivalently, by adding to $\overline{B}$ two intervals $\overline{s}$ and $\overline{t}$ with 0 tolerances such that $\overline{s}$ (resp. $\overline{t}$) starts and terminates before (resp. after) all intervals in $\mathcal{I}$. In other words, the vertex set of $H(G)$ is $B \cup \{s,t\}$ and there is a directed edge from x to y if x and y are not adjacent in G', and interval $\overline{x}$ terminates before interval $\overline{y}$.

We associate a set-valued function $S(e)$ and a weight function $w(e)$ with each directed edge $e = (x,y)$ in $H(G)$. The set $S(e)$ consists of all unbounded vertices $u_k \in G$ that are not adjacent either to x or to y and whose corresponding unbounded intervals $\overline{u_k}$ terminate after $r(x)$ and before $r(y)$. Note that for a directed edge $e = (s,b)$ in $H(G)$, $S(e)$ consists of those unbounded vertices $u_k \in G$ that are not adjacent to b and whose corresponding unbounded intervals $\overline{u_k}$ terminate before $\overline{b}$ does. A similar statement holds for edges directed toward the sink t, with the word "after" replacing the word "before". Note that $S((s,t)) = U$. Hence each set $S(e)$ is an independent set. The weight function w is defined as follows:

$$w(e) = \begin{cases} |S(e)| & \text{if } e = (b,t); \\ |S(e)| + 1 & \text{otherwise.} \end{cases}$$

The motivation for the definition of the weight function will become apparent later.

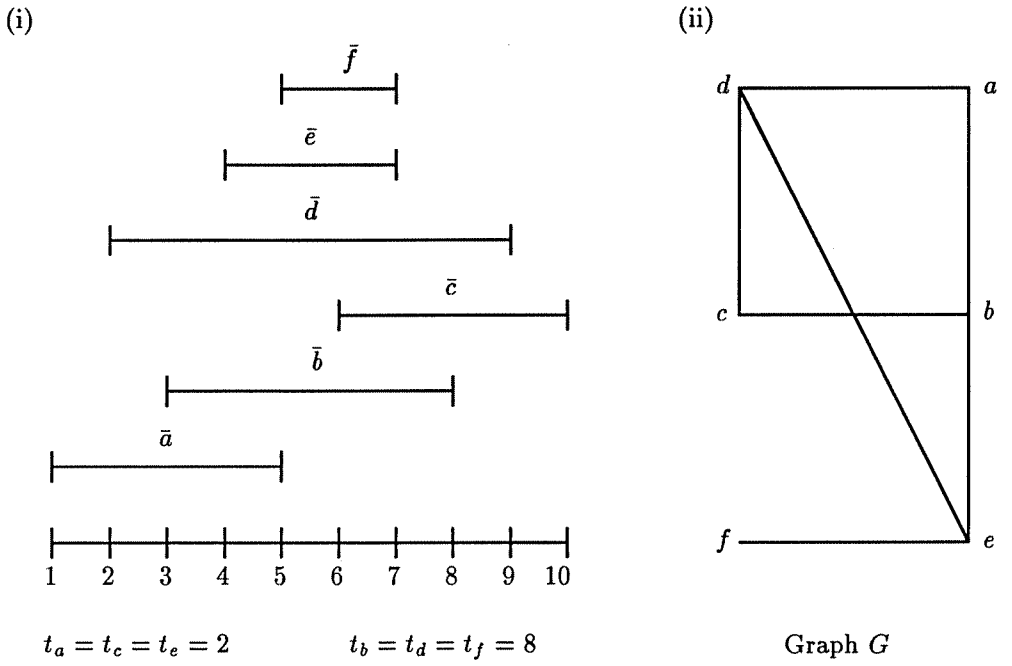
An example of the construction of $H(G)$ for a given tolerance graph G is illustrated in Figure 3.3 in which

$$S((s,c)) = S((a,c)) = S((a,t)) = \{f\}, \quad S((s,t)) = \{b,d,f\},$$

and for all other edges $S(e) = \emptyset$.

Given below is an informal description of the algorithm to compute the maximum independent set in a tolerance graph.

We prove that the above algorithm correctly computes the maximum independent set in a tolerance graph. Our proof proceeds as follows. We show that there is a one-to-one correspondence between independent sets in G and s-t paths in H, and then we argue that the longest path in H must correspond to a maximum independent set in G. Let $P = \{e_0 = (s, b_{i_1}), e_1 = (b_{i_1}, b_{i_2}), \ldots, e_k = (b_{i_k}, t)\}$ be any s-t path in H. By the construction of H, we have $1 \leq i_1 < \ldots < i_k \leq n'$. We first note that each of the sets $S(e_i), 0 \leq i \leq k$, is a subset of U. Since unbounded intervals do not intersect each other, U forms an independent set in G. Thus $\cup_{i=0}^{k} S(e_i)$ forms an independent set in G. In Lemma 2 we show that the set $B_P = \{b_{i_1}, \ldots, b_{i_k}\}$ forms an independent set in G. Then in Lemma 3, we show that each of the sets $S(e_i) \cup B_P$ is independent for $0 \leq i \leq k$. We can conclude from Lemmas 2 and 3 that $\bigcup_{i=0}^{k} S(e_i) \cup B_P$ forms an independent set in G.



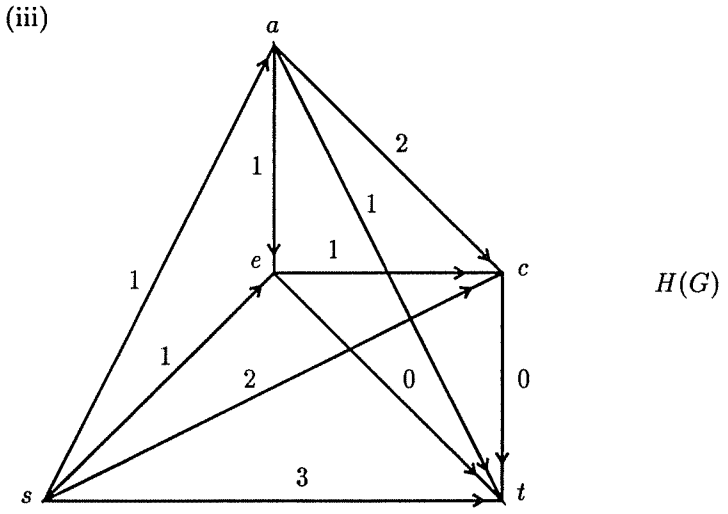


Figure 3.3: (ii) A tolerance graph G.
(i) Its tolerance representation, and
(iii) the constructed directed graph $H(G)$.

Algorithm

Given a tolerance graph $G(V, E)$, and its tolerance representation $\langle \mathcal{I}, \mathcal{T} \rangle$, construct a weighted directed graph H as described above. Find the longest weighted path in H from s to t. Let the longest path be $P = \{e_0 = (s, b_{i_1}), e_1 = (b_{i_1}, b_{i_2}), \ldots, e_k = (b_{i_k}, t)\}$. Let $B_P = \{b_{i_1}, b_{i_2}, \ldots, b_{i_k}\}$ be the set of *internal vertices* on P. Then the maximum independent set in the tolerance graph G is

$$S = \bigcup_{i=0}^{k} S(e_i) \cup \{b_{i_1}, b_{i_2}, \ldots, b_{i_k}\}.$$

Figure 3.4: Maximum independent set algorithm on tolerance graphs

It is also clear from the definition that the sets $S(e_i), 0 \leq i \leq k$, are disjoint. Hence $\bigcup_{i=0}^{k} S(e_i) \cup B_P$ is an independent set in G of cardinality $k + \sum_{i=0}^{k} |S(e_i)|$. By similar arguments, an independent set in G corresponds to an s-t path in H. Thus if P is the longest path from s to t in H, then $\bigcup_{i=0}^{k} S(e_i) \cup B_P$ is a maximum independent set in G.

Lemma 2: $B_P = \{b_{i_1}, \ldots, b_{i_k}\}$ forms an independent set in G.

Proof: The edges of H are oriented using the right end point orientation. We shall show that this results in a transitive orientation of the edges in H. That is, we shall show that if H has edges $e = (b_i, b_j)$ and $e' = (b_j, b_k)$, then it also has an edge $e'' = (b_i, b_k)$. Since an edge $e = (b_i, b_j)$ in H implies that there is no edge between b_i and b_j in G, the transitivity of edges in H is sufficient to prove that the set of vertices in a path in H corresponds to an independent set in G.

If $e = (b_i, b_j)$ is an edge in H, then the intervals $\overline{b_i}$ and $\overline{b_j}$ do not conflict. Similarly, existence of edge e' implies that intervals $\overline{b_j}$ and $\overline{b_k}$ do not conflict. We will show that if intervals $\overline{b_i}$ and $\overline{b_j}$ do not conflict, and intervals $\overline{b_j}$ and $\overline{b_k}$ do not conflict, then $\overline{b_i}$ and $\overline{b_k}$ are also non-conflicting. It is clear from the direction of edges e and e' that $i < j < k$, or equivalently, $r(b_i) < r(b_j) < r(b_k)$. Also, if two bounded intervals do not conflict, then neither one can completely contain the other, since otherwise, the amount of the intersection of the intervals is equal to the length of the smaller interval, which in turn is at least as large as the tolerance of the smaller interval. Hence intervals $\overline{b_i}$ and $\overline{b_j}$ do not contain each other. Similarly, intervals $\overline{b_j}$ and $\overline{b_k}$ do not contain each other. Hence, this implies that the left end points of intervals $\overline{b_i}$, $\overline{b_j}$, and $\overline{b_k}$ are also in increasing order, that is, $l(b_i) < l(b_j) < l(b_k)$. By the construction of $H(G)$, we know that $|\overline{b_i} \cap \overline{b_j}| < t_i$, and $|\overline{b_j} \cap \overline{b_k}| < t_k$. Based on the ordering of the right end points and the left end points of $\overline{b_i}$, $\overline{b_j}$, and $\overline{b_k}$, we can deduce that $|\overline{b_i} \cap \overline{b_k}| < |\overline{b_i} \cap \overline{b_j}|$, and $|\overline{b_i} \cap \overline{b_k}| < |\overline{b_j} \cap \overline{b_k}|$. It follows that $|\overline{b_i} \cap \overline{b_k}| < \min\{|\overline{b_i} \cap \overline{b_j}|, |\overline{b_j} \cap \overline{b_k}|\} < \min\{t_i, t_k\}$. Hence intervals $\overline{b_i}$ and $\overline{b_k}$ do not conflict. Thus there must be an edge $e'' = (b_i, b_k)$ in H, proving that the edges in H are transitive. This implies that since the vertices in B_P lie on a path in H, B_P forms an independent set in G, thus proving Lemma 2. ∎

The next lemma shows that we can extend the independent set B_P, which consists only of bounded vertices, to include unbounded vertices.

Lemma 3: Let $P = \{e_0 = (s, b_{i_1}), e_1 = (b_{i_1}, b_{i_2}), \ldots, e_k = (b_{i_k}, t)\}$ be any directed path from s to t in $H(G)$. Then $S(e_j) \cup B_P$ is an independent set in G, $1 \leq j \leq k$.

Proof: The sets $S(e_j)$ and B_P are each independent sets in G. So we only need to show that there is no edge with one end point in $S(e_j)$ and the other endpoint in B_P. That is, it suffices to show that if $u \in S(e_j)$, for some $0 \leq j \leq k$, and $b_{i_l} \in B_P$ for some $1 \leq l \leq k$, then the vertices u and b_{i_l} are not adjacent in G. In other words, we need to show that the intervals $\overline{u}$ and $\overline{b_{i_l}}$ do not conflict.

Assume first that $1 \leq l < j < k$. If an unbounded interval and a bounded interval do not conflict, then the unbounded interval cannot completely contain the bounded interval. Hence interval $\overline{u}$ cannot completely contain interval $\overline{b_{i_j}}$. Using the arguments from Lemma 2, we know that neither $\overline{b_{i_j}}$ nor $\overline{b_{i_l}}$ can completely contain the other. Thus the left end points and the right end points of intervals $\overline{b_{i_l}}$, $\overline{b_{i_j}}$, and $\overline{u}$ are ordered as follows: $l(b_{i_l}) < l(b_{i_j}) < l(u)$, and $r(b_{i_l}) < r(b_{i_j}) < r(u)$. Hence $|\overline{u} \cap \overline{b_{i_l}}| < |\overline{b_{i_l}} \cap \overline{b_{i_j}}|$. Furthermore, the intervals $\overline{b_{i_j}}$ and $\overline{b_{i_l}}$ do not conflict and hence $|\overline{b_{i_l}} \cap \overline{b_{i_j}}| < \min\{t_{b_{i_l}}, t_{b_{i_j}}\} \leq t_{b_{i_l}} = \min\{t_{b_{i_l}}, t_u\}$. It follows that $\overline{u}$ and $\overline{b_{i_l}}$ do not conflict and hence u and b_{i_l} are not adjacent in G.

The case $j = k$ follows verbatim if we let $t = b_{i_{k+1}}$. The arguments hold if we assume that the vertex t represents a bounded interval with tolerance 0, and which starts (and terminates) after all the other intervals in the input. A similar argument can be used to handle the case $1 \leq j + 1 < l \leq k$. Finally, if $l = j$ or if $l = j + 1$, then by the definition of $S(e_j)$, $(u, b_{i_l}) \notin E(G)$. ∎

Lemma 3 implies that an independent set in G_B consisting of internal vertices of a path P from s to t in $H(G)$, can be extended to an independent set $I_P = B_P \cup S(e_0) \cup S(e_1) \ldots \cup S(e_k)$ in G. In the example of Figure 3.3 there are three paths of total weight 3, $P_1 = ((s,t))$, $P_2 = ((s,a),(a,c),(c,t))$, and $P_3 = ((s,a),(a,e),(e,c),(c,t))$. Their corresponding independent sets are $I_{P_1} = \{b, d, f\}$, $I_{P_2} = \{a, c, f\}$, and $I_{P_3} = \{a, c, e\}$. Thus, selecting edge $e = (b_i, b_j)$, $1 \leq j \leq k$ to be included in a path P from s to t in $H(G)$ is equivalent to selecting $S(e) \cup \{b_j\}$ to be included in the independent set I_P. Since the sets $S(e_j)$ are disjoint, this means that each edge in the path, except the last edge, identifies $|S(e)| + 1 = w(e)$ vertices in the corresponding independent set. The last edge identifies only $|S(e)| = w(e)$ vertices. In other words, each path from s to t in $H(G)$ corresponds to an independent set in G whose size is the sum of the weights of the edges in the path. This observation is the basis for our algorithm to find an independent set of maximum size in a general tolerance graph.

Proof of Theorem 1: We first construct the weighted directed graph $H(G)$. Let P be the longest weighted path from s to t in $H(G)$. Denote s by b_{i_0} and denote t by $b_{i_{k+1}}$. We claim that the largest independent set in G is

$$S = \bigcup_{i=0}^{k} S(e_i) \cup B_P.$$

Lemma 2 implies that S is an independent set. In order to show that S is an independent set of maximal cardinality, it suffices to show that every independent set S in G is equal to $\bigcup_{i=0}^{k} S(e_i) \cup B_P$ for some path P in $H(G)$. So let S be an independent set in G. Let $S_B = \{b_{i_1}, b_{i_2}, \ldots, b_{i_k}\}$ be an ordered set consisting of all the bounded vertices in S ordered by the right end points of the corresponding intervals. Let S_U be the set of all unbounded intervals in S. If S_B is empty then P consists of the single edge (s,t) and $S = S_U = U$. Otherwise, the independent set S_B in G induces a clique in $\overline{G}$. Moreover, the directed edges, $e_j = (b_{i_j}, b_{i_{j+1}})$, joining consecutive vertices in S_B form a directed path P from s to t in the right end point orientation of $\overline{G}$. Hence P is a path in $H(G)$. We can now partition S_U into $k+1$ subsets $S(e_j)$ for $0 \leq j \leq k$. In this partition a vertex $u \in S_U$ belongs to $S(e_j)$ if $r(b_{i_j}) < r(u) < r(b_{i_{j+1}})$. Hence the algorithm described above correctly computes the maximum independent set in a general tolerance graph.

The construction of the unweighted right end point orientation of $H(G)$ can be done in $\mathcal{O}(|B^2|)$. The weight function w can be computed in time $\mathcal{O}(|B|^2 \log |U|)$, assuming, as we did, that U is sorted. If U is empty then this stage can be done in constant time. The longest weighted path in $H(G)$ can be found in time linear in the size of $H(G)$ [Go80]. The size of $H(G)$ is $\mathcal{O}(|B|^2 + \log |w|) = \mathcal{O}(|B|^2 + \log |U|)$, where $|w|$ is the largest weight in $H(G)$. All these steps combined yield a total time complexity of $\mathcal{O}(|B|^2 \log |U|)$. ∎

Note that when the input graph is a bounded tolerance graph, and consequently a co-comparability graph, all the weights in $H(G)$ are 1 except for edges joined to the sink. In this case our algorithm reduces to the algorithm for co-comparability graphs, which finds a longest path in an unweighted digraph,
The next corollary follows from the fact that tolerance graphs are perfect.

Corollary 4: Given a tolerance graph $G = (V, E)$ and a regular tolerance representation $\langle \mathcal{I}, T \rangle$ of G, there is an $\mathcal{O}(|V|^2)$ algorithm to find the clique cover number of G.

3.4. Maximum Clique Algorithm

Let G_B be the subgraph of G induced by the vertices in B. Since the vertices in U form an independent set, any clique in G can have at most one vertex from U. If there are no vertices from U in some maximum clique, then the largest clique in G is the same as the largest clique in G_B. Then G_B is a co-comparability graph [GM82], and Andras Frank [F80] showed an efficient algorithm to find the maximum clique in a co-comparability graph. Since we know how to handle the case when there are no vertices from U, let us assume that every maximum clique contains some vertex from U and that vertex $u_i \in U$ is in a maximum clique of G. Then there exists a maximum clique consisting of vertex u_i and a set of vertices $C_i \subseteq B$ lying in the neighborhood of vertex u_i. If we denote the neighborhood of u_i by $N(u_i)$, then it is easy to see that the vertices in C_i form a maximum clique in the subgraph of G induced by the vertices in $N(u_i) \cap B$. The

Algorithm

Let the largest clique in G_B be C'_0. For each $u_i \in U$, let C_i be the largest clique in the subgraph induced by $N(u_i) \cap B$. Let $C'_i = C_i \cup \{u_i\}$, $1 \leq i \leq n''$. Then the maximum clique in G is the largest of the cliques $C'_0, C'_1, \ldots, C'_{n''}$.

Figure 3.5: Maximum clique algorithm on tolerance graphs

subgraph induced by vertices in $N(u_i) \cap B$ is also a co-comparability graph. Hence C_i can be determined easily by using Frank's algorithm [F80].

The arguments given before the algorithm provide sufficient proof of correctness of the above algorithm.

3.5. Approximate Coloring Algorithm

Since tolerance graphs are perfect†, their chromatic number can be determined in polynomial time using the ellipsoid algorithm [GLS81]. The techniques from [GLS81] can also be used to find the optimal coloring of tolerance graphs. What we present here is a much more efficient algorithm that colors any tolerance graph using at most one color more than the optimal number of colors.

There are efficient algorithms to color co-comparability graphs [F80]. Since G_B is a co-comparability graph, there is an efficient algorithm to color G_B optimally. We also know that the set of vertices in U form an independent set and hence can be colored with one additional color. Since the chromatic number of G cannot be less than the chromatic number of G_B, it is clear that we could not have used more than $\chi(G)+1$ colors, where $\chi(G)$ is the chromatic number of G.

It remains open to find an efficient algorithm to find an optimal coloring of a tolerance graph.

3.6. Algorithm to find the Maximum Induced Bipartite Subgraph

We present here a very brief sketch of the algorithm. The details will not be described in this report. The algorithm is similar to the algorithm for finding the maximum independent set. Given a tolerance graph G with n vertices, the algorithm constructs a weighted directed graph $H(G)$ with $O(n^2)$ vertices, and then finds the longest path in it.

As before, we assume that G has bounded vertices $B = \{b_1, \ldots, b_p\}$ and unbounded vertices $U = \{u_1, \ldots, u_q\}$, and that the right end points are ordered. That is, $r(b_1) < r(b_2) < \cdots < r(b_p)$, and $r(u_1) <$

† for definition see Appendix

$r(u_2) < \cdots < r(u_q)$. The vertices of H consist of ordered pairs (b_i, b_j), for $1 \leq i, j \leq p$. Graph H has directed edges (b_i, b_j) to (b_i, b_k) if $j < k$ and b_j and b_k do not conflict. Graph H also has directed edges (b_i, b_j) to (b_k, b_j) if $i < k$ and b_i and b_k do not conflict. We also add a source vertex s, and a sink vertex t, and join it to all the other vertices. The point of this construction is that a path from s to t in H corresponds to a bipartite subgraph of the bounded vertices of G. Hence a path P in H of the form $P = \{s, (b_{i_1}, b_{j_1}), \ldots, (b_{i_k}, b_{i_l}), t\}$ corresponds to a bipartite subgraph with two independent sets $\{b_{i_1}, \ldots, b_{i_k}\}$ and $\{b_{j_1}, \ldots, b_{j_l}\}$. Both these independent sets only contain bounded vertices. The unbounded intervals are accounted for by assigning appropriate weights to the edges. So what remains is to specify the weights on the edges connecting these vertices. The weight of an edge (b_i, b_j) to (b_k, b_j) will be assigned in such a way as to reflect the number of unbounded intervals that can be added to the independent set that includes b_i and b_k, and which terminates between the right end points of b_i and b_k. There are a number of special cases to consider in assigning these weights, which we do not describe here.

It is worth mentioning that the above algorithm is a reasonably straightforward generalization of the algorithm for finding the maximum independent set in a tolerance graph. Moreover, the algorithm can also be generalized to find the maximum $k-$colorable subgraph in a tolerance graph. As might be expected this involves finding the longest path in a weighted directed graph with $O(n^k)$ vertices.

Chapter 4

Generalization of Lovász's Sandwich Theorem

4.1. Introduction

In this chapter we present new inequalities that help to find bounds on the size of the maximum k-colorable subgraph. We prove that these bounds can be computed in polynomial time. In the process, we generalize some results due to Wilf [Wil67], and the sandwich theorem due to Lovász [Lov86]. The techniques used to prove these inequalities are algebraic in nature.

As before, given a graph $G = (V, E)$ with n vertices, we denote the size of the largest clique by $\omega(G)$, the size of the largest independent set by $\alpha(G)$, and the chromatic number by $\chi(G)$. Let $\alpha_k(G)$ be the size of the maximum induced k-partite subgraph in G. Our inequalities find bounds for this graph parameter $\alpha_k(G)$. Let $\omega_k(G)$ be the size of the largest induced subgraph that can be covered by k cliques. An independent set in a graph corresponds to a clique in the complement graph (denoted by $\overline{G}$). Hence it is easy to see that $\omega(\overline{G}) = \alpha(G)$, and that $\omega_k(\overline{G}) = \alpha_k(G)$. Thus, finding inequalities to bound $\alpha_k(G)$ would also provide inequalities to bound $\omega_k(G)$.

Just as $\alpha_k(G)$ and $\omega_k(G)$ generalize $\alpha(G)$ and $\omega(G)$, we also define $\chi_k(G)$, or the k-multichromatic number, which generalizes the chromatic number $\chi(G)$. We define the k-multichromatic number $\chi_k(G)$ to be the minimum number of colors needed for a k-multicoloring of the graph G. A k-multicoloring of a graph is obtained by assigning each vertex a set of k colors in such a way that any two adjacent vertices are assigned disjoint sets of colors.

For general graphs, it is NP-hard to determine any of these six graph invariants mentioned above. However, as pointed out earlier, these parameters need not be hard to compute when the inputs are restricted to special classes of graphs. Lovász [Lov86] devised an interesting technique to demonstrate a polynomial-time algorithm to compute $\omega(G)$ and $\chi(G)$ (and also $\alpha(G)$) for a perfect graph. A *perfect graph* is a graph for which the size of the largest clique and the chromatic number are equal for every induced subgraph. Lovász's *Sandwich Theorem* [Lov86] demonstrated a polynomial-time computable function $\vartheta(G)$, whose value always lies between $\omega(G)$ and $\chi(G)$. Since, by definition, $\omega(G) = \chi(G)$ for perfect graphs, the sandwich theorem gives a polynomial-time algorithm for computing $\omega(G)$ and $\chi(G)$ for perfect graphs [GLS81]. Lovász's theorem also provides a new bound for the graph parameters $\omega(G)$ and $\chi(G)$, both of which are hard to

compute. The inequality presented in Lovász's theorem becomes all the more important because equality is attained in a number of interesting cases.

Our study was aimed at extending Lovász's results to obtain bounds for $\omega_k(G)$ (and consequently for $\alpha_k(G)$). Here we present a generalization of Lovász's sandwich theorem by demonstrating a polynomial-time computable function $\vartheta_k(G)$ that satisfies the following inequality:

$$\omega_k(G) \leq \vartheta_k(G) \leq \chi_k(G), \quad \text{for } 1 \leq k \leq n.$$

Hence the theorem provides a polynomial-time computable bound for both $\omega_k(G)$ and $\chi_k(G)$.

In his monograph [Lov86], Lovász states that it would be interesting to find other sandwich theorems. In other words, it would be interesting to have two NP-hard graph invariants that sandwich a graph invariant polynomial-time computable function. The motivation behind finding sandwich theorems is that this technique could help to efficiently compute interesting graph parameters for some special classes of graphs. They could also be useful in finding good upper or lower bounds for certain graph parameters, which in turn could help in designing heuristic or approximation algorithms.

Section 4.2 contains the first and essential step in proving that $\vartheta_k(G) \geq \omega_k(G)$. It provides a generalization of Wilf's Theorem [Wil67], which relates the largest eigenvalue of $A(G)$ to $\chi(G)$ and $\omega(G)$. Our sandwich theorem is proved in Section 4.3. Section 4.3 also includes the definition of $\chi_k(G)$. Finally, in Section 4.4 we discuss the special case of perfect graphs and k-perfect graphs and discuss some open problems.

4.2. A Generalization of Wilf's Inequality

We define the *adjacency matrix* $A(G) = (a_{ij})$ of G as follows: $a_{ij} = 1$ if either $i = j$, or i and j are adjacent. Otherwise $a_{ij} = 0$. In this chapter, we will study algebraic properties of the adjacency matrix of the graph and relate it to some interesting graph parameters. In particular, we will look at the eigenvalues of the adjacency matrix of the graph, as well as the eigenvalues of matrices derived from the adjacency matrix, and relate them to cliques and colorings of graphs. A square matrix B of order n has an *eigenvalue* λ, if λ is a solution of the equation:

$$B\overline{e} = \lambda\overline{e},$$

for some $n \times 1$ vector $\overline{e}$. If the matrix B is of order n, then it has n eigenvalues. These eigenvalues need not be distinct, and may be real or complex. The multiset of eigenvalues of the adjacency matrix of a graph is called its *spectrum*.

The adjacency matrix of a graph is a square, real, and symmetric matrix. In fact, all the matrices we will be dealing with in this chapter will be square, real, and symmetric. This implies that all the eigenvalues of these matrices will be real [CDS80]. Note that the adjacency matrix as defined above has 1's along

the diagonal instead of 0's as defined in [CDS80]. Consequently, the eigenvalues of the adjacency matrix as defined above are shifted by 1 from the eigenvalues of the adjacency matrix defined with 0's along the diagonal. Throughout this chapter λ_i denotes the i^{th} largest eigenvalue of the matrix in question.

Wilf [Wil67] showed that the largest eigenvalue of the adjacency matrix of a graph (with 1's along the diagonal), is at least as large as the chromatic number of the graph, which in turn is at least as large as the size of the largest clique in the graph. Our first theorem generalizes this result. We show that for any k between 1 and n, the sum of the k largest eigenvalues of the adjacency matrix is at least as large as the size of the largest induced subgraph of G that can be covered by k cliques.

Theorem 1: Let $G(V, E)$ be a graph with adjacency matrix A. Then

$$\sum_{i=1}^{k} \lambda_i(A) \geq \omega_k(G), \quad \forall\, k \leq n,$$

where $\lambda_i(A)$ is the i^{th} largest eigenvalue of A and $\omega_k(G)$ is the size of the largest induced subgraph of G that can be covered by k cliques.

We will need some lemmas before we prove Theorem 1. Lemma 2 is an elementary result from linear algebra, which we state here without proof. Note that the *trace* of a matrix is the sum of the entries along its diagonal.

Lemma 2: [ND77] For any square matrix A of order n with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$,

$$trace(A) = \sum_{i=1}^{n} a_{ii} = \sum_{i=1}^{n} \lambda_i(A).$$

Lemma 3: For any real, symmetric matrix A of order n with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ (in non-increasing order), $\sum_{i=1}^{k} a_{ii} \leq \sum_{i=1}^{k} \lambda_i(A)$, $\forall\, k \leq n$.

Proof: Let $A^{(k)}$ be the principal submatrix of A with rows and columns $1, 2, \ldots, k$. Let the eigenvalues of $A^{(k)}$ be $\lambda_1^{(k)} \geq \lambda_2^{(k)} \geq \ldots \geq \lambda_k^{(k)}$. By the interlacing theorem (see, for example, [MM64], p. 119), we have

$\lambda_1^{(k+1)} \geq \lambda_1^{(k)} \geq \lambda_2^{(k+1)} \geq \ldots \geq \lambda_k^{(k)} \geq \lambda_{k+1}^{(k+1)}$. Hence,

$$
\begin{aligned}
trace(A^{(k)} = \sum_{i=1}^{k} a_{ii} &= \sum_{i=1}^{k} \lambda_i^{(k)} \quad \text{(from Lemma 2)} \\
&\leq \sum_{i=1}^{k} \lambda_i^{(k+1)} \\
&\vdots \\
&\leq \sum_{i=1}^{k} \lambda_i^{(n)} \\
&\leq \sum_{i=1}^{k} \lambda_i(A)
\end{aligned}
$$

Hence the lemma is proved. ∎

Note that since simultaneous permutation of the rows and columns leaves the spectrum (or the set of eigenvalues) unchanged, $\sum_{i=1}^{k} \lambda_i(A)$ is at least as large as the sum of the k largest diagonal elements of A.

The following lemma is an elementary result in linear algebra (see, for example, [ND77]). A *non-singular* matrix is a matrix whose determinant is non-zero. A matrix B is an *orthogonal* matrix if $B^T B = I$, where I is the identity matrix. We denote a matrix with $\lambda_1, \lambda_2, \ldots, \lambda_n$ along the diagonal and 0's everywhere else by the notation $\text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$.

Lemma 4: [ND77] For a real, symmetric matrix A there exists a non-singular, orthogonal matrix P, such that $P^T AP = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) = D$, where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of A. Consequently, the spectra of A and $P^T AP$ are identical.

Proof of Theorem 1: Assume that the graph G has an induced subgraph H that can be covered by k cliques. Let $C_1, C_2, \ldots, C_k$ be any set of k disjoint cliques that cover H. Denote their sizes by $c_1 \geq c_2 \geq \ldots \geq c_k$. Let A_H be the adjacency matrix of the induced subgraph H. Let A' be the adjacency matrix of a graph G' composed of k disjoint cliques of sizes $c_1, c_2, \ldots, c_k$. Assume that both A_H and A' are matrices of order n'. Thus $c_1 + c_2 + \ldots + c_k = n'$. Then,

$$
A' = \begin{pmatrix}
J_{c_1} & 0 & \ldots & 0 \\
0 & J_{c_2} & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & J_{c_k}
\end{pmatrix}
$$

where J_i is a matrix of all 1's of order i. Furthermore, after relabeling the vertices of G, we may assume that

$$
A_H = \begin{pmatrix}
J_{c_1} & A_{12} & \ldots & A_{1k} \\
A_{21} & J_{c_2} & \ldots & A_{2k} \\
\vdots & \vdots & \ddots & \vdots \\
A_{k1} & A_{k2} & \ldots & J_{c_k}
\end{pmatrix}
$$

By Lemma 4, there exist matrices P_i of order c_i that diagonalize J_{c_i} for $i = 1, \ldots, k$. More precisely, $P_i^T J_{c_i} P_i = \text{diag}(c_i, 0, \ldots, 0)$. Let

$$P = \begin{pmatrix} P_1 & 0 & \ldots & 0 \\ 0 & P_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & P_k \end{pmatrix}$$

Then it follows that

$$P^T A' P = \text{diag}(\lambda_1(A'), \ldots, \lambda_k(A'), 0, \ldots, 0),$$

with $\lambda_i(A') = c_i$ for $i = 1, \ldots, k$. It is also clear that $P^T A' P$ and $P^T A_H P$ have the same elements along the diagonal. Hence

$$\begin{aligned} \sum_{i=1}^{k} \lambda_i(A_H) &= \sum_{i=1}^{k} \lambda_i(P^T A_H P) \\ &\geq \text{sum of the } k \text{ largest diagonal elements of } P^T A_H P \\ &= \text{sum of the } k \text{ largest diagonal elements of } P^T A' P = \sum_{i=1}^{k} c_i. \end{aligned}$$

It follows from the interlacing theorem that since A_H is a principal submatrix of A, $\lambda_i(A) \geq \lambda_i(A_H)$ for $i = 1, \ldots, k$. Hence the theorem follows. ∎

We illustrate Theorem 1 with a simple example. Consider a simple cycle with 5 vertices, denoted by C_5. The adjacency matrix of this graph is shown below:

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

The spectrum, or the multiset of eigenvalues of this matrix is

$$\{3.0, 1.618, 1.618, -0.618, -0.618\}.$$

Hence $\lambda_1(A) = 3$, $\lambda_2(A) = \lambda_3(A) = 1.618$, Since there are no triangles in C_5, the largest clique consists of an edge. Hence

$$3 = \lambda_1 > \omega_1(C_5) = \omega(C_5) = 2.$$

Moreover, since the graph has 2 non-adjacent edges, $\omega_2(C_5) = 4$. Thus,

$$4.618 = \lambda_1(A) + \lambda_2(A) > \omega_2(C_5) = 4.$$

Since the entire graph can be covered by 3 cliques, it is clear that $\omega_3(C_5) = 5$, and hence

$$6.236 = \lambda_1(A) + \lambda_2(A) + \lambda_3(A) > \omega_3(C_5) = 5.$$

In general, the gap between the two sides of the inequality in Theorem 1 may be arbitrarily large. For the graph C_5, the gap between $\lambda_1(A)$ and $\omega(G)$ is 1.0. The gap between $\lambda_1(A) + \lambda_2(A)$ and $\omega_2(C_5)$ is 0.618, and the gap between $\lambda_1(A) + \lambda_2(A) + \lambda_3(A)$ and $\omega_3(C_5$ is 1.236. It seems to be a very hard problem to characterize graphs for which $\sum_{i=1}^{k} \lambda_i(A) = \omega_k(G)$, even for a fixed value of k.

As mentioned before, $\sum_{i=1}^{k} \lambda_i(A)$ can be considered as a bound for $\omega_k(G)$. In the next section, we show how to improve on this bound, and we show how to simultaneously compute a bound for $\chi_k(G)$.

4.3. The Sandwich Theorem

In an attempt to estimate the parameters $\omega(G)$ and $\chi(G)$, Lovász introduced a function $\vartheta(G)$ that always lies between $\omega(G)$ and $\chi(G)$, and which can be computed in polynomial time [Lov86]. Hence Lovász's Theorem can be stated as follows:

Theorem 5: For any graph G, there exists a polynomial-time computable function $\vartheta(G)$, which satisfies the following inequality:

$$\omega(G) \leq \vartheta(G) \leq \chi(G).$$

We generalize Lovász's Theorem by defining a function $\vartheta_k(G)$ and proving the following theorem:

Theorem 6: For any graph G, there exists a polynomial-time computable function $\vartheta_k(G)$, which satisfies the following inequality:

$$\omega_k(G) \leq \vartheta_k(G) \leq \chi_k(G), \quad \text{for } 1 \leq k \leq n.$$

Theorem 1 provides a polynomial-time computable upper bound for $\omega_k(G)$. Theorem 6 refines this bound.

We break up the proof of Theorem 6 into three parts. It will follow from Theorem 7 that $\omega_k(G) \leq \vartheta_k(G)$. In Theorem 9 we prove that $\vartheta_k(G) \leq \chi_k(G)$. Finally, in Theorem 12 we prove that $\vartheta_k(G)$ can be computed in polynomial time.

We first describe Lovász's function $\vartheta(G)$, and then we describe the function $\vartheta_k(G)$. Let G be a graph defined on $V(G) = \{1, \ldots, n\}$. Let $\mathcal{A}(G)$ be the set of all real symmetric $n \times n$ matrices $A = (a_{ij})$ for which $a_{ij} = 1$ iff $i = j$ or i and j are adjacent in G. The elements of A corresponding to non-adjacent positions are allowed to be arbitrary real numbers, provided that the matrix remains symmetric. The function $\vartheta(G)$ is defined as:

$$\vartheta(G) = \min\{\lambda_1(A) : A \in \mathcal{A}(G)\}.$$

Lovász showed [Lov86] that the minimum is attained by showing that the minimization needs to be performed on a compact subset of $\mathcal{A}(G)$, hence proving that $\vartheta(G)$ is well-defined.
Analogous to $\vartheta(G)$, we define $\vartheta_k(G)$ as follows:

$$\vartheta_k(G) = \min\Big\{\sum_{i=1}^{k} \lambda_i(A) : A \in \mathcal{A}(G)\Big\}, \quad \text{for } 1 \leq k \leq n.$$

Similar to Lovász's argument, we show in Theorem 12 that this minimum is attained by showing that the minimization needs to be performed on a compact subset of $\mathcal{A}(G)$. Thus $\vartheta(G)$ is also a well-defined function.

Theorem 6 generalizes Lovász's result, since for $k = 1$, it is clear that $\omega_k(G) = \omega(G)$ and $\chi_k(G) = \chi(G)$, and in this case the function that we find, $\vartheta_k(G)$, is identical to Lovász's function, $\vartheta(G)$.

Theorem 7: For every matrix $B \in \mathcal{A}(G)$,

$$\sum_{i=1}^{k} \lambda_i(B) \geq \omega_k(G), \quad \text{for } 1 \leq k \leq n.$$

Proof of Theorem 7: The proof follows from Theorem 1 by replacing A by B, and A_H by the corresponding member of $\mathcal{A}(H)$. It is clear that all of the arguments still hold. ∎

Corollary 8: For any graph G, $\omega_k(G) \leq \vartheta_k(G)$, for $1 \leq k \leq n$.

Theorem 9: For any graph G, $\vartheta_k(G) \leq \chi_k(G)$, for $1 \leq k \leq n$.

Before we prove Theorem 9, we first show in Lemma 10 that a k-multicoloring of a graph is equivalent to a regular coloring of a modified graph. Define the *Composition* of two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ to be the graph $G(V, E) = G_1[G_2]$ such that $V = V_1 \times V_2$ and $((x_i, y_i), (x_j, y_j)) \in E$ iff either $x_i = x_j$ and $(y_i, y_j) \in E_2$, or $(x_i, x_j) \in E_1$. Given a graph G, let $G^{(k)} = G[K_k]$, where K_k is the complete graph on k vertices. The graph $G^{(k)}$ can be considered as the graph obtained by replacing each vertex v_i in G by the graph G_i, where each G_i is the complete graph on k vertices, and then adding all possible edges between G_i and G_j if v_i and v_j were adjacent in G.

Lemma 10: For any graph G, $\chi_k(G) = \chi(G^{(k)})$.

Proof: Every k-multicoloring of G can be converted into a coloring of $G^{(k)}$ by assigning the set of k distinct colors given to vertex v_i to the k vertices in G_i. Since this process is reversible, the lemma is proved. ∎

Lemma 11: Let G be a graph with n vertices. There exists a real, symmetric matrix B of order nk such that $\vartheta(G^{(k)}) = \lambda_1(B)$, and $B_{ij} = 1$ for $i = j$ and for i, j adjacent. Furthermore, B can be divided into n^2 blocks (possibly after a rearrangement of rows and columns), where each block is of size $k \times k$ and all the entries within a block are equal.

Proof: As mentioned earlier, the graph $G^{(k)}$ can be considered as the graph obtained by replacing each vertex v_i in G by a graph G_i, where each G_i is the complete graph on k vertices, and then adding the appropriate edges. Each of the k vertices in G_i are similar, in the sense that they are adjacent to the same set of vertices outside of G_i, besides being adjacent to each other.

By Lovász's Theorem, $\vartheta(G^{(k)}) \leq \chi(G^{(k)})$. Moreover, there exists a real, symmetric matrix $B \in \mathcal{A}(G^{(k)})$ of order nk such that $\lambda_1(B) = \vartheta(G^{(k)})$.

Since $\lambda_1(B)$ is a convex function of the entries in B [Fan49], it follows that there exists a matrix B such that $\lambda_1(B) = \vartheta(G^{(k)})$ and the rows (columns) corresponding to the similar (and mutually adjacent) vertices within each G_i are identical. Consequently, B can be divided into n sets of k rows (columns) such that each of the rows (columns) in a set is identical. Since B is symmetric, this proves that there exists a permutation matrix P such that PBP^{-1} can be divided into n^2 blocks, where each block is of size $k \times k$ and all the entries within a block are equal. ∎

Proof of Theorem 9: Let B be the real, symmetric matrix of order nk such that $\lambda_1(B) = \vartheta(G^{(k)})$. Furthermore, assume that B has the structure as in Lemma 11. That is, it can be divided into n^2 blocks (possibly after a rearrangement of rows and columns), where each block is of size $k \times k$ and all the entries within a block are equal. Lemma 11 guarantees the existence of such a matrix B. Now consider the $n \times n$ symmetric matrix A obtained from B by replacing each $k \times k$ block in B by any entry from that block (the entries are all equal).

It is obvious that if

$$\mathbf{e} = (x_1, \ldots, x_n)$$

is an eigenvector of A, then

$$\mathbf{e}' = (x_{11}, \ldots, x_{1k}, x_{21}, \ldots, x_{2k}, \ldots, x_{n1}, \ldots, x_{nk})$$

is an eigenvector of B such that

$$x_{ij} = x_i, \quad j = 1, \ldots, k, \quad i = 1, \ldots, n.$$

Hence if $\lambda_1(A), \ldots, \lambda_n(A)$ are the n eigenvalues of A, and $\lambda_1(B), \ldots, \lambda_{nk}(B)$ are the nk eigenvalues of B, then they are related as follows:

$$\lambda_{k(i-1)+j}(B) = k\lambda_i(A), \quad j = 1, \ldots, k.$$

In particular, $\lambda_1(B) = k\lambda_1(A)$. Define $\Phi_k(A) = \sum_{i=1}^{k} \lambda_i(A)$. Hence,

$$\Phi_k(A) = \sum_{i=1}^{k} \lambda_i(A) \leq k\lambda_1(A) = \lambda_1(B) \leq \chi(G^{(k)}) = \chi_k(G).$$

It is clear that $A \in \mathcal{A}(G)$. Since $\vartheta_k(G) = \min\{\Phi_k(M) : M \in \mathcal{A}(G)\}$, we have

$$\vartheta_k(G) \leq \Phi_k(A) \leq \chi_k(G),$$

and the theorem is proved. ∎

Theorem 12: Given graph G, $\vartheta_k(G)$ can be computed in polynomial time for $1 \leq k \leq n$.

Proof of Theorem 12: The proof that $\vartheta_k(G)$ can be computed in polynomial time is very similar to Lovász's proof that $\vartheta(G)$ can be computed in polynomial time [Lov86]. It was proved by Fan [Fan49] that for all $n \times n$ symmetric matrices A, the functions

$$\Phi_k(A) = \sum_{i=1}^{k} \lambda_i(A), \quad k = 1, \ldots, n,$$

are increasing and convex. This convex function has to be minimized over the affine subspace $\mathcal{A}$, which can be done using the ellipsoid algorithm, provided we can find a ball or cube that is known *a priori* to contain the minimum.

If the largest entry in matrix A is denoted by $\|A\|_\infty \leq n^2$, then it was shown by Lovász [Lov86] that the matrix A that minimizes $\lambda_1(A)$ over $\mathcal{A}$ satisfies the property that $\|A\|_\infty \leq n^2$. It turns out that similar arguments show that the matrix B that minimizes $\Phi_k(B)$ over $\mathcal{A}$ satisfies the property that $\|B\|_\infty \leq n^3$. The argument proceeds as follows: Suppose that $B \in \mathcal{A}$ minimizes $\Phi_k(B)$. Then we can write $B = T^* D T$, where T is an orthogonal matrix, T^* is the conjugate transpose of T, and D is the diagonal matrix formed by the eigenvalues $\lambda_1, \ldots, \lambda_n$ (in non-increasing order) of B. Note that $\sum_{i=1}^{n} \lambda_i = trace(B) = n$, and hence $\lambda_1 \geq \lambda_i = n - \sum_{j \neq i} \lambda_j \geq n - (n-1)\lambda_1$ for all i. Since trivially $\lambda_1 \leq kn \leq n^2$, we have $|\lambda_i| \leq n^3$. Hence

$$|A_{ij}| = \left| \sum_{k=1}^{n} \lambda_k(G) T_{ik} T_{jk} \right| \leq n^3 \sum_{k=1}^{n} |T_{ik}||T_{jk}|$$

$$\leq n^3 \left(\sum_{k=1}^{n} |T_{ik}|^2 \right)^{\frac{1}{2}} \left(\sum_{k=1}^{n} |T_{jk}|^2 \right)^{\frac{1}{2}} \leq n^3 .$$

So to determine $\vartheta_k(G)$ it suffices to minimize $\Phi_k(A)$ over the region $\{A \in \mathcal{A} : \|A\|_\infty \leq n^3\}$. Since for each rational $A \in \mathcal{A}$, $\Phi_k(A)$ can be computed in polynomial time, we have proved that $\vartheta_k(G)$ can be computed in polynomial time. ∎

Proof of Theorem 6: This theorem follows from Corollary 8, and Theorems 10, and 12. ∎

Consider again the example of C_5, the simple cycle with five vertices. As mentioned earlier, $\omega_2(C_5) = 4$, and it turns out that $\vartheta_2(C_5) = 2\sqrt{5}$. Clearly, the bound obtained from the inequality in Theorem 6 will

always be at least as good as the bound obtained from the inequality in Theorem 1. For the example of C_5, Theorem 6 improves on the bound from Theorem 1.

In the following section, we discuss the consequences of the generalized sandwich theorem, particularly for perfect and t-perfect graphs. We study the family of graphs for which the generalized sandwich theorem is satisfied with equality.

4.4. Consequences of the Generalized Sandwich Theorem

4.4.1. Case of Perfect Graphs

As mentioned earlier, for perfect graphs, $\omega(G) = \chi(G)$, and hence the function $\vartheta(G)$ coincides with $\omega(G)$ and $\chi(G)$, a fact that was exploited by Lovász, et al. [GLS81] to design a polynomial-time algorithm to compute $\omega(G)$ and $\chi(G)$ for perfect graphs. If $\omega_k(G) = \chi_k(G)$ for all perfect graphs, then our generalized sandwich theorem could help to compute $\omega_k(G)$ and $\chi_k(G)$ in polynomial time. Unfortunately, $\omega_k(G)$ and $\chi_k(G)$ need not be equal for all perfect graphs. A simple example is sufficient to prove this fact. Consider K_n, the complete graph on n vertices. K_n is a perfect graph. But $\omega_k(G) = \frac{\chi_k(G)}{k}$ for $1 \leq k \leq n$. Hence $\omega_k(G)$ and $\chi_k(G)$ can be arbitrarily far apart. In fact, for K_n $\omega_k(G) = \vartheta_k(G) < \chi_k(G)$, for $k > 1$.

For the case of perfect graphs, it is not surprising that $\omega_k(G)$ and $\chi_k(G)$ are not necessarily equal. It is known that $\chi_k(G)$ can be computed in polynomial time for perfect graphs. This follows from the fact that $\chi(G)$ can be computed in polynomial time for perfect graphs [GLS81], and $\chi_k(G) = \chi(G^{(k)}) = k\chi(G)$ is true for this class of graphs. In contrast, for arbitrary k, computing $\omega_k(G)$ for perfect graphs is a NP-hard problem. This follows from the fact that it is NP-hard to compute $\omega_k(G)$ (for k arbitrary) for the class of chordal graphs [YG87], which is a subclass of the class of perfect graphs. It is also unlikely that $\omega_k(G) = \vartheta_k(G)$ for all perfect graphs for the same reason that $\vartheta_k(G)$ can be computed in polynomial time, while $\omega_k(G)$ is NP-hard to compute.

It is clear that computing $\vartheta_k(G)$ gives a polynomial-time computable bound for both $\omega_k(G)$ and $\chi_k(G)$, for $1 \leq k \leq n$. The example of the complete graph on n vertices showed that $\vartheta_k(G)$ and $\chi_k(G)$ can be arbitrarily far apart, even for perfect graphs. It is an open question to determine how far apart $\omega_k(G)$ and $\vartheta_k(G)$ can be for perfect graphs, if $k > 1$. It is also an open question to determine how far apart $\omega_k(G)$ and $\vartheta_k(G)$ can be for arbitrary graphs, even for the case $k = 1$.

We mention an interesting open question regarding the integrality of the ϑ functions defined in this chapter. Since for perfect graphs, $\omega(G) = \vartheta(G) = \chi(G)$, it is clear that $\vartheta(G)$ is integral for perfect graphs. It would be interesting to study whether for perfect graphs, $\vartheta_k(G)$, $k > 1$ is integral.

In section 4.4.2, we study the graphs for which $\omega_k(G) = \chi_k(G)$ for some k between 1 and n.

4.4.2. Equality Case

The natural question to ask is whether there exist any classes of graphs for which the generalized sandwich theorem is satisfied with equality. In other words, it would be interesting to characterize the class of graphs for which $\omega_k(G) = \chi_k(G)$. We are not aware of a naturally occurring class of graphs for which this is true. Proposition 1 gives a simple observation about the class of graphs for which $\omega_k(G) = \chi_k(G)$. It also indicates that this class of graphs is a non-trivial one.

Proposition 1: If graph G satisfies $\omega_k(G) = \chi_k(G)$ for some $k \leq n$, then G has at least k disjoint cliques of size $\omega(G)$.

Proof: For any graph G, $\omega_k(G) \leq k\omega(G) \leq \chi_k(G) \leq k\chi(G)$. Since we know that G satisfies $\omega_k(G) = \chi_k(G)$ for some $k \leq n$, it means that $\omega_k(G) = k\omega(G)$. Hence G has a subgraph of size $k\omega(G)$ that can be covered by k cliques. Since G has no clique of size greater than $\omega(G)$, G must have at least k cliques of size $\omega(G)$. ∎

Our aim now is to construct a graph G for which our generalized sandwich theorem helps to compute $\omega_k(G)$ and $\chi_k(G)$, and for which previously known techniques do not apply. Hence if G is such a graph, then $\omega_k(G) = \chi_k(G)$. Moreover, by proposition 1, graph G also satisfies the property that $\omega_k(G) = k\omega(G)$. This implies that for a graph G, if the generalized sandwich theorem helps to compute $\omega_k(G)$, it also helps to compute $\omega(G) = \omega_k(G)/k$. Therefore, we need to construct a graph G for which previously known techniques can help compute neither $\omega(G)$ nor $\omega_k(G)$. In particular, such a graph must be imperfect, since for perfect graphs $\omega(G)$ can be computed in polynomial time.

The graph C_5 is an example of an imperfect graph for which $\omega_2(C_5) = 2\omega(C_5)$. However, $\omega_2(C_5) < \chi_2(C_5)$. We now consider another example of a graph G that is imperfect, but for which $\omega_2(G) = 2\omega(G) = \chi_2(G)$.

Consider the graph $G(V, E)$ shown in figure 4.1. Its vertex set $V(G)$ is a union of three sets A, B and C, where $A = \{a_1, a_2, a_3, a_4, a_5\}$, $B = \{b_1, b_2, b_3, b_4, b_5\}$, and $C = \{c_1, c_2\}$. The vertices in A form a simple cycle of 5 vertices with edges (a_1, a_2), (a_2, a_3), (a_3, a_4), (a_4, a_5), and (a_5, a_1). The vertices in B also form a simple cycle of 5 vertices with edges (b_1, b_2), (b_2, b_3), (b_3, b_4), (b_4, b_5), and (b_5, b_1). Every vertex in A is joined by an edge to every vertex in B. Finally, vertex c_1 is connected to a_1, a_2, b_1, and b_2, while vertex c_2 is connected to a_4, a_5, b_4, and b_5. Vertices c_1 and c_2 are not adjacent. We will now show that the graph G satisfies our requirements. In other words, we will show that $\omega(G) < \vartheta(G) < \chi(G)$, while $\omega_2(G) = \vartheta_2(G) = \chi_2(G)$, which implies that previously known techniques are not applicable to compute $\omega(G)$, while the generalized sandwich theorem can be applied to compute $\omega_2(G), \chi_2(G)$, and consequently to compute $\omega(G)$. We note that G is imperfect since it has an induced odd cycle of length 5.

It is easy to verify that $\{a_1, a_2, c_1, b_1, b_2\}$ and $\{a_4, a_5, c_2, b_4, b_5\}$ are two maximum cliques in G. Hence $\omega(G) = \omega_1(G) = 5$, and $\omega_2(G) = 10$. Each of the two vertex sets A and B form odd cycles. Hence 3 colors are needed to color either A or B. The colors used to color vertices in A have to be distinct from the ones

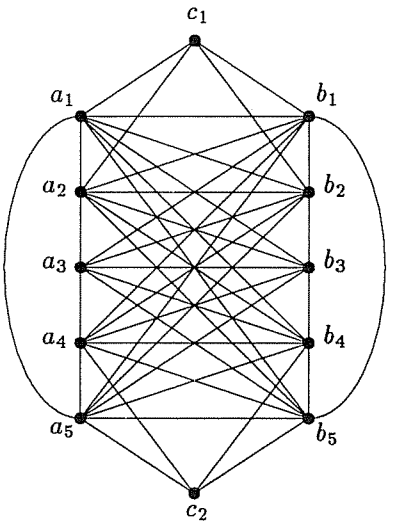


Figure 4.1: Graph G

used to color vertices in B. Hence $\chi(G) \geq 6$. It is easy to verify that $\chi(G) = 6$, if the six independent sets $\{a_1, a_4\}, \{b_1, b_4\}, \{a_2, a_5\}, \{b_2, b_5\}, \{a_3, c_1\}$, and $\{b_3, c_2\}$ are each assigned one color each. Lovász [Lov79] has shown that $\vartheta(C_5) = \sqrt{5}$. Since G has the graph C_5 as an induced subgraph, $\vartheta(G) \geq \vartheta(C_5) = \sqrt{5}$. Using techniques similar to those used by Lovász in [Lov79], it can be shown that, in fact, $\vartheta(G) = 2\sqrt{5}$. Hence, we have shown that $\omega(G) < \vartheta(G) < \chi(G)$. Now we are left to prove that $\chi_2(G) = 10$. Clearly, $\chi_2(G) \geq \omega_2(G) = 10$. Color the vertices a_1, a_2, a_3, a_4, a_5 with the 2-sets $\{1,2\}$, $\{3,4\}$, $\{5,1\}$, $\{2,3\}$, and $\{4,5\}$ respectively. Now color the vertices b_1, b_2, b_3, b_4, b_5 with the 2-sets $\{1',2'\}$, $\{2',3'\}$, $\{3',4'\}$, $\{4',5'\}$, and $\{5',1'\}$ respectively. Finally, color the vertex c_1 with the 2-set $\{5,5'\}$, and the vertex c_2 with the 2-set $\{1,1'\}$. It is trivial to verify that this is a valid 2-multicoloring with 10 colors.

Summarizing the above example, we have a graph G for which $\omega(G) < \vartheta(G) < \chi(G)$, and for which $\omega_2(G) = \vartheta_2(G) = \chi_2(G)$. Computing Lovász's function $\vartheta(G)$ cannot help in computing $\omega(G)$. However, since $\omega_2(G) = \vartheta_2(G) = \chi_2(G) = 2\omega(G)$, the generalized sandwich theorem helps us to compute $\omega_2(G), \chi_2(G)$, and $\omega(G)$ (but not $\chi(G)$).

The above example can be generalized for an arbitrary value of k, thus producing a family of graphs for which the generalized sandwich theorem is useful in computing certain graph parameters. It remains an open problem to properly characterize the family of graphs for which the generalized sandwich theorem is useful in determining any of the interesting graph parameters.

4.4.3. Case of k-Perfect Graphs

We explained earlier why the generalized sandwich theorem is not useful to compute $\alpha_k(G)$, $\omega_k(G)$, and $\chi_k(G)$ for all perfect graphs. The example of the perfect graph K_n (the complete graph on n vertices) exposed one of the main shortcomings of the generalized sandwich theorem. The parameter $\chi_k(G)$ can be larger than n (the number of vertices in G), whereas $\omega_k(G)$ and $\vartheta_k(G)$ are bounded above by n. It is possible that a better sandwich theorem can be obtained by replacing $\chi_k(G)$ by some other parameter.

Greene [Gre76] and Greene and Kleitman [GK76] studied a parameter, which we denote by $\psi_k(G)$, and which they define as follows:

$$\psi_k(G) = \min\{k\chi(G - S) + |S| : S \subseteq V(G)\}$$

Clearly, $\psi_1(G) = \psi(G)$, and $\psi_k(G) \leq n$, for any $k \geq 1$. Furthermore, Lovász [Lov83] defined a class of graphs called the *k-perfect graphs*. A graph G is k-perfect if $\omega_k(H) = \psi_k(H)$ is true for every induced subgraph H of G. Lovász [Lov83] also showed that this is a non-trivial class of graphs by proving that comparability and co-comparability graphs are k-perfect, for all k.

It would be interesting to determine if the inequalities in the generalized sandwich theorem still hold if we replace $\chi_k(G)$ by $\psi_k(G)$. Unfortunately, we have not been able to prove or disprove the above statement. If proved, it would give an algorithm to compute $\omega_k(G)$ and $\psi_k(G)$ for k-perfect graphs, which is an interesting open question posed by Lovász in [Lov83].

Chapter 5

Approximation Algorithms

In Chapter 4 we showed new techniques to find a polynomial-time computable upper bound for the size of the largest subgraph in a graph that can be covered with k cliques, and the size of the maximum k-colorable subgraph in the graph.

Having found a polynomial-time computable upper bound for the size of the maximum k-colorable subgraph, we show in this chapter how devising an approximation algorithm for the maximum k-colorable subgraph problem is equivalent to finding a lower bound for this same graph parameter. If an approximation algorithm is guaranteed to find a large k-colorable subgraph, then the size of the k-colorable subgraph found by this algorithm is effectively a lower bound for the size of the largest k-colorable subgraph. As we prove later, it is possible to design approximation algorithms only for special cases.

Our aim in this chapter is to find approximation algorithms for the Maximum k-Colorable Subgraph (MkCS) Problem. We shall first study why this problem is hard. In Section 5.1 we show that the MkCS problem is polynomially equivalent to the Maximum Independent Set (MIS) Problem. In Section 5.2 we prove negative results about the existence of "good" approximation algorithms for the MkCS problem by showing that for general graphs, it is unlikely that any "good" approximation algorithms can be found for the MkCS problem. In contrast, we also show that "good" approximation algorithms can be found for the problem when the inputs are restricted to some special classes of graphs. In particular, in Section 5.3 we show that greedy approximation algorithm is guaranteed to perform well for those special classes of graphs for which there exist polynomial-time algorithms to solve the MIS problem. In Section 5.3.1 we analyze the greedy algorithm for the case $k = 2$, i. e., for the maximum induced bipartite subgraph problem (MIBS). In Section 5.3.2 we generalize the approximation algorithm for the MIBS problem to an approximation algorithm for the MkCS problem, and we find an optimal performance ratio for this algorithm. To prove that this ratio is optimal, we construct worst-case examples to show that there are examples for which the performance ratios are asymptotically reached, thus showing that our analysis is tight.

5.1. Polynomial Equivalence of the Maximum k-Colorable Subgraph Problem to the Maximum Independent Set Problem

In this section we first show that the Maximum k-Colorable Subgraph (MkCS) Problem is polynomial-time equivalent to the Maximum Independent Set (MIS) Problem. In other words, the MkCS problem can be reduced to the MIS problem in polynomial time and vice versa.

The MkCS problem was proved to be NP-Hard by Lewis and Yannakakis [LY80]. In fact, they proved a much stronger and more general result. They considered the following problem, which they called the *node-deletion* problem : For a fixed graph property Π, find the minimum number of vertices that must be deleted from a given graph so that the resulting subgraph satisfies Π. They show that if Π satisfies the two conditions given below then the node-deletion problem is NP-Hard. The two conditions are: (1) Π is a *non-trivial* property (i. e., it is true for infinitely many graphs and false for infinitely many graphs); (2) Π is *hereditary* on induced subgraphs (i. e., if a graph satisfies the property, then every induced subgraph also satisfies the property); Furthermore, if the property Π can be tested in polynomial time, then the node-deletion problem is NP-Complete.

It is obvious that the property "the graph is k-colorable" is both non-trivial and hereditary on induced subgraphs. Thus the MkCS problem is NP-Hard. For the case of $k \leq 2$, the property "the graph is k-colorable" can be tested in polynomial time and hence the maximum 2-colorable subgraph problem is NP-Complete. The proof given by Lewis and Yannakakis [LY80] gives powerful techniques to prove the NP-Hardness of a large class of problems. The arguments used in the proof are existential and are based on an ordering of graphs and on Ramsey's Theorem applied to graphs. Hence, even though this powerful result proves that a number of node-deletion problems are hard, it fails to provide us with an intuition about why any particular problem is hard.

We now present a simple and direct proof of NP-Hardness of the MkCS problem. We show that it is polynomially equivalent to the MIS problem. This provides some intuition behind the hardness of the MkCS problem.

Theorem 1: The MIS problem can be reduced in polynomial time to the MkCS problem.

Proof of Theorem 1: Given a graph $G(V, E)$ and a number B as an instance of the MIS problem, we need to determine if G contains an independent set of size at least B. Construct G' by making k copies of G and adding all edges connecting vertices in different copies.

More formally, given $G(V, E)$, with $V = \{v_i : 1 \leq i \leq n\}$, construct $G'(V', E')$ as follows:

$$V' = \{v_{ij} : 1 \leq i \leq n, 1 \leq j \leq k\}$$
$$E' = \{(v_{ij}, v_{kl}) : j \neq l, \text{ or } j = l \text{ and } (v_i, v_k) \in E\}$$

We claim that G has an independent set of size B iff G' has a k-colorable subgraph of size kB.

One direction of the claim is easy to prove. If G has an independent set of size at least B, pick the same set of vertices from each of the k copies of G in G'. These vertices induce a k-colorable subgraph of size kB.

For the other direction, assume that G' has a k-colorable subgraph of size kB. Since the subgraph is k-colorable, it can be covered with k independent sets $S_1, S_2, \ldots, S_k$. Any S_i must lie completely within one of the k copies G_j, $1 \leq j \leq k$, since any two vertices in different copies are adjacent. Furthermore, at least one of the independent sets S_i must be of size at least B. Hence there exists an independent set of size

at least B in one of the k copies of G in G', which means that there is an independent set of size B in G. Thus the claim is proved. Thus the MIS problem can be reduced in polynomial time to the MkCS problem, completing the proof of Theorem 1. ∎

Since the MIS problem is NP-Hard, the MkCS problem is also NP-Hard. Next we show a simple reduction from the MkCS problem to the MIS problem.

Theorem 2: The MkCS problem can be reduced in polynomial time to the MIS problem.

Proof of Theorem 2: Given a graph $G(V, E)$ and a number B as an instance of the MkCS problem, we need to determine if G contains a k-colorable subgraph of size at least B. Construct G'' by making k copies of G. There are k copies of each vertex $v_i \in V$ in G'' (call them $v_{i1}, \ldots, v_{ik}$). In G'' connect the k copies of a vertex to form a clique, i. e., the vertices $v_{i1}, \ldots, v_{ik}$ form a clique in G''.

Formally, given $G(V, E)$, with $V = \{v_i : 1 \leq i \leq n\}$, we construct the graph $G''(V'', E'')$ as follows:

$$V'' = \{v_{ij} : 1 \leq i \leq n, 1 \leq j \leq k\}$$
$$E'' = \{(v_{ij}, v_{kl}) : i = k \text{ and } j \neq l, \text{or } j = l \text{ and } (v_i, v_k) \in E\}$$

We claim that G has a k-colorable subgraph of size B iff G'' has an independent set of size B.

If G has a k-colorable subgraph of size B, then the subgraph can be covered with k independent sets $S_1, S_2, \ldots, S_k$. Let $S = \{v_{ij} : v_i \in S_j, 1 \leq j \leq k\}$. Clearly S is an independent set in G'' of size B.

If G'' has an independent set S of size B, construct sets $S_1, \ldots, S_k$ as follows. For every vertex $v_{ij} \in S$, put v_i in S_j. If for some $i \neq l$, we have $v_i, v_l \in S_j$, then $(v_{ij}, v_{lj}) \notin E''$, which implies that $(v_i, v_l) \notin E$. Hence each of the sets S_j, $1 \leq j \leq k$, is an independent set in G. Thus $S_1 \cup \ldots \cup S_k$ induces a k-colorable subgraph in G of size equal to B.

Clearly the above reduction can be done in polynomial time. This completes the proof of Theorem 2. ∎

Corollary 3: The MIS and the MkCS problem are polynomial-time equivalent.

5.2. Approximation Algorithms for the MkCS problem – Negative Results

The *Absolute Performance Ratio* of an approximation algorithm A for a maximization problem P is the infimum of the ratio between the size of the optimal solution and the size of the approximate solution over all instances of P. It is denoted by $R_A(P)$. If $R_A(P)$ can be shown to be bounded by a constant α for a NP-hard problem P, then we call algorithm A to be a "good" approximation algorithm for problem P.

Theorem 4 below is a well-known negative result regarding the existence of a good approximation algorithm for the Maximum Independent Set problem (MIS) on general graphs. It shows that if some

algorithm A for problem P is guaranteed to produce a solution whose size is within a constant factor α (for some $\alpha > 1$) of the optimal solution for P, then this algorithm can always be improved upon. In this case, the theorem shows that we can always construct another algorithm (call it A_β) for problem P that is guaranteed to produce a solution whose size is within a constant factor β, where β is any constant greater than 1. This theorem essentially rules out the possibility of a good approximation algorithm for the MIS problem.

Theorem 4: [GJ79] If $R_A(MIS) < \alpha$ for some $\alpha > 1$, then for any $\beta > 1$, there is an approximation algorithm A_β with $R_{A_\beta}(MIS) < \beta$.

Another consequence of the theorem is that since for $k = 1$, the MkCS problem is identical to the MIS problem, no good approximation algorithms are likely to exist for the MkCS problem on general graphs. Even though good approximation algorithms are not likely to exist for the MkCS problem, there may still exist good approximation algorithms when the input is restricted to special classes of graphs. In section 5.3.1 we present good approximation algorithms for the maximum bipartite subgraph problem (or the maximum 2-colorable subgraph problem) on those classes of graphs for which there exist polynomial-time algorithms to solve the MIS problem. We prove a near optimal bound on the absolute performance ratio of the algorithm. In particular, this approximation algorithm would be good on circle graphs, and perfect graphs. In Section 5.3.2 we outline a natural extension of this algorithm to the MkCS problem for the same classes of graphs.

5.3. Approximation Algorithms for Special Classes of Graphs

Consider classes of graphs for which the maximum independent set problem can be solved in polynomial time. Circle graphs and perfect graphs are two classes of graphs for which this is true ([Ga73],[GLS81]). A polynomial time algorithm for the maximum independent set problem provides us with simple approximation algorithms for the MkCS problem on graphs satisfying the above mentioned property. The idea is simply to find the maximum independent set, delete it, find another maximum independent set in the remaining graph, and continue until k independent sets are found. We call this the *naive* approximation algorithm or the *greedy* approximation algorithm. We show here that this simple idea does, in fact, provide a good approximation algorithm for the problem. We first analyze the greedy approximation algorithm for the MIBS problem. This is done in section 5.3.1, and later in section 5.3.2 we analyze the greedy approximation algorithm for the MkCS problem.

As mentioned in the introductory chapter, the MIBS problem has important applications in a problem from VLSI (the via minimization problem). In this application the input is restricted to circle graphs. It was recently shown [SL87] that the MIBS problem is NP-Hard for circle graphs. Thus the greedy approximation algorithm provides a good heuristic for the via minimization problem.

5.3.1. Analysis for the MIBS Case

Denote by R_2 the ratio of the size of a maximum induced bipartite subgraph in the graph to the size of the induced bipartite subgraph obtained using the algorithm. We start with a straightforward analysis of the algorithm. Later we show an improved analysis of the same algorithm.

Let P and Q be two independent sets in G such that the subgraph induced by vertices in P and Q is a largest 2-colorable (bipartite) subgraph in G. Let A and B be the two independent sets of G that are produced by the greedy algorithm. Denote the number of vertices in A, B, P, and Q by a, b, p, and q, respectively. We know that $a \geq b$. Without loss of generality, we can assume that $p \geq q$. It is obvious that $(a+b) \geq a \geq p \geq q$. Hence

$$(a+b) \geq (p+q)/2$$

Thus, a simple analysis proves that the naive approximation algorithm provides an induced bipartite subgraph that is at least half the size of a maximum induced bipartite subgraph. Hence we have shown that $R_2 \leq 2$.

We proceed with a more careful analysis and show that, in fact, $R_2 \leq 4/3$.

Theorem 5: If R_2 is the absolute performance ratio of the naive approximation algorithm for the MIBS problem, then $R_2 \leq 4/3$.

Proof of Theorem 5: Let A, B, P, and Q be as described in the previous section. Let $P_1 = P \cap A$ and $P_2 = P - P_1$. Let the number of vertices in P_1 be p_1, and the number of vertices in P_2 be p_2. Similarly, let $Q_1 = Q \cap A$ and $Q_2 = Q - Q_1$. Once again let q_1 be the number of vertices in Q_1, and q_2 be the number of vertices in Q_2.

The key point to note here is that in the algorithm when the vertices in A were removed, we removed the vertices in P_1 and Q_1 (and possibly more). In the remaining graph, P_2 and Q_2 must still be independent sets, and since we found the largest independent set in the remaining graph, the independent set B must be at least as large as P_2 and Q_2. Thus, $b \geq \max\{p_2, q_2\}$. We shall consider two cases:

Case 1 : If $q_1 \leq q_2$, then $b \geq q_2 \geq q/2$. Then

$$\begin{aligned}(a+b) &\geq p + (q/2) \\ &\geq (3/4)(p+q)\end{aligned}$$

Case 2 : Now assume that $q_1 \geq q_2$. Since $b \geq p_2$, we have

$$\begin{aligned}(a+b) &\geq (p_1+q_1) + p_2 \geq p + (q/2) \\ &\geq (3/4)(p+q)\end{aligned}$$

In either case we have proved that $R_2 \leq 4/3$. That completes the proof of Theorem 4. ∎

We prove in Section 5.3.3 that this bound is tight by showing examples for which the bound is reached asymptotically.

5.3.2. Analysis of the Approximation Algorithm for the MkCS Problem

The naive approximation algorithm described above can be extended to find the largest k-colorable subgraph. Hence instead of stopping after finding two independent sets in a greedy fashion, it could be extended to find k independent sets in a greedy fashion. Theorem 6 proves that this algorithm would guarantee a solution with a performance ratio of $2k/(k+1)$, which we denote by R_k.

Theorem 6: If R_k denotes the absolute performance ratio for the naive approximation algorithm for the MkCS problem, then

$$R_k \leq \frac{2k}{k+1}$$

Proof of Theorem 6: We shall prove this by induction on k. It is true for $k = 2$ since $R_2 \leq 4/3$. Now assume that it is true for $(k-1)$, i. e., assume that $R_{k-1} \leq 2(k-1)/k$, where R_{k-1} is the absolute performance ratio of the naive approximation algorithm to find the largest $(k-1)$-colorable subgraph. Let the k independent sets returned by the algorithm be $A_1, A_2, \ldots, A_k$. Let their sizes be $a_1, \ldots, a_k$, respectively. Let a maximum k-colorable subgraph consist of the independent sets $P_1, P_2, \ldots, P_k$. Similarly, let their sizes be $p_1 \geq \ldots \geq p_k$.

For $1 \leq i \leq k$, let $P_{i1} = P_i \cap A$, and $P_{i2} = P_i - P_{i1}$. Let the number of vertices in P_{i1} be p_{i1}, and the number of vertices in P_{i2} be p_{i2}.

Similar to the earlier analysis, the key point is that in the algorithm when the vertices in $\cup_{i=1}^{k-1} A_i$ were removed, we removed the vertices in P_{i1}, $1 \leq i \leq k$ (and possibly more). In the remaining graph, P_{i2}, $1 \leq i \leq k$ must still be independent sets, and since we found the largest independent set in the remaining graph, the independent set B must be at least as large as the largest of P_{i2}, $1 \leq i \leq k$. Thus, $a_k \geq \max\{p_{12}, \ldots, p_{k2}\}$. Similar to the analysis in the previous section, we shall consider two cases:

Case 1 : If $\exists i$ such that $p_{i1} \leq p_{i2}$, then $a_k \geq p_{i2} \geq p_i/2 \geq p_k/2$. Then

$$\sum_{i=1}^{k} a_i \geq \frac{1}{R_{k-1}} \sum_{i=1}^{k-1} p_i + \frac{p_k}{2}$$

$$\geq \frac{k}{2(k-1)} \sum_{i=1}^{k-1} p_i + \frac{p_k}{2}$$

Case 2 : Now assume that for $1 \leq i \leq n$, $p_{i1} \geq p_{i2}$. Since $b \geq p_{12}$, we have

$$\sum_{i=1}^{k} a_i \geq \sum_{i=1}^{k} p_{i1} + p_{12}$$

$$\geq p_1 + \frac{1}{2}\sum_{i=2}^{k} p_i$$

$$= \Big(\frac{k}{2(k-1)} + \frac{k-2}{2(k-1)}\Big)p_1 + \Big(\frac{k}{2(k-1)} - \frac{1}{2(k-1)}\Big)\sum_{i=2}^{k-1} p_i + \frac{p_k}{2}$$

Since $(k-2)p_1 \geq \sum_{i=2}^{k-1} p_i$, we have

$$\sum_{i=1}^{k} a_i \geq \frac{k}{2(k-1)}\sum_{i=1}^{k-1} p_i + \frac{p_k}{2}$$

We note that both the cases have resulted in the same inequality. We need to simplify this inequality in order to obtain the desired inequality relating $\sum_{i=1}^{k} a_i$ and $\sum_{i=1}^{k} p_i$. Hence

$$\sum_{i=1}^{k} a_i \geq \frac{k}{2(k-1)}\sum_{i=1}^{k-1} p_i + \frac{p_k}{2}$$

$$= \frac{1}{2}\sum_{i=1}^{k-1} p_i + \frac{1}{2(k-1)}\sum_{i=1}^{k-1} p_i + \frac{p_k}{2}$$

$$= \frac{1}{2}\sum_{i=1}^{k} p_i + \frac{1}{2(k-1)}\sum_{i=1}^{k-1} p_i$$

Since $k\sum_{i=1}^{k-1} p_i \geq (k-1)\sum_{i=1}^{k} p_i$, the second term in the right side of the above inequality can be replaced by $\sum_{i=1}^{k} p_i/2k$. Hence we obtain the desired result, namely,

$$\sum_{i=1}^{k} a_i \geq \frac{k+1}{2k}\sum_{i=1}^{k} p_i$$

We have thus proved that

$$R_k \leq \frac{2k}{k+1}$$

That ends the proof of Theorem 5. ∎

As discussed earlier, the size of the k-colorable subgraph found by the greedy approximation algorithm is a lower bound for the size of the maximum k-colorable subgraph in the graph. We note again that the greedy approximation algorithm runs in polynomial time only if the MIS problem has a polynomial-time solution on that class of graphs. Hence this lower bound can be computed in polynomial time only for those classes of graphs for which the MIS problem is solvable in polynomial time. In particular, this approximation algorithm runs in polynomial time for the class of circle graphs.

5.3.3. Worst Case Examples

In this section we give examples of circle graphs for which the ratio R_k is asymptotically achieved. These examples prove that the analyses as described above give a tight bound for the greedy algorithms.

Let the vertex set of graph H be $V = A_1 \cup A_2 \cup B_1 \cup B_2$. Let $|A_1| = |A_2| = (n+1)$, and $|B_1| = |B_2| = n$. The edges of graph H consist of all possible edges between the vertex sets A_1 and B_2, A_2 and B_1, and B_1 and B_2. The largest 2-colorable subgraph is the entire graph H of size $(4n + 2)$. The naive approximation algorithm would return the subgraph induced by vertices in $A_1 \cup A_2 \cup B_1$ or $A_1 \cup A_2 \cup B_2$, both of which are of size $(3n + 2)$. It is clear that as n becomes large the ratio of the size of the largest bipartite subgraph to the size of the subgraph obtained by the greedy algorithm is 4/3. It is also easy to verify that the graph H is a circle graph.

The above example can be easily generalized to obtain a worst case example for the greedy approximation algorithm to find the largest k-colorable subgraph. We describe the construction below: Let the graph be denoted by $H^{(k)}$ with vertex set V and edge set E. Let $V = \{A_1 \cup A_2 \cup \ldots \cup A_k\} \bigcup \{B_1 \cup B_2 \cup \ldots \cup B_k\}$. Let $|A_i| = (n + 1)$, $1 \leq i \leq k$, and $|B_i| = n$, $1 \leq i \leq k$. The edges of the graph consist of all possible edges between the vertex sets:

(1) B_i and A_j, $k \geq j > i \geq 1$;

(2) B_k and A_j, $k > j \geq 1$;

(3) A_i and A_j, $k \geq j > i \geq 2$; and

(4) B_i and B_j, $k \geq j > i \geq 1$.

Once again, the largest k-colorable subgraph is the entire graph, which is of size $2kn + k$. The approximation algorithm would return the subgraph induced by the vertices in $A_1 \cup A_2 \cup \ldots \cup A_k \cup B_i$, for some $1 \leq i \leq k$. This solution is of size $(k + 1)n + k$. It is clear that the ratio of the optimal solution to that of the approximate solution is asymptotically equal to $k/(k + 1)$. It is tedious, but straightforward, to check that $H^{(k)}$ is a circle graph.

Chapter 6

Conclusions and Open Problems

In this dissertation we studied the maximum k-colorable subgraph (MkCS) problem, which is an NP-hard problem. In this chapter we summarize our conclusions on the various approaches we used to deal with the hardness of this problem, and we mention a few open questions.

The first approach we took was to solve various versions of this problem on several special classes of graphs. We studied several special classes of graphs all of which are of practical significance, and all of which are related to the class of interval graphs. Our algorithms, besides giving insight into the hardness of the maximum k-colorable subgraph problem, also gave insight into the structural and algorithmic aspects of the concerned special classes of graphs.

In particular, we have polynomial-time algorithms for the maximum bipartite subgraph problem on *interval graphs* and *circular-arc graphs*. We mentioned in chapter 2 that Yannakakis and Gavril [YG87] have shown that the greedy algorithm can be extended to solve the MkCS problem on interval graphs. However, on the class of circular-arc graphs it is not known whether a polynomial-time algorithm exists for the MkCS problem for any fixed k. Note that for arbitrary k the MkCS problem is NP-hard on circular-arc graphs. Curiously enough, it remains open even for the special case of $k = 3$.

We also designed polynomial-time algorithms for the maximum independent set problem, the maximum bipartite subgraph problem and the maximum k-colorable subgraph problem for fixed k on the class of *tolerance graphs*. The algorithmic aspects of the class of tolerance graphs have not been studied before. Numerous efficient algorithms have been designed for the class of co-comparability graphs [Go80]. Since bounded tolerance graphs are co-comparability graphs, algorithms for co-comparability graphs also apply to bounded tolerance graphs. However, the presence of unbounded vertices in general tolerance graphs make problems on them more complex. The most important open problem on tolerance graphs seems to be the recognition problem, a problem that was mentioned in [GMT84]. There is no known polynomial-time algorithm to recognize tolerance graphs. Based on our experience with designing algorithms for tolerance graphs, we believe that tolerance graphs are closer to comparability graphs rather than interval graphs. Hence while designing algorithms for tolerance graphs, it might be better to try adapting algorithms that have been designed for comparability graphs. Our algorithms give some insight into many of the properties of tolerance graphs.

Our algorithm for the maximum bipartite subgraph problem on tolerance graphs is not very efficient. It would be interesting to see if the algorithm can be improved. Since tolerance graphs are perfect, polynomial-time algorithms exist for computing the chromatic number of a tolerance graph. However, the algorithms

designed for perfect graphs [GLS81] are not very efficient as they use the ellipsoid algorithm. We have designed an approximation algorithm for the general coloring problem that provides a coloring using at most one color more than the optimal number of colors needed. There is no known practical algorithm for the coloring problem on tolerance graphs.

We have studied problems on other related special classes of graphs. We have results on the classes of *directed path graphs* [Nar89] and *trapezoidal graphs* [Nar89a]. We have not included them in this report.

The second approach that we used was to employ algebraic techniques to tackle hard problems. Our most significant results were obtained when we applied algebraic techniques to estimate the size of the maximum k-colorable subgraph ($\alpha_k(G)$). We obtained inequalities that bound $\alpha_k(G)$ and other graph parameters. The interesting feature of these inequalities was that it was in the form of a *sandwich theorem.* A sandwich theorem is an inequality that sandwiches a polynomial-time computable function between two hard to compute parameters. Sandwich theorems become useful whenever the inequalities are satisfied with equality for interesting cases. Lovász had used one such sandwich theorem to design polynomial-time algorithms for the class of perfect graphs. Our sandwich theorem helps us to design polynomial-time algorithms to compute $\alpha_k(G)$ and other graph parameters for a restricted family of graphs.

Wilf's inequality seemed to imply that the largest eigenvalue of the adjacency matrix gives some information about the largest clique in the graph. We wanted to explore whether the second largest eigenvalue of the adjacency matrix gives information about the second largest clique of the graph. We generalized Wilf's result and proved that the sum of the two largest eigenvalues is an upper bound on the size of the largest subgraph that can be covered with two cliques. The generalized sandwich theorem indicates strongly that the sum of k largest eigenvalues of the adjacency matrix provides information about the largest subgraph that can be covered with k cliques. Following studies due to Lovász, we also study matrices related to the adjacency matrix. It is generally believed that the adjacency matrix does not sufficiently reflect information about structures (cliques, for example) in the graph. Hence, the adjacency matrix has to be 'tuned' in order to extract the necessary information.

We are studying the question of whether the generalized sandwich theorem helps to design polynomial-time algorithms to compute graph parameters for any naturally occurring class of graphs. One important question that has not been resolved in this dissertation is whether the generalized sandwich theorem holds if we replace the k-multichromatic number $\chi_k(G)$ by $\psi_k(G)$ (See section 4.4.3 for definition). The parameter $\psi_k(G)$ is related to $\chi_k(G)$, but has some desirable properties. An affirmative answer to the above question would give a polynomial-time algorithm to compute the size of the largest k-colorable subgraph for complements of a class of graphs known as the *k-perfect graphs.*

It is known that the function $\vartheta(G)$ used by Lovász in his sandwich theorem has integral values for all perfect graphs. A related question is to determine whether the functions $\vartheta_k(G)$ defined in the generalized sandwich theorem are integral for any interesting classes of graphs.

The algebraic properties of naturally occurring special classes of graphs have not been studied. Such a study could help design more efficient algorithms for special classes of graphs.

Designing approximation algorithms was the next approach we took. We prove that no good approximation algorithms can be designed for the maximum k-colorable subgraph problem on general graphs. However, we designed good approximation algorithms for the general coloring problem on tolerance graphs. Another good approximation algorithm that we designed for the maximum k-colorable subgraph problem runs in polynomial time for any class of graphs for which there exists a polynomial-time algorithm to compute the maximum independent set.

We believe that the algebraic methods used hold a lot of promise and deserve further study.

References

[Ber61] Berge, Claude, Farbung von Graphen, deren sämtliche bzw., deren ungerade Kreise starr sind, *Wiss. Z. Martin-Luther-Univ., Halle-Wittenberg Math.-Natur,* Reihe, 1961, pp. 114-115.

[Ber62] Berge, Claude, Sur une conjecture relative au problème des codes optimaux, *Comm. 13ieme Assemblee Gen. URSI,* Tokyo, 1962.

[B74] Biggs, N., *Algebraic Graph Theory,* Cambridge Univ. Press, Cambridge, 1971.

[Bol85] Bollobás, B., *Random Graphs,* Academic Press, London, 1985.

[Bou87] Bouchet, A., Reducing Prime Graphs and Recognizing Circle Graphs, *Combinatorica,* **7** (1986), pp. 243-254.

[BL76] Booth, K. S., and G. S. Lueker, Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms, *J. Comput. System Sci.,* **13** (1976), pp. 335-379.

[CD80] Conte, S. D., and C. de Boor, *Elementary Numerical Analysis - An Algorithmic Approach,* 3rd edition, McGraw Hill Book Company, 1980.

[Cook71] Cook, S. A., The Complexity of Theorem-Proving Procedures, Proc. 3rd Ann. ACM STOC, New York, 1971, pp. 151-158.

[CDS80] Cvetković, D. M., M. Doob, H. Sachs, *Spectra of Graphs: Theory and Application,* Academic Press, New York, 1979.

[CNS81] Chiba, N., T. Nishizeki, and N. Saito, A Linear 5-coloring Algorithm of Planar Graphs, *J. of Algorithms,* **2** (1981), pp. 317-327.

[DGP86] Dagan, I., Golumbic, M., and Pinter, R., Trapezoidal Graphs and their Coloring, Tech. Report No. 88.206, IBM Israel, November 1986.

[EI71] Even, S., and Itai, A., Queues, Stacks, and Graphs, in *Theory of Machines and Computations,* pp. 71-86, Academic Press, New York, 1971.

[Fan49] Fan, K., On a Theorem of Weyl Concerning Eigenvalues of Linear Transformations I, *Proc. Natl. Acad. Sci. USA,* **35** (1949), pp. 652-655.

[F80] Frank, András, On Chain and Antichain Families of a Partially Ordered Set, *J. Combin. Theory, Ser. B,* **29** (1980), pp. 176-184.

[F84] Frederickson, G. N., On Linear-Time Algorithms for Five-Coloring Planar Graphs, *Inform. Proc. Letters,* **19** (1984), pp. 219-224.

[Ga73] Gavril, F., Algorithms for a maximum clique and a maximum independent set of a Circle Graph, *Networks,* **3** (1973), pp. 261-273.

[Ga74] Gavril, F., Algorithms on Circular-Arc Graphs, *Networks,* **4** (1974), pp. 357-369.

[Go80] Golumbic, M. C., *Algorithmic Graph Theory And Perfect Graphs,* Academic Press, New York, 1980.

[Gr76] Greene, C., Some Partitions Associated with a Partially Ordered Set, *J. Combin. Theory, Series A,* **20** (1976), pp. 69-79.

[GH64] Gilmore, P., and Hoffman, A., A Characterization of Comparability Graphs and Interval Graphs, *Canad. J. Math.,* **16** (1964), pp. 539-548.

[GH86] Golumbic, M. C., and Hammer, P.L., Stability in Circular-Arc Graphs, Tech. Report No. 88.196, IBM Israel, August 1986.

[GHS85] Gabor, C. P., W. -L. Hsu and K. J. Supowit, Recognizing circle graphs in polynomial time, *Proc. 26th FOCS,* (1985), pp. 106-116.

[GJ79] Garey, M. R., and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W. H. Freeman And Company, New York, 1979.

[GJMP80] Garey, M. R., D. S. Johnson, G. L. Miller, and C. H. Papadimitriou, The complexity of coloring circular-arcs and chords, *SIAM J. Alg. and Disc. Methods,* **1** (1980), pp. 216-227.

[GK76] Greene, C., D. Kleitman, The Structure of Sperner k-families, *J. Combin. Theory, Series A,* **20** (1976), pp. 41-68.

[GLL82] Gupta, U. I., D. T. Lee and Y. -T. Leung, Efficient algorithms for interval graphs and circular-arc graphs, *Networks,* **12** (1982), pp. 459-467.

[GLS81] Grötshcel, M., L. Lovász, and A. Schrijver, The Ellipsoid Method and its Consequences in Combinatorial Optimization, *Combinatorica,* **1** (1981), pp. 169-197.

[GM82] Golumbic, M. C., and Monma, C., A Generalization of Interval Graphs with Tolerances, *Proc. 13th SouthEastern Conf. on Combinatorics, Graph Theory and Computing, Congressus Numerantium, Utilitas Math., Winnipeg,* **35** (1982), pp. 321-331.

[GMT84] Golumbic, M. C., Monma, C., and Trotter, W., Tolerance Graphs, *Disc. Appl. Math.,* **9** (1984), pp. 157-170.

[Har69] Harary, F., *Graph Theory,* Addison Wesley, Reading, Mass., 1969.

[Joh85] Johnson, D. S., The NP-Completeness Column: An Ongoing Guide, *J. of Algorithms,* **6** (1985), pp. 434-451.

[JN76] Judin, D. B., and A. S. Nemirovskii, Informational Complexity and effective methods for the solution of convex extremal problems, *Ekonomika; Matematicheskie Metody,* **12** (1976), pp. 357-369.

[Karp72] Karp, R. M., Reducibility among Combinatorial Problems, in *Com[;exity of Computer Computations,* eds. R. E. Miller and J. W. Thatcher, Plenum Press, New York, 1972, pp. 85-103.

[Kha79] Khachiyan, L. G., A polynomial algorithm for linear programming, *Soviet Math. Dokl.,* **20** (1979), pp. 191-194.

[Lov72] Lovász, L., A Characterization of Perfect Graphs, *J. Combin. Theory, Series B,* **13** (1972), pp. 95-98.

[Lov79] Lovász, L., On the Shannon Capacity of a Graph, *IEEE Trans. Inform. Theory,* **25** (1979), pp. 1-7.

[Lov83] Lovász, L., Perfect Graphs, in *Graph Theory, 2,* eds. L. W. Beineke, and R. J. Wilson, Academic Press, London, 1983.

[Lov86] Lovász, L., *An Algorithmic Theory of Numbers, Graphs and Convexity,* CBMS 50, SIAM, Philadelphia, 1986.

[LY80] Lewis, J. M., and M. Yannakakis, The node-deletion problem for hereditary properties is NP-Complete, *J. of Comp. System Sci.,* **20** (1980), pp. 219-230.

[Ma45] Marczewski, E., Sur seux propriétés des classes d'ensembles, *Fund. Math.,* **33** (1945), pp. 303-307.

[MM64] Marcus, M., and H. Minc, *A Survey of Matrix Theory and Matrix Inequalities,* Allyn and Bacon, Inc., Boston, 1964.

[MN88] Masuda, S., and Nakajima, K., An Optimal Algorithm for Finding a Maximum Indpendent Set of a Circular-Arc Graph, *SIAM J. Comput.,* **17** (1988), pp. 41-52.

[MS84] Marek-Sadowska, M., An unconstrained topological via minimization problem for two-layer routing, *IEEE Trans. on Computer-Aided Design,* **3** (1984), pp. 184-190.

[Nar89] Narasimhan, G., A Note on the Hamiltonian Circuit Problem on Directed Path Graphs, To appear in *Information Processing Letters.*

[Nar89a] Narasimhan, G., Algorithms on Trapezoidal Graphs, Unpublished results.

[ND77] Noble, B., and J. Daniel, *Applied Linear Algebra,* 2nd edition, Prentice Hall, Inc., NJ 07632, 1977.

[NMN87] Naclerio, N. J., Masuda, S., and Nakajima, K., Via Minimization for Gridless Layouts, *24th ACM/IEEE Design Automation Conference,* (1987), pp. 159-165.

[PS82] Papdimitriou, C. H., and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity,* Prentice Hall, New Jersey, 1982.

[Sho77] Shor, N. Z., Cut-off method with space extension in convex programming problems, *Cybernetics,* **13** (1977), pp. 94-96.

[SL87] Sarrafzadeh, M., and Lee, D.T., A New Approach to Topological Via Minimization, Tech. Report, Center for Integrated Microelectronic Systems, NorthWestern Univ., November 1987.

[Tar72] Tarjan, R., Sorting Using Networks of Queues and Stacks, *JACM,* **19** (1972), pp. 341-356.

[Tuc71] Tucker, A., Matrix Characterizations of Circular-Arc Graphs, *Pacific J. Math,* **39** (1971), pp. 535-545.

[Tuc75] Tucker, A., Coloring a family of circular-arcs, *SIAM J. Appl. Math.,* **29** (1975), pp. 493-502.

[Tuc80] Tucker, A., An efficient test for circular-arc graphs, *SIAM J. Comput.,* **9** (1980), pp. 1-24.

[Wil67] Wilf, H. S., The Eigenvalues of a Graph and its Chromatic Number, *J. of London Math. Soc.,* **42** (1967), pp. 330-332.

[YG87] Yannakakis, M., and F. Gavril, The Maximum k–Colorable Subgraph Problem for Chordal Graphs, *Inform. Proc. Letters,* **24** (1987), pp. 133-137.

Appendix

Table of Relevant Results

GRAPH CLASS	MEMBER	MIS	CLIQUE	CLIPAR
Perfect	Open	P [GLS81]	P [GLS81]	P [GLS81]
Interval	P [BL76]	P [GLL82]	P [GLL82]	P [GLL82]
Circular-Arc	P [Tuc80]	P [GLL82]	P [Hsu85]	P [GLL82]
Circle	P [GHS85]	P [Ga74]	P [Hsu85]	Open
Tolerance	Open	P [N*]	P [N*]	P [GLS81]
Trapezoidal	Open	P [Nar89a]	P [GLS81]	P [GLS81]

GRAPH CLASS	CHRNUM	MIBS	MkCS (fixed k)	MkCS (arb. k)
Perfect	P [GLS81]	Open	Open	N [YG87]
Interval	P [GLL82]	P [N*]	P [YG87]	P [YG87]
Circular-Arc	N [GJMP80]	P [N*]	Open	N [GJMP80]
Circle	N [GJMP80]	N [SL87]	N [SL87]	N [GJMP80]
Tolerance	P [GLS81]	P [N*]	P [N*]	P [GLS81]
Trapezoidal	P [GLS81]	Open	Open	Open

The above chart tabulates complexity results for eight graph problems when inputs are restricted to the classes of graphs mentioned in the first column titled GRAPH CLASS. MEMBER is the recognition problem. MIS is the problem of computing the maximum independent set. CLIQUE is the problem of finding the maximum clique. CLIPAR is the problem of finding the smallest clique cover that covers all the vertices. CHRNUM is the general graph coloring problem. MIBS is the maximum induced bipartite subgraph problem. The maximum k-colorable subgraph problem for fixed k and arbitrary k are the problems in the last two columns. P stands for polynomial-time solvable. N stands for NP-hard and Open means that the problem is open. Reference [N*] refers to results in this thesis.

Some Definitions

Permutation Graph:

If π is a permutation of the numbers $1, 2, \ldots, n$, then the graph $G[\pi] = (V, E)$ is defined as follows: $V = \{1, 2, \ldots, n\}$ and $(i, j) \in E \Leftrightarrow (i - j)(\pi_i^{-1} - \pi_j^{-1}) < 0$. An undirected graph is called a *permutation graph* if there exists a permutation π for which G is isomorphic to $G[\pi]$.

Perfect Graph:

A graph is perfect if and only if the chromatic number equals the clique number for every induced subgraph of G.

Planar Graph:

A graph is planar if the graph can be embedded on a plane surface.