INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Results on Perfect Graphs

by

Mark David Timothy Petrick

A thesis
presented to Lakehead University
in fulfilment of the
thesis requirement for the degree of
Master of Science
in
Computer Science

Thunder Bay, Ontario, Canada, 2000

Abstract

The chromatic number of a graph G is the least number of colours that can be assigned to the vertices of G such that two adjacent vertices are assigned different colours. The clique number of a graph G is the size of the largest clique that is an induced subgraph of G. The notion of perfect graphs was first introduced by Claude Berge in 1960. He defined a graph G to be *perfect* if the chromatic number of H is equal to the clique number of H for every induced subgraph $H \subseteq G$. He also conjectured that perfect graphs are exactly the class of graphs with no induced odd hole (a chordless odd cycle of greater than or equal to five vertices) or no induced complement of an odd hole, an odd anti-hole. This conjecture, that still remains an open problem, is better known as the *Strong Perfect Graph Conjecture* (or SPGC). An equivalent statement to SPGC is that minimal imperfect graphs are odd holes and odd anti-holes.

Fonlupt conjectured that all minimal imperfect graphs with a minimal cutset that is the union of more than one disjoint clique, must be an odd hole. In this thesis we prove that any hole-free graph G with a minimal custet C that is the union of vertex-disjoint cliques must have a clique in each component of $G - C$ that sees all of C. We further prove that minimal imperfect graphs with a minimal cutset that is the union of two disjoint cliques have a hole.

Since the introduction of perfectly orderable graphs by Chvátal in 1984, many classes of perfectly orderable graphs and their recognition algorithms have been identified. Perfectly ordered graphs are those graphs G such that for each induced ordered subgraph H of G, the greedy (or, sequential) colouring algorithm produces an optimal colouring of H. Hoàng and Reed previously studied six natural subclasses of perfectly orderable graphs that are defined by the orientations of the P_4's. Four of the six classes can be recognized in polynomial time. The recognition problem for the fifth class has been proven to be NP-complete. In this thesis, we discuss the problem of recognition for the

sixth class, known as one-in-one-out graphs. Also, we consider pyramid-free graphs with the same orientation as one-in-one-out graphs and prove that this class of graphs cannot contain a directed 3-cycle of more than one equivalence class.

Acknowledgements

First, I would like to thank my supervisor, Dr. Chính Hoàng, for his insights, patience, and guidance during the preparation of this thesis. I am indebted to him for all of his time and effort over the years at Lakehead University.

I would also like to thank Dr. Elaine Eschen and Dr. R. Sritharan. The thoughts, observations, and discussions in Florida, Toronto, and Thunder Bay contributed greatly towards this work. I would also like to thank Dr. Wendy Huang for her helpful comments and suggestions as a reader.

I would like to acknowledge Lakehead University and the Natural Sciences and Engineering Research Council for their financial support.

I am grateful to my girlfriend for her constant encouragement, motivation and understanding during the writing of this thesis. Thanks Rebecca!

I am also very thankful for my friends from Lakehead University, Brian, Chris, and Shawn to only name a few, and the members of TBISC, especially Jeff, Hal, Rick and Karen for their kindness and friendship.

Finally, I would like to express a special thank you to my parents, my brother Ron, and my aunts for all their love and support.

Contents

1 Introduction 1

1.1 Motivation 3

1.2 Organization of the Thesis 4

2 Background 6

2.1 Sequential Graph Colouring 6

2.2 Bounds on the Colouring Problem 9

2.3 Perfectly Orderable Graphs 10

2.4 Recognizing Perfectly Orderable Graphs 13

2.5 Classical Perfectly Orderable Graphs 14

2.5.1 Comparability Graphs 14

2.5.2 Triangulated Graphs 15

2.5.3 Interval Graphs 16

2.6 Orientations 17

2.6.1 P_4-Simplicial Graphs 18

2.6.2 Generalization of Comparability and Bipolarizable Graphs . . . 19

2.6.3 Bipolarizable Graphs 19

2.6.4 P_4-Indifference Graphs 21

2.6.5 P_4-Comparability Graphs 22

2.7 Minimal Non Perfectly Orderable Graphs 22
2.8 More Classes of Perfectly Orderable Graphs 24
2.8.1 Complements of Tolerance Graphs 24
2.8.2 Charming Graphs 24
2.8.3 Nice Graphs 25
2.8.4 Brittle Graphs 25

3 One-In-One-Out Graphs **27**
3.1 Finding a Recognition Algorithm 29
3.2 Graph Generation 31
3.3 Client/Server Model for Graph Generation 33
3.4 A Subclass of One-In-One-Out Graphs 34

4 Minimal Imperfect Graphs **52**
4.1 Structure of Minimal Imperfect Graphs 53
4.2 Connectivity of Minimal Imperfect Graphs 54
4.3 Hole-free Minimal Imperfect Graphs 58

5 Conclusions and Future Research **65**
5.1 One-In-One-Out Graphs 65
5.2 Graph Generation 66
5.3 Minimal Imperfect Graphs 66

Bibliography **68**

List of Tables

2.1 Six classes of perfectly orderable graphs. 18

List of Figures

2.1 A list of obstructions 11
2.2 a, b, c, d form an obstruction 13
2.3 P_3 with $a < b, b < c$ 14
2.4 Transitive orientation 15
2.5 P_3 with $a < b, c < b$ 15
2.6 Simplicial orientation 16
2.7 An obstruction 17
2.8 Three types of oriented P_4's 17
2.9 The clique wheels $CW(2)$ and $CW(4)$ 20
2.10 A list of minimal non bipolarizable graphs 20
2.11 A list of minimal non P_4-indifference graphs 21
2.12 A list of some MNPO graphs 23
2.13 House and Domino graphs 26
2.14 The list of all minimal non supperbrittle graphs 26

3.1 Generalization of Type 2 and Type 3 P_4's 28
3.2 A graph oriented with Type 2 and Type 3 P_4's containing a directed cycle 30
3.3 Connect the two 3-cycles to join the equivalence classes 31
3.4 Connect the two graphs to join the equivalence classes 31

3.5 A pyramid 35
3.6 Graphs forbidden in Type 2 and Type 3 orientations 35
3.7 Some possible configurations for vertices $\{d, e, f, g, h, i\}$ 36
3.8 The P_4 on $\{a, b, h, i\}$ shares an edge with the house 37
3.9 vertices h, i are not part of the house 38
3.10 if $e = f$ and $d \neq g$ then the P_4 $aceg$ has a forbidden orientation 39
3.11 ab, bc are not forced from the same wing of a P_4 40
3.12 if $d = g$ and $e \neq f$ then $\{a, b, c, e, d\}$ induce a house 41
3.13 if $e = g$ and $d \neq f$ then the P_4 $bafe$ has a forbidden orientation 41
3.14 if $d = g$ then $\{a, b, c, e, g\}$ induces a house 42
3.15 if $e = g$ then the P_4 $edcb$ has a forbidden orientation 42
3.16 Two cases where each P_4 is unique 43
3.17 The Symmetric Case 44
3.18 $bh \in E$ 45
3.19 $cd \in E$ 46
3.20 $af \in E$ 47
3.21 ge cannot be an edge 49
3.22 gi cannot be an edge 50
3.23 ei cannot be an edge 51

4.1 Initial configuration 56
4.2 y must see all of $C - K_1$ 57
4.3 Regardless if wy is an edge, there is a hole 57
4.4 x is adjacent to the most vertices in K_1 59
4.5 A hole exists in both Case 1 and Case 2 60
4.6 A hole is created if $a \in K_2$ misses x 61

4.7 K_1 cannot be size ω 62
4.8 P and a common neighbour to x and y form a hole 64

Chapter 1

Introduction

Since first being introduced sometime in the last century, finding an efficient method to optimally colour a graph has attracted the interests of many mathematicians. The *graph colouring problem* is to assign a colour to each vertex of a graph in such a manner that no two adjacent vertices are assigned the same colour and that the minimum number of colours are used. Also, one of the most famous problems in graph theory deals with graph colouring. It is best known as the *Four Colour Problem*. Given a map of different countries whose borders meet to form connected regions, can we colour the regions using no more than four colours so that countries that share a common border have different colours? Other applications of graph colouring include Very Large Scale Integration (VLSI) problems and various scheduling problems such as determinng the minimum number of classrooms necessary for a school's timetable of classes. Many significant results and new applications have been obtained in this field; however, the graph colouring problem still manages to be one of the most intractable problems in discrete mathematics.

A formal graph theoretic definition for the graph colouring problems is as follows:

Definition 1 *A graph $G = (V, E)$ is k-colourable if and only if there is an assignment of k colours to its vertices such that two vertices receive different colours if they are adjacent. The graph colouring problem is then to find the smallest number k for which the graph is k-colourable. This number is also know as the chromatic number of G denoted $\chi(G)$.*

The initial formulation of the four colour problem occurred when cartographers were asked to colour a political map using a minimum set of colours in such a way that no two adjacent countries were given the same colour. The graph $G = (V, E)$ is used to represent the whole map. Each country, is assigned a unique vertex in V and if two countries, $u, v \in V$, share a common border, then uv is an edge in G, $(uv \in E)$. It was then predicted, that only four colours would be required to completely colour any such map. Hence the name the four colour problem. The four colour problem was later solved by Appel and Haken [AH77a, AH77b].

Another common application of graph colouring is used in various scheduling problems. For example, consider a timetable of classes. Each class is assigned a specific room and if two classes occur at the same time, they must be given different rooms. By using graph colouring and finding the chromatic number, we can then determine the minimum number of rooms required for the timetable. We create a graph to represent the timetable by assigning each course or class to a separate vertex of the graph. Two vertices are then joined by an edge if and only if the two courses occur at the same time. Hence, adjacent vertices in the graph represent courses that cannot share the same room. So, when the graph is coloured these vertices will be assigned different colours to represent that they will require different classrooms (the colours represent the classrooms). Such a graph is called an *interval graph.*

1.1 Motivation

The applications of graph colouring and finding the chromatic number for various graphs are very extensive. In the examples previously mentioned, we did not specify a method for obtaining the result and only said that the result would be useful to solve the problem. If the graph that represents our problem is small enough, it is easy for us to assign a colour to a vertex and then a different one to the neighbours of that vertex and progress through the graph assigning colours in this fashion. We may need to go back and make modifications to the colours if we run into trouble, however, with a little bit of time we should be able to come up with an optimal colouring. This method of trial and error may work for small graphs, but what do we do when the graph contains hundreds of vertices? We may be able to colour a graph such as this, but, will we then be able to determine if we have an optimal colouring? Hence, a formal method for graph colouring and determining when a colouring is optimal is highly desirable. Unfortunately, in many cases, determining the chromatic number of a graph is an NP-hard problem. However, in some cases, such as the class of interval graphs, it can be done in polynomial time [Gol80].

Being able to determine the chromatic number in polynomial time is certainly beneficial, however, it raises another question. Can we determine if the graph is an interval graph so that the polynomial time algorithm to determine the chromatic number can be applied? This requires the development of another algorithm by which to recognize the class of graphs. In the case of interval graphs, a polynomial time recognition algorithm has also been developed [BL76]. So, for interval graphs, we can both recognize them and determine the chromatic number in polynomial time. In other cases, although we may have found an optimal colouring algorithm that runs in polynomial time, the recognition algorithm may be NP-complete. In this situation, determining the chromatic number

would also be NP-hard if we need to recognize the graph first and then find the optimal colouring. This gives rise to the notion of a robust algorithm introduced by Spinrad.

Many algorithms are created to work specifically on a certain type or class of graphs. If the algorithm is then executed on a graph not from the specified class, the algorithm may not realize it has bad input and the result may not be correct. The idea of a robust algorithm is to have an algorithm that will return a useful result regardless of input. So, in the case where the input is not part of the specified class, the algorithm should indicate that the input is bad; otherwise, the algorithm will return a correct result. In the case of interval graphs, we can create a robust polynomial time algorithm by combining the recognition algorithm and the optimal colouring algorithm. In the case where there is a polynomial time algorithm for finding an optimal colouring but an exponential time algorithm for recognizing it's input class, the robust algorithm would also be exponential. Hence, finding an algorithm to optimally colour a graph in polynomial time may not be as useful as it appears if we cannot also recognize the class of graphs for which the algorithm returns a correct result in polynomial time.

1.2 Organization of the Thesis

The remainder of the thesis focuses on perfectly orderable graphs and minimal imperfect graphs.

In Chapter 2, the graph colouring problem is further discussed and bounds on the number of colours of an optimal colouring are given. Both upper and lower bounds are discussed for arbitrary graphs and further refinements to these bounds are given for when the greedy colouring algorithm is used. The notion of perfectly orderable graphs is introduced along with some examples of classical perfectly orderable graphs. Orientations, analogous to graph orders, are then introduced along with six more classes of graphs that

are made from the combinations of valid P_4 orientations. This chapter is completed by looking at some properties of minimal non perfectly orderable graphs and then by briefly overviewing a few more classes of perfectly orderable graphs.

Chapter 3 focuses on the finding a recognition algorithm for the class of one-in-one-out graphs which use orientations rather than graph orders. The main problem is to determine whether a recognition algorithm can be found in polynomial time or if the problem is NP-complete. A method for finding a graph that is orientated in the same manner as one-in-one-out graphs but does not contain an acyclic orientation is given. Finding such a graph would indicate that the problem is NP-complete. Graph generation is also briefly touched on and a client/server model for a traditional method of graph generation is given. To solve, perhaps, a simpler problem than that on one-in-one-out graphs, pyramid-free graphs with the same orientation types as one-in-one-out graphs is defined. It is then shown that this subclass of one-in-one-out graphs cannot contain a directed 3-cycle of more than one equivalence class.

Chapter 4 focuses on hole-free minimal imperfect graphs. The first two sections of this chapter give some background and properties on the structure and connectivity of minimal imperfect graphs. In the second section, we prove a theorem about the components of hole-free graphs that have a minimal cutset that is the union of vertex-disjoint cliques. The main result, that makes use of the before mentioned theorem is then introduced and developed. This theorem proves that hole-free minimal imperfect graphs cannot have a minimal cutset that is the union of two disjoint cliques.

Chapter 5 reviews the main results of this thesis and discusses future work and open problems.

Chapter 2

Background

In this chapter, a more detailed look at a common graph colouring algorithm is given. We focus on the sequential (or greedy) graph colouring approach followed by an indepth study of perfectly orderable graphs.

2.1 Sequential Graph Colouring

Sequential (or greedy) graph colouring is a natural way to colour a graph. The first step is to impose an order $<$ on the vertices v_i of a graph and then to progress through the vertices in the assigned order. If $v_i < v_j$ then v_i appears before v_j in the assigned ordering. At each vertex v_i we look at all of the neighbours v_j of v_i such that $v_j < v_i$ and assign v_i the lowest possible colour not already assigned to one of these neighbours. This method is called the *greedy algorithm*. It is important to realize that the greedy algorithm does not guarantee that the colouring it produces is an optimal colouring (i.e., one using the smallest possible number of colours). The greedy colouring algorithm is also sometimes called a *colouring function*. A formal definition of a colouring function is as follows:

Definition 2 *A function f determines a colouring of a graph $G = (V, E)$, if*

$$f : V \longrightarrow \mathcal{S}, \text{ where } \mathcal{S} = \{1, 2, \ldots\} \text{ is a set of colours}$$

and

$$f(i) \neq f(j) \text{ for all } (i, j) \in E.$$

A colouring where $|\mathcal{S}| = k$ is called a k-colouring and a function that defines a k-colouring is called a k-colouring function.

From the above sequential approach, it is easy to determine an upper bound on the number of colours that the algorithm will use. Each vertex v_i can be assigned colour i since there are at most $i - 1$ vertices that have already been assigned colours. Hence, if they are all neighbours of v_i and were assigned different colours, colour i would be assigned to v_i (i.e., $f(v_i) \leq i$). It is quite probable, however, that not all of the previously coloured vertices are neighbours of v_i. In fact, v_i has at most $deg(v_i)$ (the degree of v_i) neighbours that have already been assigned colours. So, v_i can be coloured with at least one of the first $deg(v_i) + 1$ colours. Therefore, for each vertex v_i,

$$f(v_i) \leq \min\{i, deg(v_i) + 1\}.$$

To determine the number of colours used by the sequential algorithm, we must then take the largest colour assigned to any of the vertices. We will use $\chi_s(G)$ to denote the number of colours used by the sequential algorithm on graph G.

$$\chi_s(G) \leq \max_{1 \leq i \leq n} \min\{i, deg(v_i) + 1\}$$

Now, this upper bound can be refined further if we consider the possibility that not every neighbour of vertex v_i will precede it in the ordering. Hence, the upper bound on v_i is the degree of v_i in the subgraph of G induced by $v_1, v_2, \ldots, v_i$ denoted $deg_{G_i}(v_i)$. So, our upper bound is

$$\chi_s(G) \leq \max_{1 \leq i \leq n} deg_{G_i}(v_i) + 1$$

.

In the above discussion of greedy colouring, we determined an upperbound on the number of colours that the algoirthm would use. In doing so, we did not constrain the ordering of the vertices, but instead, we let the ordering be arbitrary. Welsh and Powell ([WP67]) proposed ordering the vertices in nonincreasing order of degree. This would help minimize the upperbound since for a vertex v_i where i is small, the degree would be high but the vertex would be given a small colour due to its position in the ordering. Similarly, for vertices later in the ordering, a small colour would be assigned because their degree would be low. This ordering is known as the *largest first* (or LF) ordering.

Similar to the method we used to refine the upper bound, we can refine the way in which we order the vertices. We start by looking at all of the vertices of G and choosing the vertex of lowest degree. We label this vertex v_n so it will appear last in the ordering. Next we look at the subgraph of G induced by $V - \{v_n\}$ and choose the vertex with lowest degree. We then label this vertex v_{n-1}. We continue to look at the induced subgraph of all unlabelled vertices and choose the vertex with smallest degree to label until we run out of vertices. This ordering is known as the *smallest last* (or SL) ordering.

Although making use of either the LF or SL orderings may improve the upper bound on the greedy colouring algorithm, it still does not guarantee that an optimal colouring will be obtained for an arbitrary graph G.

2.2 Bounds on the Colouring Problem

In the previous section we looked at the upper bound on the number of colours the greedy algorithm uses for colouring graphs. In this section we will continue to look at both upper and lower bounds for colouring.

The colouring problem is to find the least k for which the graph G has a k-colouring. Although the greedy algorithm does colour the graph, there is no guarantee that the colouring is an optimal colouring; so, a more complex algorithm would be necessary. In fact, Garey and Johnson ([GJ79]) proved that the graph colouring problem belongs to the class of NP-hard problems. So deriving a polynomial time algorithm for graph colouring seems impossible. Also, no formula has been derived that explicitly determines the chromatic number by looking at the graph. However, we do have some lower and upper bounds on the problem that will limit the possibilities.

Lower Bound: $\chi(G) \geq \omega(G)$

$\omega(G)$ denotes the size of the largest clique in G. If you consider a clique of size n, each vertex in the clique has the other $n-1$ vertices as neighbours. Hence, each vertex in the clique must be assigned different colours in any colouring of the graph.

Lower Bound: $\chi(G) \geq n(G)/\alpha(G)$

$n(G)$ denotes the number of vertices in G and $\alpha(G)$ denotes the size of the largest independent set in G. An *independent set* or a *stable set* in a graph G is a subset of the vertices of G whose induced subgraph has no edges. Here we can see that all vertices in an independent set could be assigned the same colour.

Upper Bound: $\chi(G) \leq n(G)$

As a worst case scenario, each vertex of the graph could be assigned its own unique colour.

Upper Bound: $\chi(G) \leq \delta(G) + 1$

$\delta(G)$ is the maximum degree of the vertices of G. As seen with the greedy algorithm, we can assign a vertex the smallest colour not assigned to any of its neighbours. So, for the vertex of maximum degree, it could potentially be assigned the colour $\delta(G)+1$. This bound has been refined further in the previous section.

All of these bounds are easy, but they are also tight. Again, consider a graph G that is a clique of size n. We already showed that $\chi(G) = \omega(G)$. For the second lower bound, since all vertices see[1] each other, the largest independent set is of size 1. Hence, $\chi(G) = n(G)/\alpha(G)$ and trivially, with $\alpha(G) = 1$, $\chi(G) = n(G)$. Finally, it is also obvious that $\chi(G) = \delta(G) + 1$. So all of our bounds are equality when the graph is a clique.

Having these bounds and the greedy algorithm for colouring graphs, the next logical question is: can we find a class of graphs for which the greedy algorithm produces an optimal colouring? In order to solve this question, V. Chvátal proposed the concept of perfectly orderable graphs.

2.3 Perfectly Orderable Graphs

The idea behind perfectly ordered graphs is to impose an order $<$ on the vertices of a graph and then to use the greedy algorithm to colour them based on this order.

Definition 3 *For an ordered graph $(G, <)$, the ordering $<$ is called* perfect *if for each induced ordered subgraph $(H, <)$ the greedy algorithm produces an optimal colouring of H.*

Furthermore,

[1]In a simple graph $G = (V, E)$, for $a, b \in V$, a sees b if and only if a is adjacent to b.

Definition 4 *An ordered graph* $(G, <)$, *that admits a perfect ordering* $<$, *is called* perfectly orderable.

In [Chv84], Chvátal also showed that some very well known classes of graphs are perfectly orderable; namely, comparability, triangulated and the complements of triangulated graphs. As well, he defined an *obstruction* to a perfect ordering.

Definition 5 ([Chv84]) *An* obstruction *is an ordered graph* $(G, <)$ *with vertices* a, b, c, d *and edges* ab, bc, cd *such that* $a < b$ *and* $d < c$. *These three structures can be seen in Figure 2.1.*

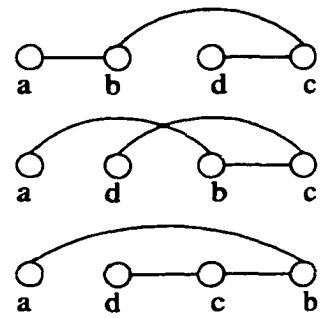


Figure 2.1: A list of obstructions

Chvátal then proved a major result.

Theorem 1 ([Chv84]) *For an ordered graph* $(G = (V, E), <)$, *the ordering* $<$ *is perfect if and only if it is obstruction-free.*

Proof: the "only if" part is easy. In order to prove the "if" part, we need to show that the greedy algorithm will produce an optimal colouring if the ordering is obstruction-free. We can do this by comparing the number of colours used by the greedy algorithm with the clique size found in the graph. If these numbers are the same, we will know that

the colouring is optimal since the clique size is a lower bound. Also, since the ordering is hereditary, this proof will hold for all induced subgraphs $(H, <)$ of $(G, <)$. To make our work easier, we first prove the following lemma.

Lemma 1 *Suppose that G has a clique Q, a stable set S disjoint from Q, and each $q \in Q$ is adjacent to some $s(q) \in S$. If $<$ is an obstruction-free ordering of G such that $s(q) < q$ for all $q \in Q$, then some $s(q) \in S$ is adjacent to all vertices of Q.*

We shall illustrate the proof with directed edges. Given an ordered graph $(G, <)$, there is a corresponding *orientation* $D(G, <)$ of G such that $\vec{ab} \in D(G, <)$ if and only if $ab \in E(G)$ and $a < b \in (G, <)$. An obstruction is the directed graph shown in Figure 2.7.

Proof: To prove this lemma we will use induction on the number of vertices in Q,$|Q|$. For the basis, suppose $|Q| = 1$. Then trivially, $q_1 \in Q$ has a neighbour $s(q_1) \in S$ that is adjacent to all the vertices of Q. So the basis is true. The induction hypothesis then states that if $|Q| = k$ and each $q \in Q$ is adjacent to some $s(q)$ in the stable set S disjoint from Q then if $<$ is an obstruction-free ordering of G such that $s(q) < q, \forall q \in Q$, then $\exists s(q) \in S$ such that $s(q)$ is adjacent to all of Q. Now we must show that the lemma is also true for $|Q| = k + 1$. From the induction hypothesis, we are guaranteed for each $q \in Q$, that $\exists q^* \in Q$ such that $s(q^*)$ is adjacent to all vertices of Q except possibly q. If any $s(q^*)$ is adjacent to q then we are done since $s(q^*)$ would be adjacent to all of Q. So, suppose $s(q^*)$ is not adjacent to q. This creates a one-to-one and onto function mapping each q^* to q. Then, let v be the vertex in Q that comes first in the ordering and $b, c \in Q$, $(v < b, c)$, such that $b^* = v$ and $c^* = b$. Hence, v has a neighbour $s(v) \in S$ that is adjacent to all vertices in Q except b. Let $d = s(v)$, we have $d = s(v) < v \leq c$. Similarly, b has a neighbour $s(b) \in S$ that is adjacent to all vertices in Q except c. Let $a = s(b)$, we have $a = s(b) < b$. The vertices a, b, c, d then form an obstruction and

we have a contradiction (see Figure 2.2 and remembering that the edge uv with $u < v$ is directed from u to v). □

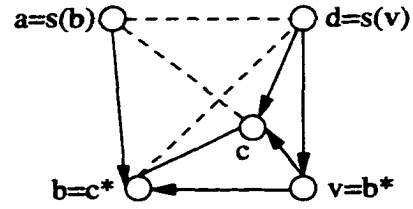


Figure 2.2: a, b, c, d form an obstruction

We can now begin the proof of Theorem 1. Let $f : V \rightarrow \mathcal{S}$, where $|\mathcal{S}| = k$, be the colouring function of the greedy algorithm on $(G, <)$. Consider the smallest i such that G has a clique of vertices $w_{i+1}, \ldots, w_k$ with $f(w_j) = j$ for all j with $i + 1 \leq j \leq k$. If $i = 0$ then we are done because the exists a clique of size k, namely, $w_1, \ldots, w_k$ so the colouring is optimal. Suppose $i > 0$. Then, each w_j has a neighbour $s(w_j)$ such that $s(w_j) < w_j$ and $f(s(w_j)) = i$; otherwise, a lower colour would be assigned to w_j. Lemma 1 then implies that there exists a vertex v such that $f(v) = i$ and that v is adjacent to all vertices w_j. This contradicts the minimality of i we chose at the beginning of the proof. □

Having now defined perfectly orderable graphs and understanding a bit of their structure, the natural question to ask is can we recognize them in polynomial time?

2.4 Recognizing Perfectly Orderable Graphs

Given an arbitrary graph G, can we determine whether the graph is perfectly orderable in polynomial time? Six years after Chvátal posed this question, Middendorf and Pfeiffer ([MP90]) proved that recognizing perfectly orderable graphs belongs to the class of NP-complete problems by using the following theorem.

Theorem 2 ([MP90]) *To decide whether a graph admits a perfect order is NP-complete.*

The proof shows a reduction of the 3SAT problem, which is known to be NP-complete. The 3SAT problem is reduced to the problem of deciding whether a graph admits an acyclic orientation such that for no induced path $P = p_1p_2p_3p_4$ of length 4, $p_1 < p_2$ and $p_4 < p_3$.

Because of this result, more emphasis was placed on finding classes of graphs that were perfectly orderable but could also be recognized in polynomial time.

2.5 Classical Perfectly Orderable Graphs

2.5.1 Comparability Graphs

Definition 6 *A graph G is a* comparability *graph if and only if it admits an order $<$ such that no chordless path with vertices a, b, c and edges ab, bc has $a < b, b < c$ (see Figure 2.3).*

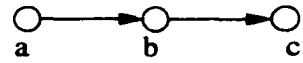


Figure 2.3: P_3 with $a < b, b < c$

From this definition, it is clear that no induced ordered subgraph of $(G, <)$ can be an obstruction since the obstruction would contain an induced subgraph isomorphic to Figure 2.3. The two valid P_3 orders are $a < b, c < b$ and $b < a, b < c$. Such an order is called a *transitive order*. Their oriented representations can be seen in Figure 2.4.

Golumbic designed a recognition algorithm for comparability graphs that runs in $O(n(G)m)$ time ([Gol84]). McConnell and Spinrad ([MS94, MS97]) later showed comparability graphs could be recognized in the same time as the complexity of matrix mul-

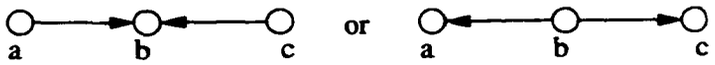


Figure 2.4: Transitive orientation

tiplication, currently $O(n^{2.376})$ ([CW90]) and Gallai ([Gal67]) found a subgraph characterization.

2.5.2 Triangulated Graphs

Triangulated graphs are called many things and can be defined in many ways. One definition is as follows:

Definition 7 *A graph G is called* triangulated *if every cycle of length strictly greater than three possesses a chord. Triangulated graphs are also known as* chordal, rigid-circuit, monotone transitive, *and* perfect elimination *graphs.*

Another definition of triangulated graphs is:

Definition 8 *A graph G is* triangulated *if and only if it admits an order $<$ such that no chordless path with vertices a, b, c and edges ab, bc has $a < b, c < b$ (see Figure 2.5).*

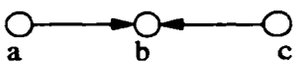


Figure 2.5: P_3 with $a < b, c < b$

From this definition, it is clear that no induced ordered subgraph of $(G, <)$ can be an obstruction since the obstruction would contain an induced subgraph isomorphic to Figure 2.5. The two valid P_3 orders are $a < b, b < c$ and $b < a, b < c$. Such an order is called a *simplicial order.* Their oriented representations can be seen in Figure 2.6.

A vertex is *simplicial* if its neighbourhood is a clique. Furthermore,

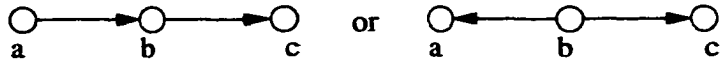


Figure 2.6: Simplicial orientation

Theorem 3 ([Dir61]) *Every triangulated graph G has a simplicial vertex. Moreover, if G is not a clique then it has two nonadjacent simplicial vertices.*

This theorem then prompted Fulkerson and Gross ([FG65]) to suggest an iterative procedure to recognize triangulated graphs based on the simplicial vertex and the hereditary property. The procedure would start with the whole graph and then look for the simplicial vertex. When found, remove this vertex from the graph, along with all of its incident edges and find another simplicial vertex and do the same. Continue until all the vertices are gone and conclude that the graph is triangulated or until no simplicial vertex can be found and the graph is not triangulated. Later, Rose, Tarjan, and Leuker ([RTL76]) used *lexicographic breadth-first search* to design a linear time algorithm for recognizing triangulated graphs.

2.5.3 Interval Graphs

Interval graphs can also be defined in many ways. Two definitions follow.

Definition 9 ([LB62]) *A graph G is an* interval *graph if and only if the following two conditions are satisfied:*

(i) G is a triangulated graph, and

(ii) any three vertices of G can be ordered in such a way that every path from the first vertex to the third vertex passes through a neighbour of the second vertex.

Definition 10 ([GH64]) *A graph G is an* interval *graph if and only if G is chordal and its complement $\overline{G}$ is a comparability graph.*

From the latter definition, interval graphs and their complements are obviously perfectly orderable since they are triangulated graphs and comparability graphs respectively. A linear time recognition algorithm was developed in [BL76] and a subgraph characterization was found in [LB62].

2.6 Orientations

Although orientations and orders are analogous, sometimes it is easier to work with orientations instead of orders.

Recall that given an ordered graph $(G, <)$, there is a corresponding *orientation* $D(G, <)$ of G such that $\vec{ab} \in D(G, <)$ if and only if $ab \in E(G)$ and $a < b \in (G, <)$. An acyclic orientation that does not contain an obstruction (see Figure 2.7) is called a *perfect orientation*. So a graph is perfectly orderable if and only if it admits a perfect orientation.

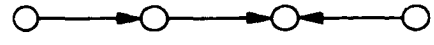


Figure 2.7: An obstruction

In a perfectly oriented graph, every P_4 must then be one of the three oriented P_4's in Figure 2.8.

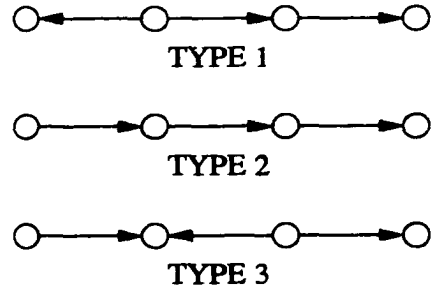


Figure 2.8: Three types of oriented P_4's

	P_4 allowed in G			Name	Recognition Complexity	Subgraph Characterization
	Type 1	Type 2	Type 3			
1	√	√		P_4-simplicial	$O(n^5)$ [HR89a]	NO
2	√		√	GCB	NP-Complete [Hoà96]	NO
3		√	√	One-in-one-out	Unknown	NO
4	√			Bipolarizable	$O(n^{3.376})$ [SJ]	YES
5		√		P_4-indifference	$O(n+m)$ [HPV]	YES
6			√	P_4-comparability	$O(m^2)$ [RS]	NO

Table 2.1: Six classes of perfectly orderable graphs.

By grouping these three types of P_4's we can create new classes of perfectly orderable graphs. Let $\mathcal{S}$ be a subset of $\{1,2,3\}$. An *$\mathcal{S}$-orientation* of a graph G is an acyclic orientation in which each $P_4 \in G$ is oriented in a manner of one of the types in $\mathcal{S}$. The set $\mathcal{S} = \{1,2,3\}$ corresponds to the class of all perfectly orderable graphs. The set $\mathcal{S} = \emptyset$ corresponds to the class of P_4-free graphs. Since, P_4-free graphs do not contain any P_4's, they cannot contain an obstruction and are therefore, trivially perfectly orderable. In [CPS85] a linear time recognition algorithm is given. The other six classes of graphs were studied by Hoàng and Reed ([HR89a]). Hoàng and Reed ([HR89a, HR89b]) showed that for $\mathcal{S} = \{1\}, \mathcal{S} = \{2\}, \mathcal{S} = \{3\}$, and $\mathcal{S} = \{1,2\}$ determining whether a graph admits an $\mathcal{S}$-orientation is polynomial. Since then, faster algorithms have been found and a summary of the results for the six classes can be seen in Table 2.1.

Next, a brief description of each of these classes except for one-in-one-out graphs which we will discuss in Chapter 3.

2.6.1 P_4-Simplicial Graphs

Definition 11 *A graph G is* P_4-simplicial *if there is an acyclic orientation of G in which every P_4 of G is of Type 1 or 2.*

In a simplicial orientation, every P_4 is constructed using the P_3's in Figure 2.6. Con-

sequently, these P_4's are all of Type 1 or 2 and the class of P_4-simplicial graphs contains all triangulated graphs.

2.6.2 Generalization of Comparability and Bipolarizable Graphs

Definition 12 *A graph G is a* GCB *(short for "generalization of comparability and bipolarizable") if there is an acyclic orientation of G in which every P_4 of G is of Type 1 or 3.*

Hoàng ([Hoà96]) proved that recognizing GCB graphs is NP-complete. Similar to an argument by Middendorf and Pfeiffer, he showed that 2-IN-3 SAT is polynomially reducible to determining whether a given graph admits a $\{1, 3\}$-orientation.

2.6.3 Bipolarizable Graphs

Definition 13 *A graph G is a* bipolarizable *graph if there is an acyclic orientation of G in which every P_4 of G is of Type 1.*

Bipolarizable graphs are one of the two classes in which a subgraph characterization has been found. Both Hertz ([Her90]) and Reed (unpublished) independently found this subgraph characterization. In order to describe this characterization, we need the following definition.

Definition 14 *An n-*clique wheel *$CW(n)$ for $n \geq 2$ is a graph formed from a clique of size n as follows: first, label the vertices of the clique $v_0, v_1, \ldots, v_{n-1}$. Then add new vertices $p_0, p_1, \ldots, p_{n-1}$ and $q_0, q_1, \ldots, q_{n-1}$ such that*

for $i = 0,1,\ldots,n-1, p_i$ is adjacent to v_j but not to v_{i+1} for $j \neq i \dotplus 1$;

for $i = 0,1,\ldots,n-1, q_i$ is adjacent to v_j but not to v_i, v_{i+1} for $j \neq i, j \neq i+1$;

$P = \{p_0, p_1, \ldots, p_{n-1}\}$ and $Q = \{q_0, q_1, \ldots, q_{n-1}\}$ are stable sets; and

p_i is adjacent to q_i but not to q_j for $j \neq i$.

Figure 2.9 shows an example of $CW(2)$ and $CW(4)$.

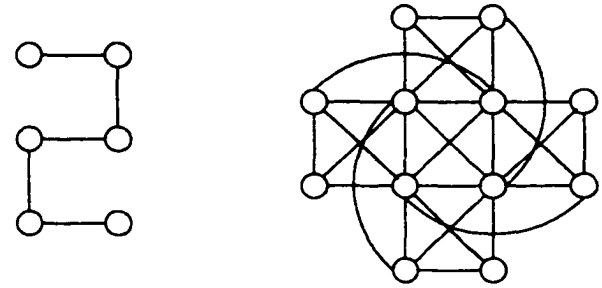


Figure 2.9: The clique wheels $CW(2)$ and $CW(4)$

Theorem 4 ([Her90, Ree]) *A graph is bipolarizable if and only if it does not contain an induced subgraph isomorphic to a clique wheel or any of the graphs shown in Figure 2.10.*

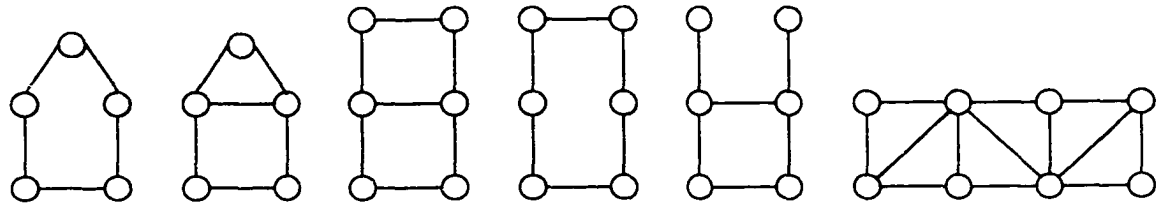


Figure 2.10: A list of minimal non bipolarizable graphs

Spinrad and Johnson ([SJ]) found a recognition algorithm for bipolarizable graphs that runs in $O(n^{3.376})$.

2.6.4 P_4-Indifference Graphs

Definition 15 *A graph G is* P_4-indifferent *if there is an acyclic orientation of G in which every P_4 of G is of Type 2.*

This class of graphs received its name because it generalized the family of graphs known as *indifference* graphs. Indifference graphs admit an acyclic orientation such that every P_3 abc has directed edges $\vec{ab}$ and $\vec{bc}$. Hoàng, Maffray, and Noy found the subgraph characterization that follows, but first we must define a hole. A *hole* is a chordless cycle with at least five vertices.

Theorem 5 ([HMN98]) *A graph is a P_4-indifference graph if and only if it does not contain an induced subgraph isomorphic to a hole or any of the graphs shown im Figure 2.11.*

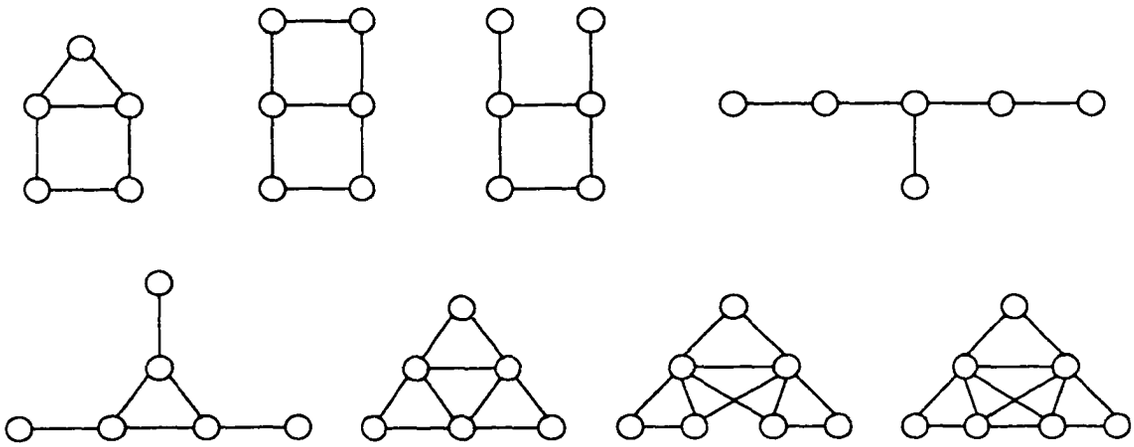


Figure 2.11: A list of minimal non P_4-indifference graphs

Habib, Paul, and Viennot ([HPV]) designed a linear time recognition algorithm based on this characterization.

2.6.5 P_4-Comparability Graphs

Definition 16 *A graph G is* P_4-comparability *if there is an acyclic orientation of G in which every P_4 of G is of Type 3.*

In a transitive orientation, every P_4 is constructed using the P_3's in Figure 2.4. Consequently, these P_4's are all of Type 3 and the class of P_4-comparability graphs contains all comparability graphs. Currently, the fastest recognition algorithm runs in $O(m^2)$ time ([RS]).

2.7 Minimal Non Perfectly Orderable Graphs

Definition 17 *A graph is* minimal non perfectly orderable *(or MNPO) if it is not perfectly orderable but each of its proper induced subgraphs is perfectly orderable. Figure 2.12 shows some example of minimal non perfectly orderable graphs.*

Before we start to look at some of the properties of MNPO graphs, we first need to define a few terms. Let $v_1v_2 \ldots v_k$ denote the chordless path P_k with vertices $v_1, v_2, \ldots, v_k$ and edges v_iv_{i+1} for i = 1, 2, ..., k - 1. The vertices v_1, v_k are called *endpoints* while vertices $v_2, v_3, \ldots, v_{k-1}$ are called *internal* vertices of P_k. In the P_4 *abcd*, vertices b, c are called *midpoints* and edges ab, cd are called *wings* of the P_4. A vertex of a graph G is called *no-mid* if it is not the midpoint of any P_4 in G. Similarly, a vertex of a graph G is called *no-end* if it is not the endpoint of any P_4 in G. The complement of a P_4 is also a P_4 and a vertex is no-mid in G if and only if it is no-end in $\overline{G}$. Chvátal noted the first fact about MNPO graphs.

Lemma 2 *No MNPO graph contains a no-mid or no-end vertex.*

Hoàng, Maffray and Preissmann proved

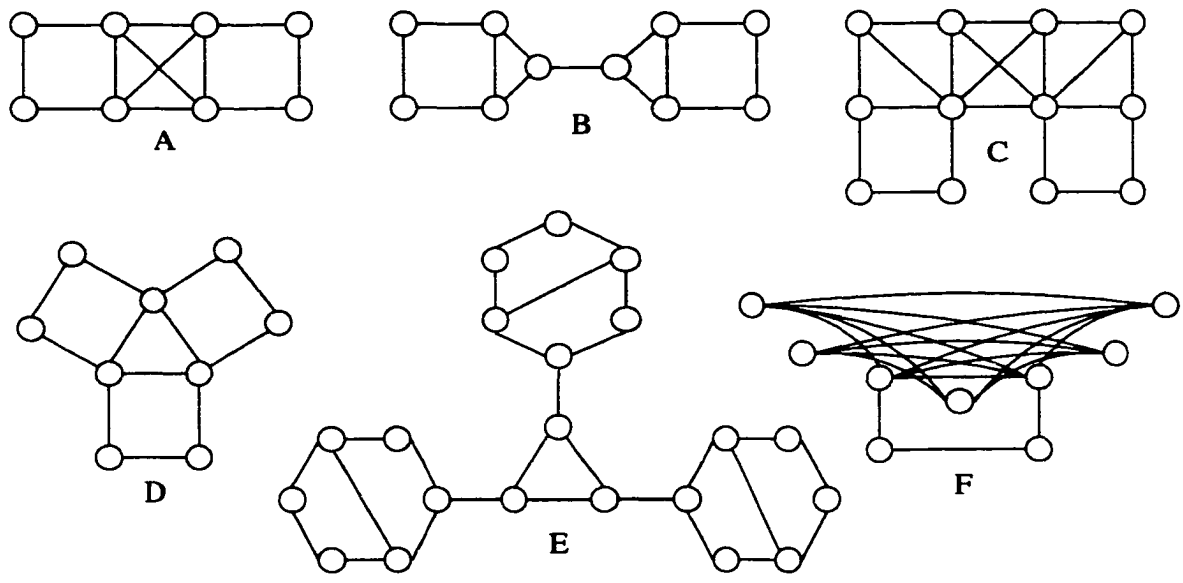


Figure 2.12: A list of some MNPO graphs

Lemma 3 ([HMP91]) *No MNPO graph contains a vertex that is the endpoint of exactly one P_4 and the midpoint of exactly one P_4.*

A *homogeneous* set of a graph $G = (V, E)$ is a set $H \subset V$ such that $|H| \geq 2$ and each $v \in V - H$ is adjacent to all or to none of the vertices in H. A homogeneous set in a graph can be found in linear time ([CH94, DGM97, MS99]) if it exist. Homogeneous sets are useful in designing recognition algorithms for several well known classes of graphs. For example, homogeneous sets are used in the recognition algorithms for comparability graphs and P_4-indifference graphs. The following lemma connects homogeneous sets, simplicial vertices, and no-mid vertices.

Lemma 4 ([HK88]) *Let G be a graph and let x be a no-mid vertex of G. Then either x is a simplicial vertex or there is a homogeneous set H of G that lies entirely in the neighbourhood of x.*

Unfortunately, for the case of MNPO graphs,

Lemma 5 *A MNPO graph cannot contain a homogeneous set.*

2.8 More Classes of Perfectly Orderable Graphs

There are many more classes of perfectly orderable graphs than have been mentioned already and many that are still waiting to be discovered. In this section, we will briefly name and define some of the classes already known to exist.

2.8.1 Complements of Tolerance Graphs

Golumbic and Monma ([GM82]) introduced *tolerance* graphs as a generalization of interval graphs. A graph $G = (V, E)$ is a tolerance graph if there exists a set $\{I_x | x \in V\}$ of closed intervals on a line and a set $\{t_x | x \in V\}$ of positive numbers satisfying $xy \in E$ if and only if $|I_x \cap I_y| \geq min\{t_x, t_y\}$, where $|I|$ denotes the length of interval I. Along with Trotter, they then proved

Theorem 6 ([GMJ84]) *Complements of tolerance graphs are perfectly orderable.*

Currently, there is no polynomial time algorithm for recognizing tolerance graphs.

2.8.2 Charming Graphs

In a graph G, a vertex v is called *charming* if v is not the endpoint of a P_5 or does not belong to a C_5 in G and is not the endpoint of a P_5 in $\overline{G}$. A graph is called *charming* if each of its induced subgraphs contains a charming vertex. The following lemma also shows a property of MNPO graphs.

Lemma 6 ([HMOP92]) *A MNPO graph cannot contain a charming vertex.*

Consequently, charming graphs are perfectly orderable. A $O(n^5)$ time recognition algorithm has also been found for the class of charming graphs.

2.8.3 Nice Graphs

In a graph G, a vertex v is called *nice* if v does not belong to a C_5 and is not the internal vertex of a P_5 in G or a P_5 in $\overline{G}$. A graph is called *nice* if each of its induced subgraphs contains a nice vertex. The following lemma also shows another property of MNPO graphs.

Lemma 7 **([FRRT99])** *A MNPO graph cannot contain a nice vertex.*

Similar to charming graphs, as a consequence of this lemma, nice graphs are also perfectly orderable and can be recognized in $O(n^5)$ time.

2.8.4 Brittle Graphs

In a graph G, a vertex is called *soft* if it is no-mid or no-end in G. A graph is called *brittle* if each of its induced subgraphs contains a soft vertex. Chvátal introduced brittle graphs along with Lemma 2 and showed that brittle graphs are perfectly orderable. One property of brittle graphs is that their complement is also brittle. Many other classes of graphs, such as triangulated graphs (and their complements), bipolarizable graphs, P_4-indifference graphs, and P_4-simplicial graphs have been determined to be brittle. A *HHD-free* graph is a graph that contains no hole with at least five vertices, no house, and no domino (see Figure 2.13). In [HK88] where HHD-free graphs were defined, it was also proven that a graph is brittle if it is HHD-free. Two recognition algorithms have been developed for brittle graphs. Spinrad and Johnson ([SJ]) designed a $O(n^3 log^2 n)$ algorithm while Schäffer ([Sch91]) solved the same problem in $O(m^2)$ time.

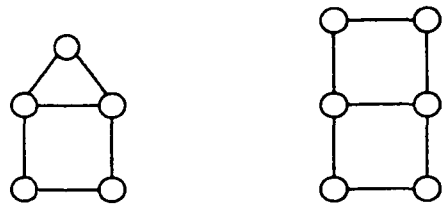


Figure 2.13: House and Domino graphs

A graph G is called *superbrittle* if every vertex is no-mid or no-end in G. Preissmann, de Werra and Mahadev found the following subgraph characterization.

Theorem 7 ([PdWM86]) *A graph is superbrittle if and only if it does not contain any induced subgraph isomorphic to any of the graphs shown in Figure 2.14.*

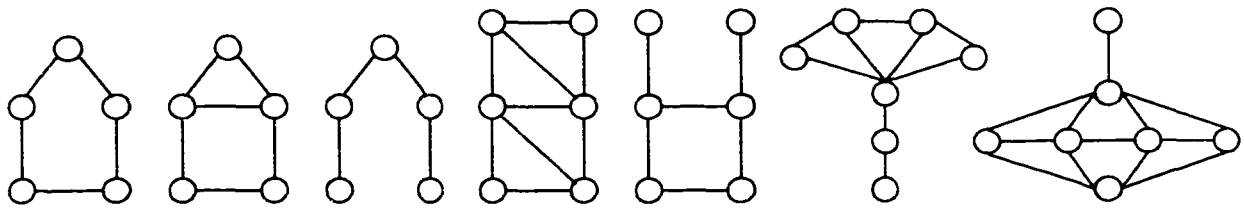


Figure 2.14: The list of all minimal non supperbrittle graphs

Chapter 3

One-In-One-Out Graphs

Chapter 2 discussed several classes of perfectly orderable graphs. It also discussed an alternative to looking at a vertex ordering. In this chapter we will look further at the use of orientations and the steps involved in trying to determine whether a class of graphs can be recognized in polynomial time. Orientations are analogous to vertex orderings in the following way: suppose $G = (V, E)$, then we orientate each edge from u to v for $uv \in E$ if and only if $u < v$ in the graph ordering. The class of graphs that we would like to determine the recognition algorithm complexity for is the class known as *one-in-one-out* graphs. This class of graphs is defined as follows:

Definition 18 *A graph G is* one-in-one-out *if there is an acyclic orientation of G in which every P_4 of G is of Type 2 or 3 (see Figure 2.8).*

As previously discussed, this class of graphs is known to be perfectly orderable since, by definition of the class, no P_4 will be oriented in the manner to create an obstruction. Since we know the graph is perfectly orderable, we also know that the greedy algorithm will produce an optimal colouring. Hence, a polynomial recognition algorithm for this class would let us identify the graph as being perfectly orderable and then colour it, all

in polynomial time.

By looking at the types of P_4 orientations allowed, we can see that this class of graphs generalizes P_4-indifference graphs as well as P_4-comparability graphs and therefore comparability graphs.

The P_4's in this class of graphs are either of Type 2 or Type 3. These two types are quite similar. The wings of each type of P_4 are the same while the middle edge is different. So, when we orient a P_4 in this class, the wings will be oriented in the same direction and it does not matter which way we orient the middle edge. Hence, we can generalize the P_4 as in Figure 3.1.

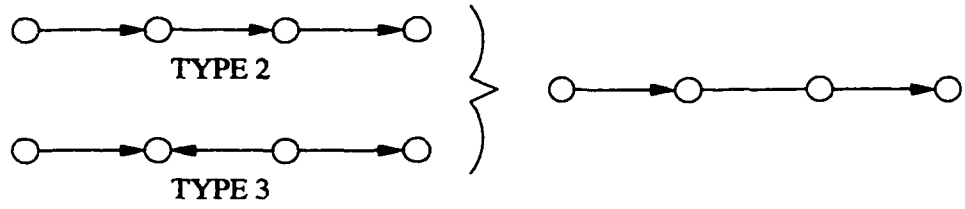


Figure 3.1: Generalization of Type 2 and Type 3 P_4's

With this generalized P_4 orientation, the wings of a P_4 $abcd$ are directed such that ab forces cd and vice versa (that is, an orientation of ab would force the orientation of cd and vice versa). Hence, if ab is directed from a to b then cd is forced to be directed from c to d. This creates a forcing of the orientation of the wings of a P_4. If two P_4's share a common wing, $abcd, cdef$ then we will denote a forcing list by $ab \rightarrow cd \rightarrow ef$.

Definition 19 *Given an oriented graph $G = (V, E)$, let $abRcd$ for $ab, cd \in E$ if $abcd$ is a P_4 or there is an edge $ef \in E$ such that $abRef$ and $cdRef$. R is an equivalence relation that partitions the edges of G into equivlanence classes. In this case, an* equivalence class *is a set of edges that are the wings of P_4's such that a forcing list can be created between any two edges in the set. Consequently, if the direction of one edge in the*

equivalence class is changed, all edges in the class must also change direction or a forbidden orientation is created.

3.1 Finding a Recognition Algorithm

Now, to determine if a polynomial time recognition algorithm for one-in-one-out graphs exists, we will try to create a graph whose orientation will contain a directed cycle of more than one equivalence class. If such a graph exists, it is unlikely that a polynomial time recognition algorithm will be found and an NP-complete reduction similar to Middendorf and Pfeiffer [MP90] or Hoàng [Hoà96] can be looked at. So our goal is to try and create a graph that contains a directed cycle made up of more than one equivalence class.

We will start with a directed 3-cycle with two equivalence classes and then add edges and edge orientations as are required. Since there are two equivalence classes, two edges of the 3-cycle will belong to the same class. Therefore, two edges in the 3-cycle can be forced from a common wing of a P_4. This would create a house structure (see Figure 2.13) where the two edges that make the roof and the edge that is the base of the house would be in the same class. However, the third edge of the 3-cycle (the ceiling of the house), which is a different class must have it's own P_4 which forces the direction of the edge in the 3-cycle (see Figure 3.2). Now, the P_4 $abfg$ is a forbidden orientation so we must add the edge af or ag. If we add only one of af, ag then the P_4 $gfac, fgac$ will merge class 1 and class 2 together, so we must add both edges. In the current state, all of the edges in the 3-cycle are part of the same class since $de \rightarrow fb \rightarrow ce \rightarrow ga \rightarrow bd \rightarrow gf$. So, class 1 merges with class 2. To fix the P_4 $edbf$ which started the forcing, we can add the edge df. We then need to add the edge ef to fix the merging of classes in $edfg$. The rest of the forced orientations appear in Figure 3.2.

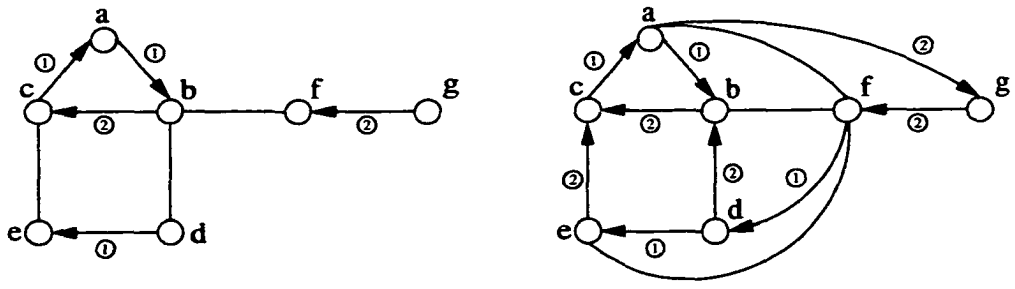


Figure 3.2: A graph oriented with Type 2 and Type 3 P_4's containing a directed cycle

With this graph, we have shown that there is the possibility that a directed cycle of more than one equivalence class could exist. However, this graph is not enough to be certain. If we change the direction of class 2, then we no longer have a directed cycle. So we need to find a graph that always has a directed cycle. Finding such a graph by trial and error or by trying to prove that certain edges must exist, is very difficult and there are several cases that have to be looked at. One method would be to have two 3-cycles where one would be directed and the other would have an edge that opposes the direction of the other two. Then, try and connect these two 3-cycles in such a manner that the single edge that opposes the directed cycle would be in the same class as one of the equivalence classes in the directed cycle. And then the remaining edges would belong to another equivalence class (see Figure 3.3). This way, when the direction of one of the classes is switched, the directed 3-cycle will switch to the non directed 3-cycle and vice versa. So the directed 3-cycle toggles as the direction toggles. However, the possibilities to create this graph are numerous.

Another method would be to use two copies of the graph from Figure 3.2 and try and join them together in a manner similar to the one above. Again, the number of possibilites to connect these two copies are numerous. Extra vertices may need to be added, as well as, several edges.

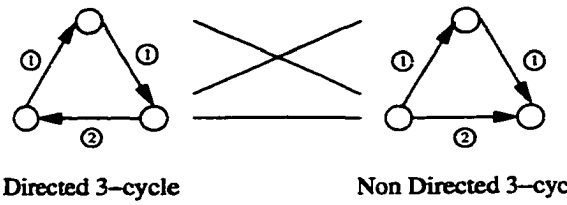


Figure 3.3: Connect the two 3-cycles to join the equivalence classes

3.2 Graph Generation

When we have so many cases to check, the logical thing seems to be to let a computer check them for us. However, the problem of joining the two graphs together is still very difficult for a computer. Consider joining the two graphs together without having to add a new vertex. In this case, we need to check all of the possible combinations of connecting 7 vertices of copy one to 7 vertices of copy two (see Figure 3.4). Since each vertex could be connect to all 7 vertices in the other copy of the graph, in total, there are 2^{49} different combinations. This problem is even too big for a computer to handle in a reasonable amount of time. Another option is to generate all of the non-isomorphic graphs and check to see if they contain the property we are looking for. This program can then be used to check for other properties also.

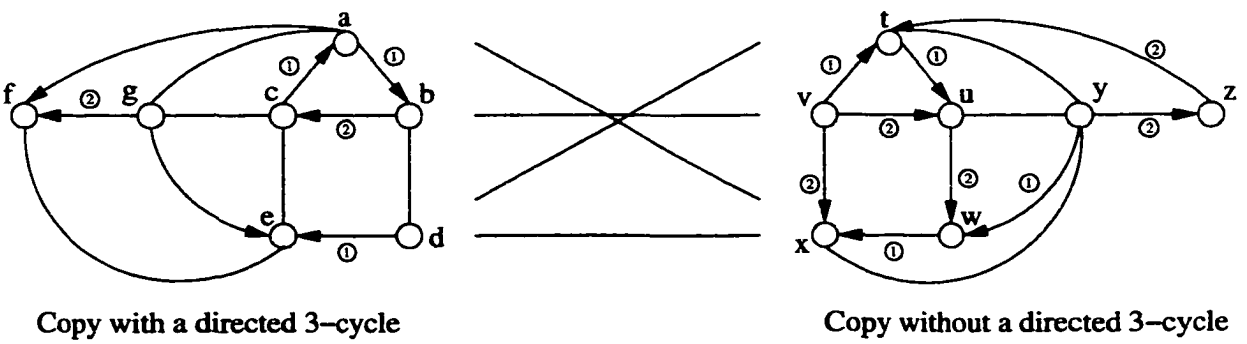


Figure 3.4: Connect the two graphs to join the equivalence classes

Many larger computers today have more than one processor. If a traditional program for generating graphs is written, it will only make use of one processor. In the next section, we will describe a client/server model for generating graphs in a traditional way.

The basic method for generating graphs will be to generate all of the non-isomorphic graphs on n vertices. The easiest way to do this is start with all of the non-isomorphic graphs on $n-1$ vertices and add a vertex. The vertex will have to be added in all of it's $2^{n-1}-1$ possible ways to connect it to the base graph. Even though we will know that all of our input graphs are non-isomorphic, it does not guarantee that all of the graphs created by will be non-isomorphic. So, each graph we create, we will have to check to see if it is a new graph.

To limit the number of graphs that we have to perform an isomorphism check on, we can store the graphs with a common degree sequence in the same file. (e.g. file 3-2-1.grp will contain all of the graphs that have 3 vertices of degree 1, 2 vertices of degree 2, and 1 vertex of degree 3.) This also speeds up the isomorphism check by limiting the number if possible mappings between the vertices of each graph. If each vertex has a unique degree, there would be only one mapping. However, if all n vertices had the same degree, we would have to check $(n^2+n)/2$ different mappings. If all of the graphs in the files are tested and fail the isomorphism check, we will have found a new graph. This graph is then written to the file and will be used in further isomorphism testing.

After finishing all of the different possible ways to add a new vertex, the program will then start the process over with a new base graph from the input file. The program halts after all of the base graphs from the input file have been used.

3.3 Client/Server Model for Graph Generation

To build a client/server model for generating graphs, it is important to decide which tasks will be done by each component of the model and to determine if there will be any synchronization problems. One such problem is similar to the classical readers/writers problem. Some processes will want to read a certain file and others will want to write to the same file. Both of these cannot be done at the same time. In fact, the file cannot be written to unless no one is reading the file. Also, only one writer can write to the file at a time. Otherwise, readers will not have current information and writers may overwrite each other. So there needs to be some device to handle the mutual exclusion of the generated graph files (labelled by degree count as mentioned above).

In this model, the server will be responsible for sending out new input graphs upon request and handling the mutual exclusion for the generated graph files. The client will be responsible for all of the actual processing of the graphs. There will need to be two-way communication between the server and the client for various situations.

When a client is started, it must first request an input graph from the server. If there are still base graphs that need to be checked, the server sends it to the client; otherwise, the server sends a terminate message. The client then begins to work on this base graph and add a new vertex. Once a graph of n vertices is created, it will need to be compared with the graphs of a similar degree count. To access this file, the client must request access to it from the server. The server then checks to see if anyone else is currently using the file. If another client is currently using the file, the other client will be placed into a waiting queue for this file. Once the client is finished with the file, it sends a release file message back to the server. The server is then able to send the go ahead message to waiting client in the queue. A client has sole access to the file until it gives up its permission back to the server.

If there are several clients, the number of messages may be large. The server may then be broken into different processes to help offset the number of messages a single process server will need to handle. One such process could be to strictly handle the requests for new input base graphs. The handling of degree count file requests can also be spread over several processes.

By using this model, we can then make full use of some of today's latest multiprocessor computers. But has technology progressed far enough for us to generate enough graphs such that either we find the graph we are looking for or to at least check enough graphs so we can be reasonably confident that it doesn't exist? The answer is an unfortunate no. On one of the fastest computers in Canada, a Silicon Graphics Origin Cray computer with 40 350MHz processors, only graphs under 9 vertices can be generated in a reasonable amount of time. Graph generation in this manner is a hard problem that becomes very cumbersome very quickly. When the graphs on under 9 vertices were checked to see if any would always have a directed cycle made of two or more equivalence classes, the result was negative. So, if such a graph exists, it must contain at least 9 vertices.

3.4 A Subclass of One-In-One-Out Graphs

In this section, we look at graphs that do not contain an induced subgraph isomorphic to a *pyramid* (see Figure 3.5).

Since it is very difficult to determine if it is possible to obtain a graph with a Type 2 and 3 orientation that also contains a directed cycle of more than one equivalence class, we will look at a slightly different problem. We will look at the class of graphs that are pyramid-free and oriented using Type 2 and 3 orientations. If we can show that this class does not contain a directed cycle of more than one equivalence class then it is probable

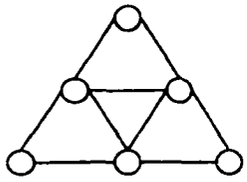


Figure 3.5: A pyramid

that we can recognize this class in polynomial time. This class of graphs is a subclass of one-in-one-out graphs and comparability graphs are a subclass of this class. So we shall start to show that a graph G admits an acyclic orientation if every P_4 in G is of Type 2 or Type 3 and G is pyramid-free.

It is clear that some common structures are forbidden in this class. For example, chordless cycles of odd length, C_{2k+1} for $k \geq 2$ and $\overline{C_6}$. When orienting the P_4's, these graphs will contain a directed cycle made up of one class. (See Figure 3.6 for a list of some forbidden subgraphs.)

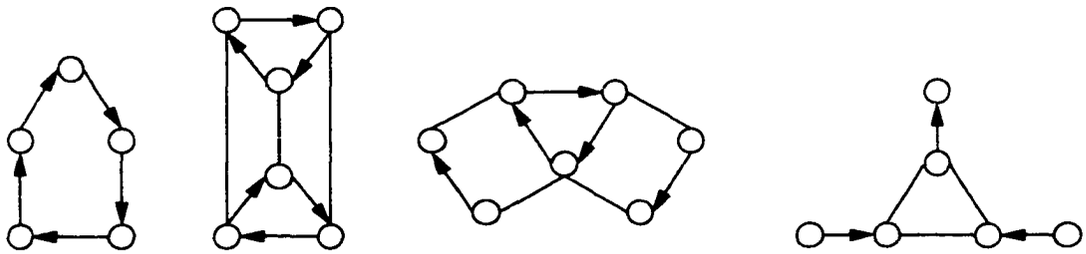


Figure 3.6: Graphs forbidden in Type 2 and Type 3 orientations

Hoàng and Petrick proved the following theorem about the class of pyramid-free graphs.

Theorem 8 *Let G be a pyramid-free graph. Suppose G admits an orientation such that each equivalence class is acyclic and each P_4 is of Type 2 or 3. Then, G cannot contain*

a directed 3-cycle whose edges belong to different equivalence classes.

Proof: We will prove the theorem by contradiction. Let $G = (V, E)$ be a pyramid-free graph which admits an orientation such that each equivalence class is acyclic and each P_4 is of Type 2 or 3. Suppose that G contains a directed 3-cycle on vertices a, b, c, where edges ab, bc, ca do not belong to the same equivalence class. Since each edge in the 3-cycle is directed, it must be the wing of some P_4 and hence, has been assigned an orientation. So, let vertices d, e, f, g, h, i to be the additional vertices of the P_4's that have a wing in the directed 3-cycle such that d, f, h will be midpoints and e, g, i will be endpoints of the P_4's. Thus, the vertex sets $\{a, b, d, e\}$, $\{b, c, f, g\}$, and $\{a, c, h, i\}$ induce P_4's (see Figure 3.7).

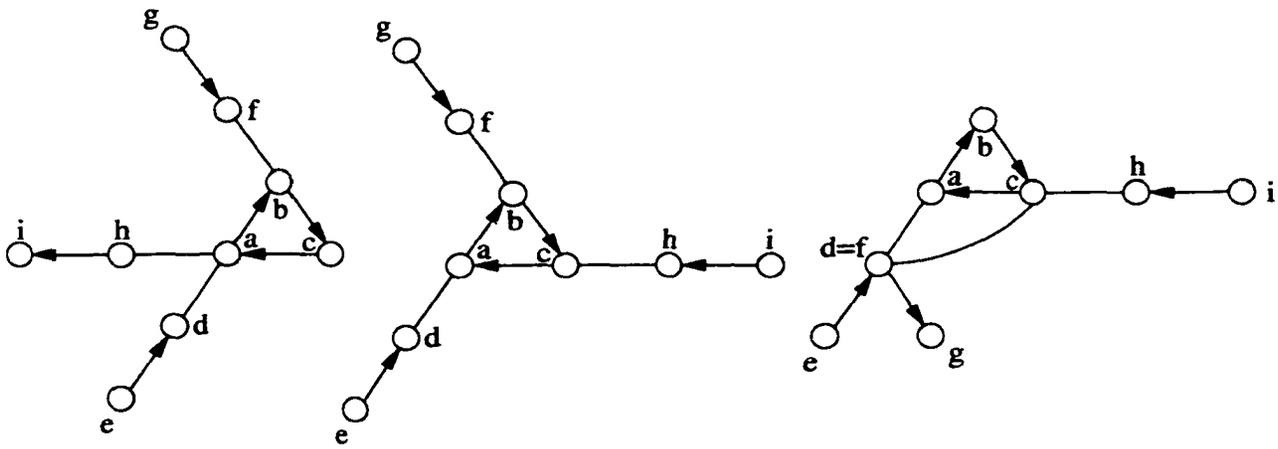


Figure 3.7: Some possible configurations for vertices $\{d, e, f, g, h, i\}$

First, we will prove the following

Fact 1 *The vertices d, e, f, g, h, i are all distinct in G. Moreover, each edge of the directed 3-cycle abc is the wing of a distinct P_4 in G.*

Proof: Suppose that the vertices d, e, f, g, h, i are not distinct in G. Since the edges

of the directed 3-cycle belong to at least two equivalence classes, we also know that the edges de, fg, hi cannot all be the same edge.

Case 1: Without loss of generality[1], suppose that de is the same edge as fg. Then, in order to maintain the cyclic orientation of the 3-cycle, $d = g$, $e = f$ and ab, bc, de are all in the same equivalence class. This graph is a ***house*** where the walls of the house, ad, ce, are undirected.

Next, it is clear that i cannot be a, b, c, d, e since $\{a, c, h, i\}$ must induce a P_4. So, i must be a distinct vertex.

Claim 1 *h is also a distinct vertex in G.*

Proof: Suppose h is not distinct in G. Then, the only possibilities are that $h = d$ or $h = e$ (see graph G_1 and G_2 respectively in Figure 3.8).

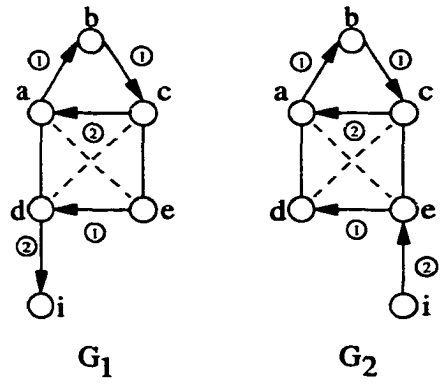


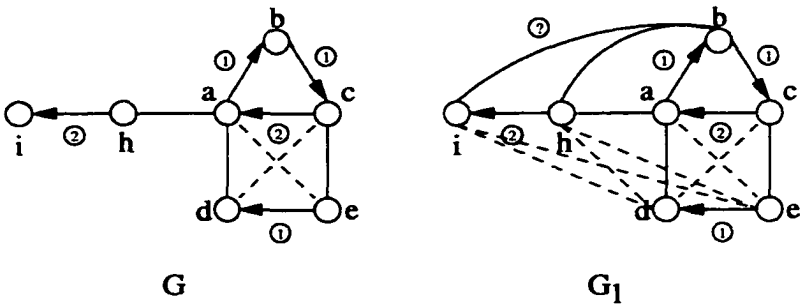
Figure 3.8: The P_4 on $\{a, b, h, i\}$ shares an edge with the house

In example G_1, the P_4 *badi* is not of Type 2 or Type 3. It has a forbidden orientation. So, there must be an edge bi. However, we then have the P_4 *dibc* which would then force equivalence class 1 to be the same as equivalence class 2 and all edges of the directed 3-cycle would be in the same equivalence class. Hence, $h \neq d$.

[1]In later cases, we will use the abbreviation WLOG for "without loss of generality".

Due to the symmetric nature of G_1 and G_2, a similar proof shows $h \neq e$. □ (end of proof for claim 1)

Since both h, i are distinct vertices, the edges of the P_4 induced by $\{a, c, h, i\}$ are distinct from the edges of the house except for ac. WLOG, assume that $cahi$ is a P_4, instead of $achi$ (see graph G, Figure 3.9).



G G_1

Figure 3.9: vertices h, i are not part of the house

In G, the P_4 $ihab$ has a forbidden orientation. So, either the edge bh or bi must exist. If bh is an edge and bi is not an edge, then $ihbc$ is a P_4 with a forbidden orientation. If bi is an edge and bh is not an edge, then the P_4 $hibc$ causes equivalence class 1 to be the same as equivalence class 2 and we would have a directed cycle within one equivalence class. Consequently, both bi, bh must be edges.

If di is an edge then ie must also be an edge otherwise, $idecbi$ would be a C_5[2]. However, if both these edges exist, then b, a, c, i, d, e induces a $\overline{C_6}$[3]. Therefore di is not an edge. Consequently, ie is also not an edge; otherwise $ibadei$ would be a C_5. Then, dh is not an edge because $\{b, h, a, i, d, c\}$ would induce a pyramid and eh is not an edge because the P_4 $dehi$ would cause equivalence class 1 to be the same as equivalence class 2 (see graph G_1 Figure 3.9).

[2] C_5 is one of the forbidden subgraphs in Figure 3.6

[3] $\overline{C_6}$ is also one of the forbidden subgraphs in Figure 3.6

However, the edges ab, bc, ca are still in the same equivalence class since, $ca \rightarrow hi \rightarrow da \rightarrow bi \rightarrow ec \rightarrow ab \rightarrow ed$ and ed forces both ab and bc, a contradiction. Hence, the edge de cannot be the same edge as fg and de, fg, hi are all distinct edges.

We know from case 1, that the edges de, fg, hi are distinct, however, this does not imply that all of the vertices d, e, f, g, h, i are distinct. Consider the possibility that two of the P_4's induced by $\{a, b, d, e\}, \{b, c, f, g\}, \{c, a, h, i\}$ share a common vertex. WLOG, suppose $bade$ is a $P4$ and the vertices b, c, f, g induce a P_4 such that f or g is not a distinct vertex.

Case 2: Suppose that f is not a distinct vertex in G, and consequently, g must be distinct in G or the graph is one discussed in case 1. $cbfg$ cannot be a P_4 if f is not distinct; otherwise, $bade$ would not be a P_4. Thus, $bcfg$ must be a P_4. If $f = e$ (see graph G, Figure 3.10), then dc must be an edge or $\{a, b, c, d, e\}$ induce a house and by case 1, a contradiction. Similarly, ga is not an edge or $\{a, b, c, e, d\}$ induce a house and by case 1, a contradiction. However, the P_4 $aceg$ cause a forbidden orientation (see graph G_1, Figure 3.10).

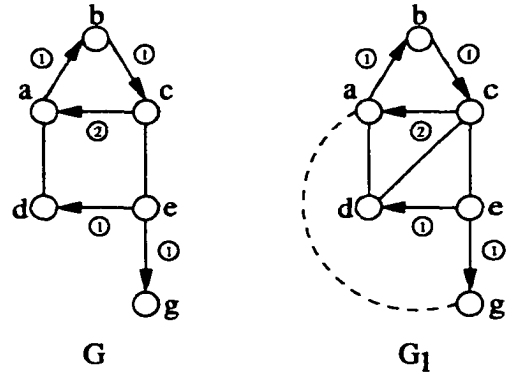


Figure 3.10: if $e = f$ and $d \neq g$ then the P_4 $aceg$ has a forbidden orientation

Thus, if f is not distinct, then $f = d$ and $bcfg$ is a P_4 (see graph G, Figure 3.11).

In this graph, there is a forbidden orientation on the P_4's $badg$ and $bcde$, so the edges

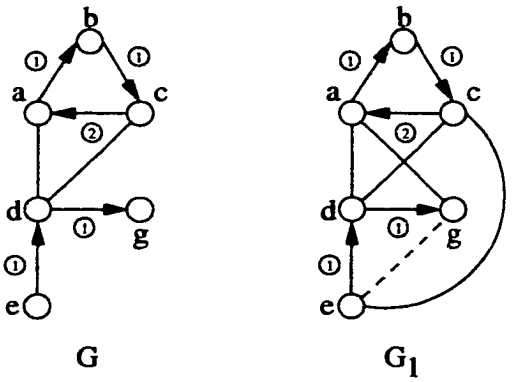


Figure 3.11: ab, bc are not forced from the same wing of a P_4

ag and ec must exist. Consequently, eg cannot be an edge or $\{b, a, c, e, g\}$ induce a house and by case 1, a contradiction. However, $\{g, d, a, e, c, b\}$ induce a pyramid (see graph G_1, Figure 3.11), a contradiction to G being a pyramid-free graph.

So, f must be a distinct vertex.

Case 3: Suppose that g is not a distinct vertex. Then g must be either the vertex d or e and either $bcfg$ or $cbfg$ is a P_4.

If $g = d$ and $bcfg$ is a P_4, (see graph G, Figure 3.12), then $edac$ is a P_4 with a forbidden orientation, so, ce must be an edge. However, then $\{a, b, c, e, d\}$ induce a house (see graph G_1, Figure 3.12) and by case 1, a contradiction.

If $g = e$ and $bcfg$ is a P_4 (see graph G, Figure 3.13), then $acfe$ is a P_4 with a forbidden orientation, so fa must be an edge. However, now the P_4 $bafe$ has a forbidden orientation (see graph G_1, Figure 3.13).

If $g = d$ and $cbfg$ is a P_4 (see graph G, Figure 3.14), then $edac$ is a P_4 with a forbidden orientation, so ce must be an edge. Now, a, b, c, e, g induce a house (see graph G_1, Figure 3.14) and by case 1, a contradiction.

Finally, if $g = e$ and $cbfg$ is a P_4 (see graph G, Figure 3.15), then $cade$ is a P_4 with a forbidden orientation, so cd must be an edge. Consequently, $edcb$ is a P_4 with a forbidden

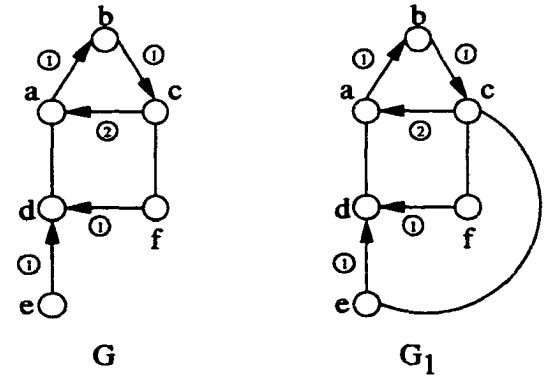


Figure 3.12: if $d = g$ and $e \neq f$ then $\{a, b, c, e, d\}$ induce a house

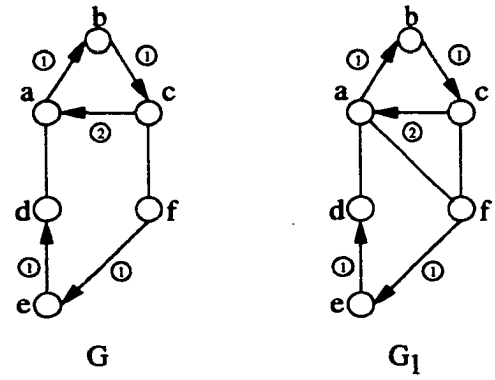


Figure 3.13: if $e = g$ and $d \neq f$ then the P_4 $bafe$ has a forbidden orientation

orientation (see graph G_1, Figure 3.15), a contradiction.

Thus, g must also be distinct in G. Moreover, no two of the P_4's induced by $\{a, b, d, e\}$, $\{b, c, f, g\}$, $\{c, a, h, i\}$ share a common vertex. Hence, we have proven our fact that the vertices d, e, f, g, h, i are distinct in G and that each edge of the directed 3-cycle is the wing of a distinct P_4. □ (end of proof for Fact 1)

Since we know that each edge of the directed 3-cycle is the wing of a distinct P_4 in G, there are two cases that we must now consider, depending on how each P_4 extends out from their corresponding edge on the 3-cycle. The two cases are: G could be symmetric

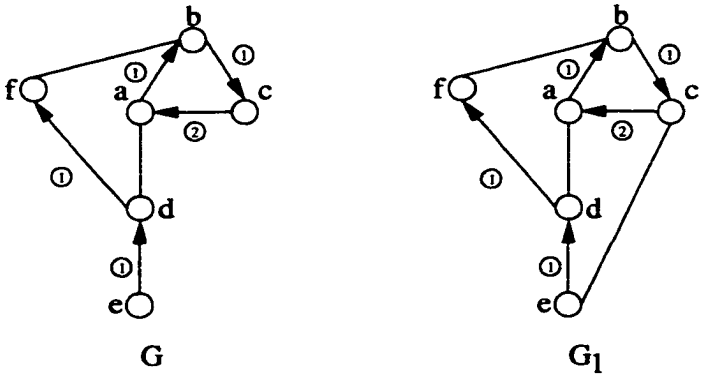


Figure 3.14: if $d = g$ then $\{a, b, c, e, g\}$ induces a house

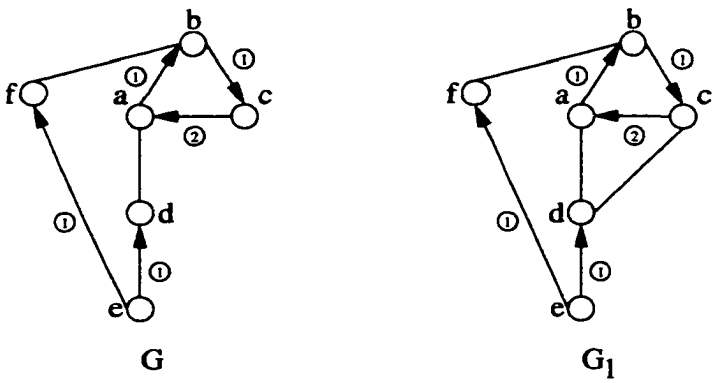


Figure 3.15: if $e = g$ then the P_4 $edcb$ has a forbidden orientation

or G could be non-symmetric (see Figure 3.16).

Case 1: In the symmetric case, P_4's extend from the directed 3-cycle in a uniform manner similar to the spokes of a wheel in G. WLOG, assume that $bade, cbfg, achi$ are P_4's (see Figure 3.16).

If $ce \notin E$ then, $cd \in E$; otherwise, the P_4 $edac$ would have a forbidden orientation. However, the P_4 $edcb$ has a forbidden orientation. Therefore, $ce \in E$. By symmetry, we can also conclude that $ag, bi \in E$.

If we look at the induced subgraph on $\{a, b, c, g, e, i\}$ we see that this forms a co-

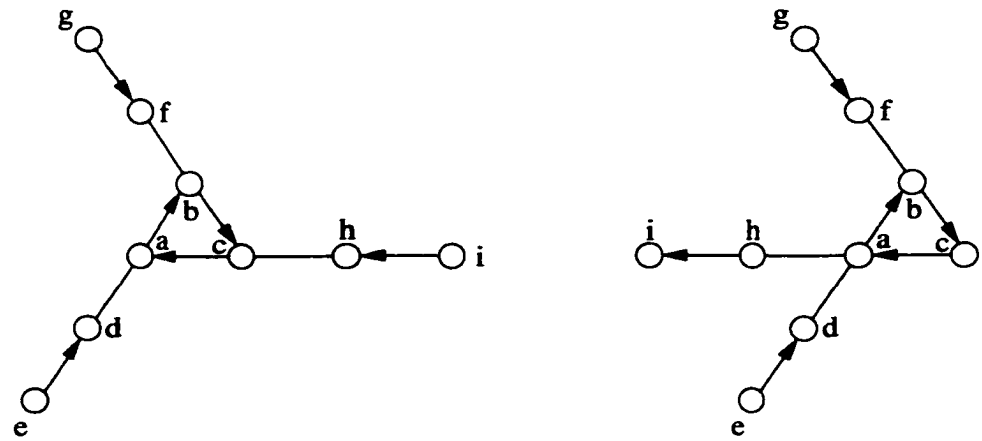


Figure 3.16: Two cases where each P_4 is unique

pyramid which is a forbidden subgraph. Since we know gc, gb, ia, ic, eb, ea must be nonedges to maintain the P_4's already defined, one edge on the vertices $\{g, e, i\}$ must exist. If all three edges gi, ie, eg existed, then, the induced subgraph on $\{a, b, c, g, e, i\}$ is a $\overline{C_6}$ which is another forbidden graph. If two edges, WLOG say gi, ge, exist, then, the induced subgraph on $\{b, c, e, g, i\}$ is a C_5 which is another forbidden graph. So, the only possibility is that only one edge on $\{g, e, i\}$ exists. Because of the symmetry of this graph, it does not matter which edge, so let $ge \in E$ and $gi, ie \notin E$.

Now, suppose $fa \notin E$ then some of the edges would be oriented in the following way (where we know a P_4 must exist)[4]: $ed \rightarrow ab \rightarrow fe \rightarrow ca \rightarrow ih$ and $ca \rightarrow gf \rightarrow bc$. Since ab, ca, bc are all in the same class, $fa \in E$.

The induced subgraph on $\{a, g, f, b, c, i\}$ would form a pyramid if $fi \in E$, so $fi \notin E$. Consequently, bi is oriented from b to i and ec from e to c because of the P_4's $gfbi$ and $ecbi$.

[4]The notation $ab \rightarrow cd$ denotes that the edges ab, cd are directed from a to b and c to d and that they are the wings of a P_4 so the direction of one forces the direction of the other.

Next, if $bh \in E$, then $eh \notin E$ or the induced subgraph on $\{b, a, c, h, i, e\}$ is a pyramid. However, the P_4 $ihce$ would force ce to be oriented from c to e which contradicts its current orientation. Thus, $bh \notin E$.

Then id must be a nonedge or it would create the forbidden orientation on the P_4 $edib$. Also, $hd \notin E$ or the induced subgraph on $\{a, b, i, h, d\}$ would be a C_5.

Suppose, $dc \notin E$ then $decb$ is a P_4 and by reviewing the forced orientations, we obtain: $bc \rightarrow ed \rightarrow ab \rightarrow ih \rightarrow ac$. Hence, bc, ab, ac are all in one equivalence class so dc must be an edge and oriented from c to d because of the P_4 $ihcd$. However, the P_4 $dcbi$ then has a forbidden orientation, a contradiction. Thus, G cannot be symmetric since this case contradicts the hypothesis that the orientation of all P_4's is either of Type 2 or 3 (see Figure 3.17).

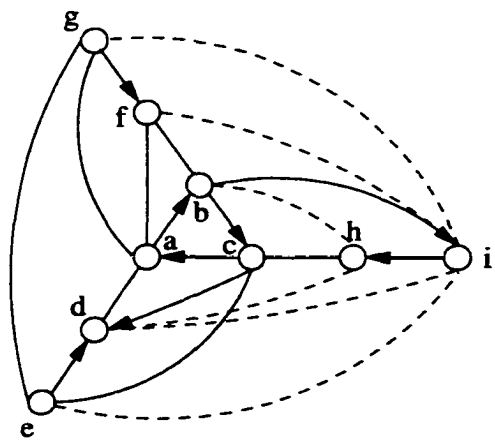


Figure 3.17: The Symmetric Case

Case 2: In the non-symmetric case, two of the P_4's extend from the directed 3-cycle similar to the symmetric case but the third P_4 extends in the opposite direction in G. WLOG, assume that $bade, cbfg, cahi$ are P_4's (see Figure 3.16). We will show that this case also has a contradiction by showing that G contains a forbiden subgraph, a co-pyramid on $\{a, b, c, e, g, i\}$. To show this, we will prove that $bh, cd, af \in E$ and

$ge, gi, ei \notin E$.

As in the symmetric case, we must have the edges ga, ib, ec. Now, the vertices $\{i, h, a, b, c, d, e\}$ induce a joint house[5] so all of the edges of the 3-cycle would fall into one equivalence class. Since this is a forbidden subgraph, G must have either the edge bh or cd or both edges. Next, we will prove that bh, cd, and af must all be edges in G.

Claim 2 *bh must be an edge in G.*

Proof: We will use a proof by contradiction. Suppose $bh \notin E$, then $af, cd \in E$ or G would contain a subgraph isomorphic to the joint house (mentioned above). However, $\{i, h, a, c, b\}$ induces a house that forces edges bc, ac to be in the same equivalence class (see graph G, Figure 3.18).

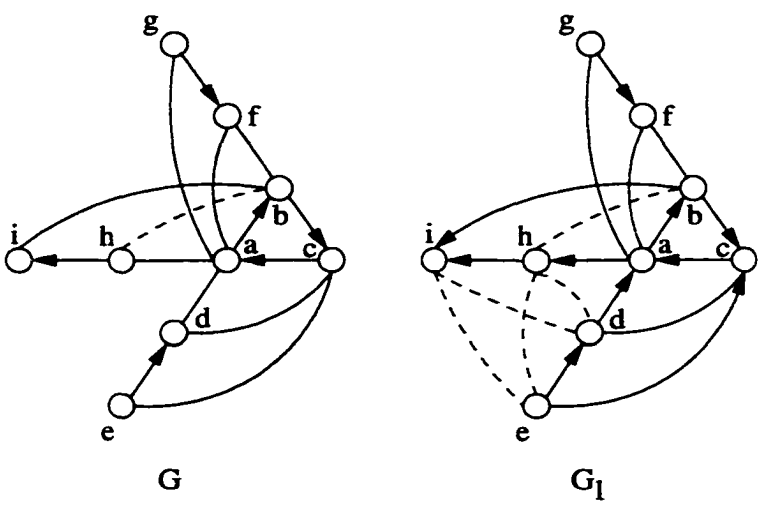


Figure 3.18: $bh \in E$

Suppose $ie \in E$ then, the P_4's $abie$, $acei$ would force ab, ac to be in the same equivalence class. Since we know bc, ac are in the same class, all three edges of the directed 3-cycle would then be in the same class, a contradiction. So, $ie \notin E$.

[5]A joint house is one of the forbidden subgraphs in Figure 3.6

eh is also not an edge or the induced subgraph on $\{e, h, i, b, c\}$ would be a C_5. This then forces $hd \notin E$ or the vertices $\{b, a, d, e, c, h\}$ would induce a pyramid. And consequently, $di \notin E$ because $edih$ would be a P_4 with a forbidden orientation.

If we look at the current forced orientation of the graph, $ca \rightarrow hi \rightarrow da \rightarrow bi \rightarrow ec \rightarrow ah \rightarrow ed \rightarrow ab$ (see graph G_1, Figure 3.18). So, ca and ab are in the same equivalence class and since bc, ac are already in the same class, all three edges of the directed 3-cycle are in the one class, a contradiction. Thus, bh must be an edge in G. □

Claim 3 *cd must be an edge in G.*

Proof: We will use a proof by contradiction similar to above. Suppose $cd \notin E$, then $bh, af \in E$ or G would contain a subgraph isomorphic to the joint house. However, $\{d, a, b, c, e\}$ induces a house that forces ab, bc to be in the same equivalence class (see graph G, Figure 3.19).

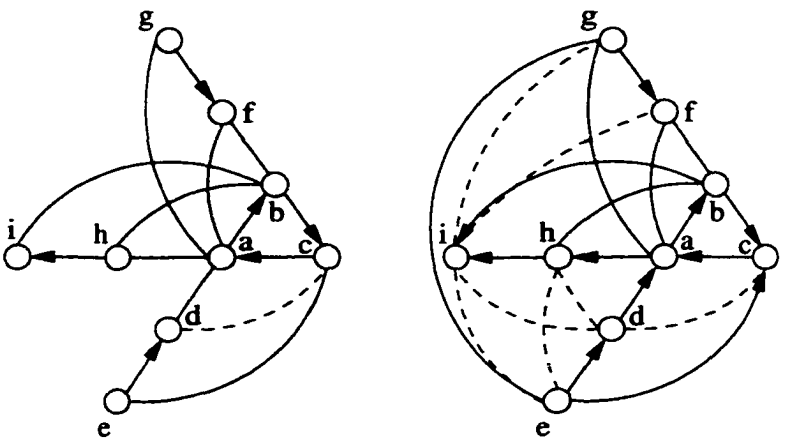


Figure 3.19: $cd \in E$

Suppose $ie \in E$ then, the P_4's $abie, acei$ would force ab, ac to be in the same equivalence class. Since we know ab, bc are in the same class, all edges of the directed 3-

cycle would then be in one class, a contradiction. So, $ie \notin E$. Similarly, $ig \notin E$ or $cagi, cbig$ would also force all edges of the directed 3-cycle in the same class. Consequently, $eg \in E$ or $\{a, b, c, e, g, i\}$ would induce a co-pyramid, a forbidden subgraph.

Then, $fi \notin E$ or $\{a, g, f, b, c, i\}$ would induce a pyramid. Also, $di \notin E$ or $\{e, d, i, b, c\}$ would induce a C_5. Then, $hd \notin E$ or the induced subgraph on $\{b, i, h, a, c, d\}$ would form a pyramid. Finally, $eh \notin E$ or the P_4 $dehi$ would have a forbidden orientation.

If we look at the current forced orientation of the P_4's in G, $ca \rightarrow hi \rightarrow da \rightarrow bi \rightarrow ec \rightarrow ah \rightarrow ed \rightarrow ab$ (see graph G_1, Figure 3.19). So, ca and ab are in the same equivalence class and since ab, bc are in the same class, all three edges of the directed 3-cycle are in one class, a contradiction. Thus, cd must be an edge in G. □

Claim 4 *af must be an edge in G.*

Proof: Again, we will use a proof by contradiction. Suppose, $af \notin E$ then $\{g, f, b, c, a\}$ induces a house which forces bc, ca to be in the same equivalence class (see graph G, Figure 3.20).

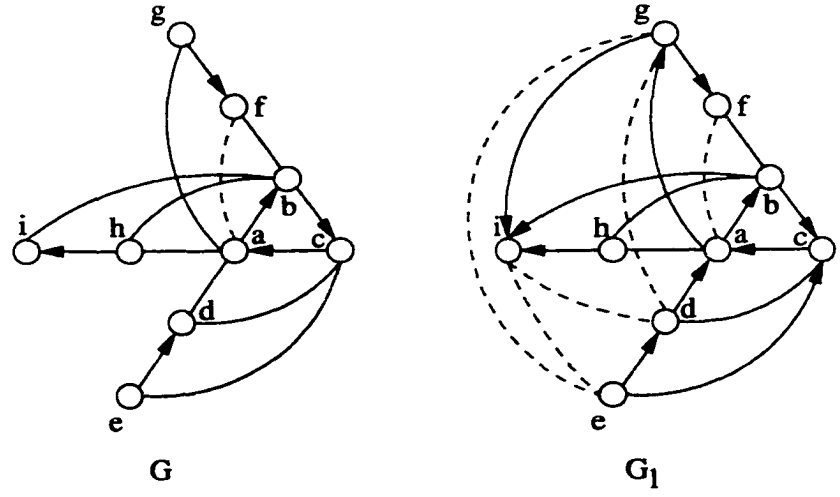


Figure 3.20: $af \in E$

Suppose $ie \in E$ then, the P_4's $abie, acei$ would force ab, ac to be in the same equivalence class. Since we know ab, bc are in the same class, all three edges of the directed 3-cycle would then be in the same class, a contradiction. So, $ie \notin E$. Similarly, $ge \notin E$ or $bceg, egab$ would also force all three edges of the driected 3-cycle to be in one class. Consequently, $gi \in E$ or $\{a, b, c, e, g, i\}$ would induce a forbidden co-pyramid subgraph.

Then, $dg \notin E$ or $\{c, b, a, d, e, g\}$ would induce a pyramid. We know that the orientation of gi must be from g to i because of the P_4 $cbig$. Then $di \notin E$ or $edig$ would be a P_4 with a forbidden orientation.

Now, if we look at the current orientation of the graph, $bc \rightarrow gi \rightarrow da \rightarrow bi \rightarrow ec \rightarrow ag \rightarrow ed \rightarrow ab$ (see graph G_1, Figure 3.20). Thus, bc, ab are in the same class and since we know bc, ca are in the same class, all three edges of the directed 3-cycle are in one class, a contradiction. Thus, af must also be an edge in G. □

Now that we have shown that af, bh, cd must edges in G, consider the induced subgraph on $\{a, b, c, e, g, i\}$, a co-pyramid. From the symmetric case, we know that there must be only one edge between $\{e, g, i\}$. To show that there is a contradiction in the non-symmetric case, we will prove that G has no edge on $\{e, g, i\}$. Consequently, G then has subgraph isomorphic to a forbidden subgraph.

Claim 5 *ge must be a nonedge in G.*

Proof: We will use a proof by contradiction. Suppose $ge \in E$ then $\{g, e, c, a, b\}$ induces a house and ab, bc must be part of the same equivalence class. Also, since ge is an edge, $ei, gi \notin E$ because of the restrictions on the before mentioned co-pyramid (see graph G, Figure 3.21).

By looking at the induced subgraph on $\{a, g, f, b, c, i\}$, if $fi \in E$ then this subgraph is a pyramid, so $fi \notin E$. If we then look at the orientation of the edges, $gf \rightarrow bi \rightarrow ec$. Hence, $he \notin E$ or there would be a forbidden orientation on the P_4 $ihec$. Also, $id \notin E$ or

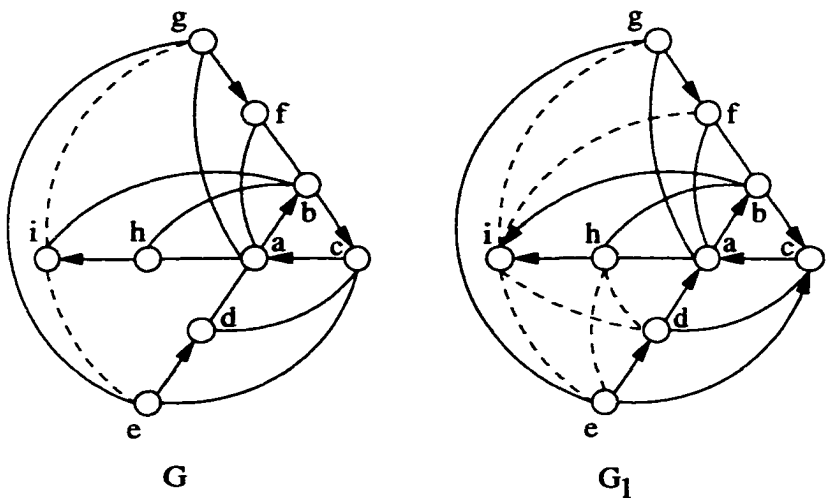


Figure 3.21: ge cannot be an edge

we would have a house on $\{i, d, e, c, b\}$ in which there would be a forbidden orientation of either $ibce$ or $bide$. If $dh \in E$ then $ab \rightarrow ed \rightarrow hi \rightarrow ca$ and since ab, bc are in the same class, all three edges of the directed cycle would be in one class, so $dh \notin E$.

Then, the forced orientation would be $ca \rightarrow hi \rightarrow da \rightarrow bi \rightarrow gf \rightarrow bc$ so ac, bc are in the same class and ab, bc, ac are all in one class (see graph G_1, Figure 3.21), a contradiction. Thus, ge must not be an edge in G. □

Claim 6 *gi must be a nonedge in G.*

Proof: Similar to above, we will show that gi must be a nonedge by using a proof by contradiction. Let $gi \in E$, then $ei \notin E$ (see graph G, Figure 3.22). Also, $\{c, a, b, g, i\}$ induces a house so ca, bc must be in the same equivalence class and gi is oriented from g to i. The edge ag is oriented from a to g because $edag$. Then, edge ec is oriented from e to c because of $ecag$.

Then, $dg \notin E$ or $\{c, b, a, d, e, g\}$ would induce a pyramid. $di, eh \notin E$ or $edig, cehi$, respectively, would have forbidden orientations. Then, $dh \notin E$ or $ab \rightarrow ed \rightarrow hi \rightarrow ac$

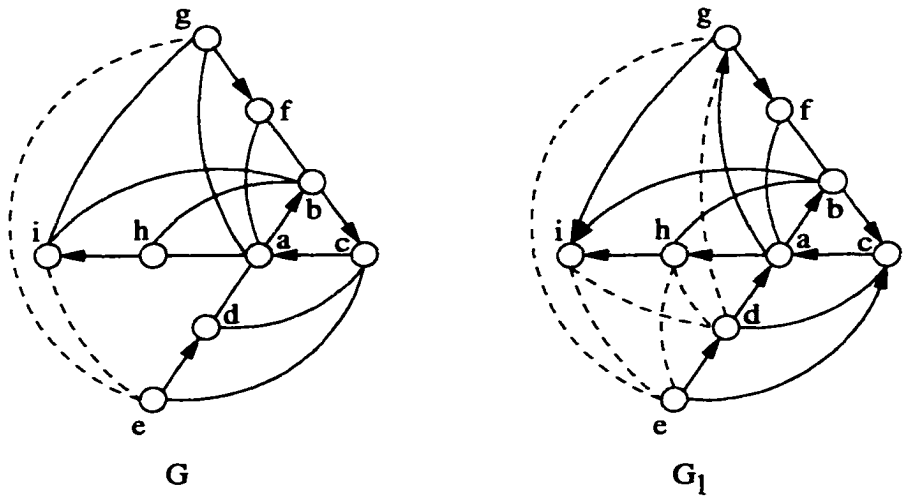


Figure 3.22: gi cannot be an edge

so ab, ac are in the same class and hence, ab, ac, bc would all be in one class.

Then by looking at the forcings, we have $ca \rightarrow hi \rightarrow da \rightarrow bi \rightarrow ec \rightarrow ah \rightarrow ed \rightarrow ab$ (see graph G_1, Figure 3.22). Thus, ab is in the same class as ca, bc, a contradiction. Thus, gi must not be an edge in G. □

Claim 7 *ei must be a nonedge in G.*

Proof: Again, we will use a proof similar to that ofge and gi, a proof by contradiction. Let $ei \in E$, then $\{a, b, c, i, e\}$ induce a house so ca, ab are in the same class and ie is oriented such that i goes to e (see graph G, Figure 3.23).

Now, $fi, dg \notin E$ or the induced subgraphs on $\{a, c, b, f, g, i\}, \{c, b, a, d, e, g\}$, respectively, would induce a pyramid. Also, $ef \notin E$ or the P_4 $iefg$ would have a forbidden orientation.

Then, the edges are forced as follows: $ie \rightarrow fb \rightarrow ce \rightarrow ga$ (see graph G_1, Figure 3.23). However, the P_4 $gade$ then has a forbidden orientation. Thus, ei must not be an edge in G. □

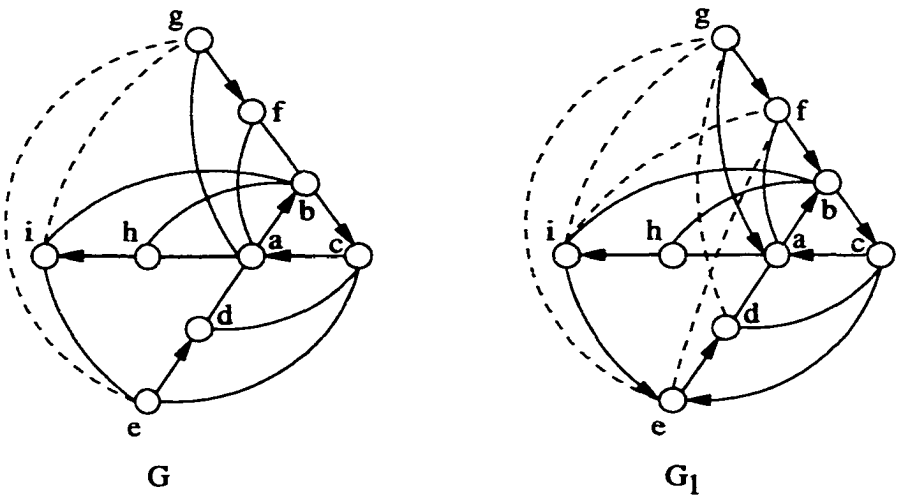


Figure 3.23: ei cannot be an edge

Thus, we have shown that ge, gi, ei cannot be edges in G. Hence, the induced subgraph on $\{a, b, c, e, g, i\}$ in G is the forbidden subgraph called a co-pyramid, a contradiction. Thus, G cannot contain a directed 3-cycle whose edges belong to different equivalence classes. □ (end of proof for Theorem 8)

Chapter 4

Minimal Imperfect Graphs

The notion of perfect graphs was first introduced by Claude Berge in 1960. He defined a graph G to be *perfect* if the chromatic number of H is equal to the clique number of H for every induced subgraph $H \subseteq G$. He also conjectured that perfect graphs are exactly the class of graphs with no induced odd hole or no induced complement of an odd hole, an odd anti-hole. This conjecture, that still remains an open problem, is better known as the *Strong Perfect Graph Conjecture* (or SPGC). An equivalent statement to SPGC is that minimal imperfect graphs are odd holes and odd anti-holes. Minimal imperfect graphs, although are not perfectly orderable themselves, have the property that every induced subgraph of a minimal imperfect graph is perfect. In this chapter we look at minimal imperfect graphs and show a property of hole-free minimal imperfect graphs. In Section 4.1 we shall start by looking at some of the properties and structure of minimal imperfect graphs. In Section 4.2 we will continue by looking at the connectivity results, followed by the main result on hole-free minimal imperfect graphs in Section 4.3.

4.1 Structure of Minimal Imperfect Graphs

In this section, we will review some of the results obtained about the structure of minimal imperfect graphs. In many cases, understanding the structure will help in developing algorithms designed specifically for this class of graphs.

Lovász ([Lov72]) proved that every minimal imperfect graph is *partitionable*. A graph G is partitionable if it has $n(G) = \alpha(G)\omega(G)+1$, $(\alpha(G), \omega(G) \geq 2)$, vertices and for each $v \in V$ the subgraph $G - v$ has a partition into $\alpha(G)$ cliques of size $\omega(G)$ and a partition into $\omega(G)$ stable sets of size $\alpha(G)$.[1]

Also, if G is a minimal imperfect graph, we know that $\chi(G) = \omega(G) + 1$ and for each vertex $v \in V$, $\chi(G - v) = \omega(G) = \omega(G - v)$.

Padberg ([Pad74]) further proved, from Lovász' result, that a minimal imperfect graph G has exactly $n(G)$ ω-cliques and that they are linearly independent. What this means is that the α-stable sets can be listed as follows. First, pick an arbitrary α-stable set S and look at the colouring of $G - s$ for all $s \in S$. The $\alpha(G)\omega(G)+1 = n(G)$ colour classes and S make up every α-stable set. Consequently, $G - v$ is uniquely colourable for all $v \in V$.[2] Later, the same properties were shown to hold for partitionable graphs by Bland, Huang, and Trotter ([BHJ79]).

Olaru ([Ola73]) proved that the minimum degree of a minimal imperfect graph is at least $2\omega - 2$. Later, this result was reproved by Markossian and Karapetian ([MK84]) and independently by Reed ([Ree86]).

---

[1] Definitions for $n(G), \alpha(G), \omega(G)$, and $\chi(G)$ can be found in Chapters 2.2: Bounds on the Colouring Problem.

[2] From this point on, $\alpha(G)$, $\chi(G)$, $\omega(G)$, and $n(G)$ will be denoted α, χ, ω, and n respectively. Parenthesis will be reserved for cases when not refering to G.

4.2 Connectivity of Minimal Imperfect Graphs

Definition 20 *The* components *of a graph G are its maximal connected subgraphs. A* cutset *or* separating set *of a graph $G = (V, E)$ is a set $C \subseteq V$ such that $G - C$ has more than one component. A* minimal cutset *C is a cutset such that no proper subset of C is a cutset. No proper subset of a minimal cutset is a cutset. A graph is k-*connected *if every cutset has at least k vertices. The* connectivity *of G is the minimum size of a cutset. In the case of cliques, the connectivity is one less than the size of the clique.*

Hougardy ([Hou91]) showed that the connectivity number of a minimal imperfect graph G is at least ω. Later, Sebő ([Seb96]) showed that minimal imperfect graphs are $(2\omega - 2)$-connected.

In this section, we will prove a theorem about the structure of the components of a hole-free graph whose minimal cutset is the union of at least two vertex-disjoint cliques.

A *hole* is a chordless cycle of at least five vertices. In proving the following theorems, we will use the property that the graph is hole-free to draw a contradiction. First, we will show a Lemma by Hayward ([Hay85]) that we will use in the proof of our theorem.

Lemma 8 ([Hay85]) *Let G be a hole-free graph with a minimal cutset C such that $G[C]$ is an independent set. Then each component of $G - C$ must have a vertex that is adjacent to all of C.*

Proof: Let G be a hole-free graph and C a minimal cutset of G such that $G[C]$ is an independent set. Also, let $A_1, A_2, \ldots, A_k$ where $k \geq 2$ be the components of $G - C$ and $c_1, c_2, \ldots, c_r$ where $|C| = r$ be the vertices of C. Hayward proved this Lemma by induction on $|C|$. Note that we need only prove the Lemma holds for A_1 and the same proof will apply to all components of G.

Suppose $r = |C| = 1$. This case is trivial since each vertex in C has a neighbour in A_1. Suppose $r = |C| = 2$. We may assume c_1, c_2 have no common neighbour in A_1, otherwise we are done. In particular, c_1, c_2 must each have a unique neighbour $v_{1,p}, v_{2,p}$ in each component A_p. Then, let P be a shortest path from $v_{1,1}$ to $v_{1,2}$ in A_1 and P' a chordless path joining c_1 to c_2 such that all interior vertices of P' belong to A_2. The union, $P \cup P'$ then induces a hole in G which contradicts our hypothesis.

Next, suppose $r = |C| \geq 3$. Since, for each i, $C - c_i$ is a minimal cutset of $G - c_i$, and the induction hypothesis implies that each component A_p has a vertex $u_{i,p}$ that is adjacent to all vertices of $C - c_i$. Consider the three vertices $u_{1,1}, u_{2,1}, u_{3,1} \in A_1$ and the three vertices $c_1, c_2, c_3 \in C$. Each $u_{i,1}$ must be nonadjacent to c_i or we would have a vertex that sees all of C. Also, $u_{i,1}$ must be nonadjacent to $u_{j,1}$ for $i \neq j$ or $c_i, u_{j,1}, u_{i,1}, c_j$ would form a hole with some vertices in C_2. Thus, $\{u_{1,1}, u_{2,1}, u_{3,1}\}$ is a stable set. However, the vertices $c_1, u_{2,1}, c_3, u_{1,1}, c2, u_{3,1}$ induce a hole in G, a contradiction. □

Eschen, Sritharan, Hoàng and Petrick used this result in the proof of the following theorem. We shall say that a set A *sees* a set B if each vertex in B is adjacent to a vertex in A.

Theorem 9 *Let G be a hole-free graph with a minimal cutset C such that $G[C]$ is the union of at least two vertex-disjoint cliques. Then each component of $G - C$ has a clique that sees C.*

Proof: By induction on the number of vertices. Let G be a hole-free graph with a minimal cutset C such that $G[C]$ is the union of vertex-disjoint cliques $K_1, K_2, \ldots, K_r$ with $r \geq 2$. Let the components of $G - C$ be $A_1, \ldots, A_t$ with $t \geq 2$. We only need to prove the Theorem for A_1.

If $|K_i| = 1$ for all i then we are done by Lemma 8. Thus, we may assume that $|K_1| \geq 2$. Let x be a vertex in K_1. Since $C - x$ is a minimal cutset of $G - x$ and the

components of $(G - x) - (C - x)$ are those of $G - C$, we may assume that each A_i has a clique that sees $C - x$. Let W be the clique in A_1 that sees $C - x$. We may assume that x sees no vertex of W, for otherwise we are done. Let $B_x = N(x) \cap A_1$. Let y be a vertex in B_x such that, in A_1, there is a chordless path $P(y, W)$ from y to a vertex in W and the length of $P(y, W)$ is shortest among all paths from a vertex in B_x to a vertex in W. Let R be the clique consisting of y and all the neighbours of y in W (see Figure 4.1). Next, we are going to show that R sees C.

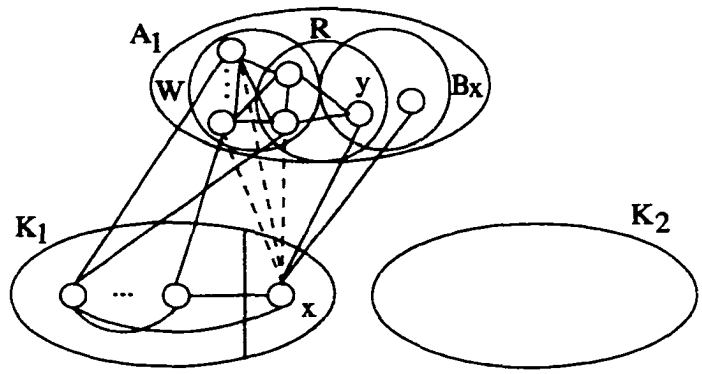


Figure 4.1: Initial configuration

Suppose there is a vertex z in $C - K_1$ that misses y. Vertex z must see a vertex u in W by the choice of W. In A_1 there is a chorldess path P whose endpoints are y and u such that all vertices except y miss x (the existence of $P(y, W)$ implies the existence of P). Since z misses y, in $P \cup \{x, z\}$ there is a chordless path P' of length at least three joining x to z. Lemma 8 implies there is a vertex $v \in A_2$ that sees both x and z. Now, P' and v induce a hole in G, a contradiction (see Figure 4.2). Thus, we know that y sees all vertices of $C - K_1$.

Now, we may assume that there is a vertex $u \in K_1$ that sees no vertex of R, for otherwise we are done. The choice of W implies that the set $B_u = N(u) \cap W \neq \emptyset$. The definition of R implies that y sees no vertex in B. Consider a vertex $k \in K_2$. If k sees

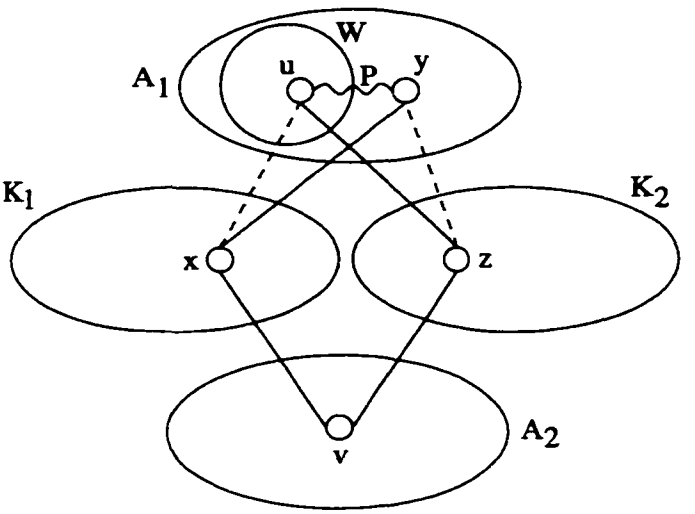


Figure 4.2: y must see all of $C - K_1$

a vertex $b \in B_u$ then the vertices y, x, u, b, k induce a hole in G. Thus, we may assume that k sees no vertex in B_u. The choice of W implies that k sees a vertex $w \in W$. If w sees y then $\{y, x, u, b, w\}$ induce a hole in G; if w misses y then $\{y, x, u, b, w, k\}$ induces a hole in G (see Figure 4.3). □

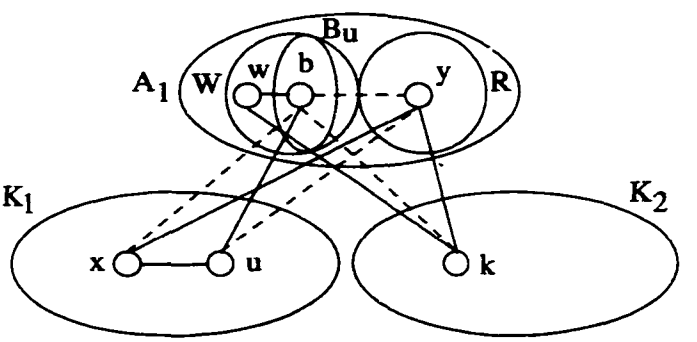


Figure 4.3: Regardless if wy is an edge, there is a hole

We will use this result in the following section.

4.3 Hole-free Minimal Imperfect Graphs

It is known that a minimal cutset C of a minimal imperfect graph G cannot be a clique. Tucker ([Tuc84]) proved that if a minimal cutset C in a minimal imperfect graph G is a stable set, then G must be an odd hole. Fonlupt ([FS90]) then conjectured, as a generalization of Tucker's theorem, that

Conjecture 1 ([FS90]) *Let G be a minimal imperfect graph with a minimal cutset C. If C is the union of two or more disjoint cliques, then G must be an odd hole.*

In this section, we will prove a theorem by Sritharan, Hoàng and Petrick that contributes to the validity of Fonlupt's conjecture.

Theorem 10 *If G is a minimal imperfect graph with no holes, then G cannot have a minimal cutset that is the disjoint union of two cliques.*

Proof: We will prove the theorem by contradiction. In accordance with the theorem hypothesis, let G be a minimal imperfect graph with no holes and C a minimal cutset that is the disjoint union of two cliques K_1 and K_2.

First, we will show that C does not contain a clique of size ω. So, let $|K_1| = \omega$ and let $|K_2|$ be any size. By drawing a contradiction to $|K_1| = \omega$ and not fixing the size of K_2, we will prove that the cutset cannot contain a clique of size ω. Next, let A_1 and A_2 be two disjoint components of $G - C$.

From Theorem 9 we know that component A_i must have a clique W_i that is adjacent to all vertices of C, for $i = 1, 2$. Among all vertices in W_1, let x be the vertex that is adjacent to the most vertices in K_1. WLOG, suppose $x \in A_1$. Now, x cannot be adjacent to all the vertices of K_1 or we would contradict the choice of k_1 since $K_1 \cup \{x\}$ would then be a clique of size $\omega + 1$. So, let y' be a vertex in K_1 that is not adjacent to x.

Then, y' must be adjacent to some vertex y in the clique W_1 since W_1 is adjacent to all of K_1. Consequently, y must not be adjacent to some $x' \in K_1$ that x is not adjacent to; otherwise, y would contradict the fact that x is adjacent to the most vertices in K_1 since y would be adjacent to all neighbours of x in K_1 and also y' (see Figure 4.4).

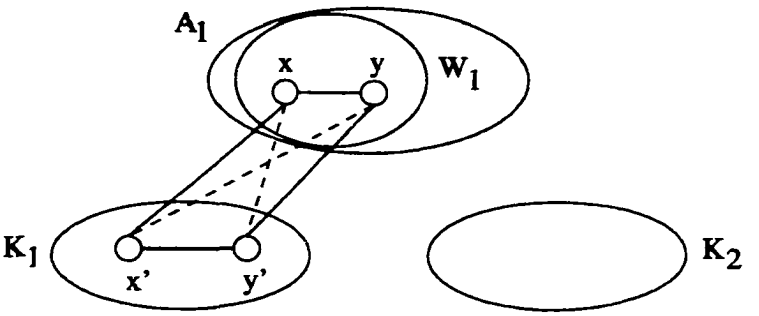


Figure 4.4: x is adjacent to the most vertices in K_1

Claim 8 *For all $a \in K_2$, a must see both x and y.*

Proof: Let $a \in K_2$. Then, first we will show that a cannot see only one of x or y.

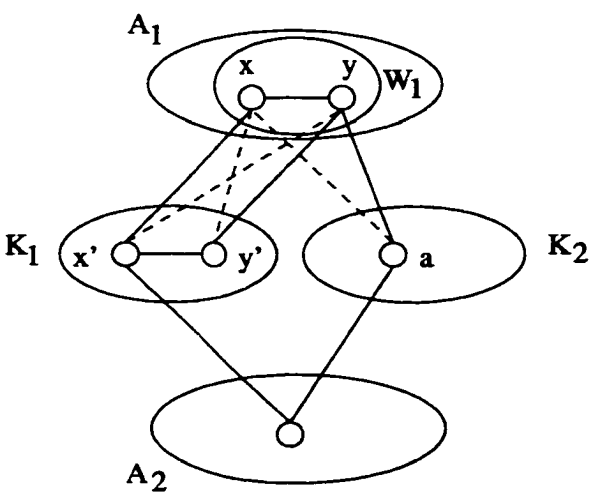
Case 1: Suppose that a is adjacent to y and nonadjacent to x. Then $ayxx'$ forms a P_4 and by taking the shortest path from a to x' going through A_2, we would have a hole of at least five vertices (see Figure 4.5).

Case 2: Suppose that a is adjacent to x and nonadjacent to y. Then $axyy'$ forms a P_4 and by taking the shortest path from a to y' going through A_2, we would create a hole of at least five vertices. So, a must either be adjacent to both x and y or nonadjacent to both x and y (see Figure 4.5).

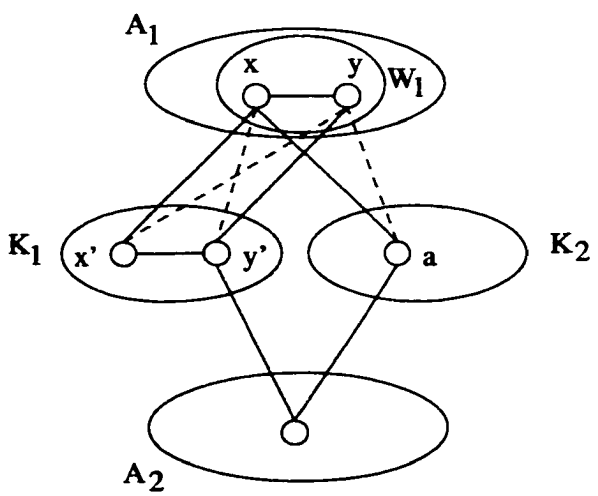
Fact 2 *Let G be a hole-free graph and let x, y be two non-adjacent vertices in a minimal cutset C of G. Then each component of $G - C$ has a vertex that is adjacent to both x and y.*

Proof: Let G be a hole-free graph with a minimal cutset C where $x, y \in C$ and x is non-adjacent to y. Let A_1, A_2 be two components of $G - C$. WLOG suppose that A_1 does not have a vertex adjacent to both x and y. Since C is a minimal cutset of G, x and y must be adjacent to some vertices a, b respectively in A_1. Then, by finding the shortest path from a to b in A_1 and the shortest path from y to x in A_2, we have a chordless cycle of at least five vertices, a contradiction. □

Next, suppose a misses both x, y. Then, since the clique W_1 sees all of C, there exists some vertex $z \in W_1$ that is adjacent to a. If z is not adjacent to y' (or x') then $azyy'$ ($azxx'$) forms a P_4 and by Fact 2, this P_4 and some vertex in A_2 induces a hole (see Figure 4.6). So, z must be adjacent to y'. Since z is adjacent to y', there must exist a vertex $w' \in K_1$ that is adjacent to x but not to z. Then $azxw'$ forms a P_4 and by Fact 2, this P_4 and some vertex in A_2 induces a hole (see Figure 4.6). Therefore, every vertex $a \in K_2$ is adjacent to both x and y. □



Case 1: a is adjacent to y but not x.



Case 2: a is adjacent to x but not y

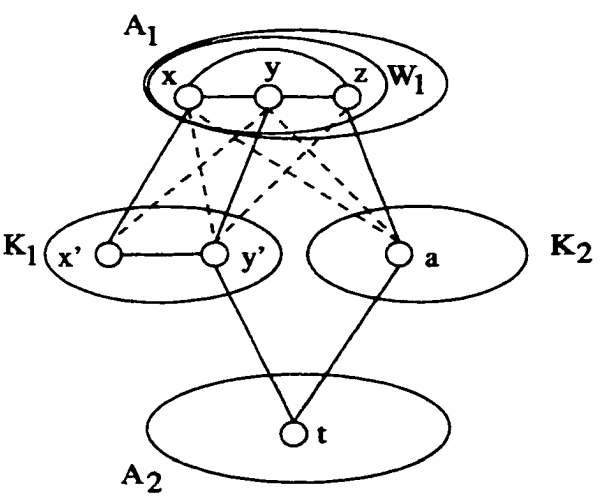
Figure 4.5: A hole exists in both Case 1 and Case 2

Note that Claim 8 also shows that when $|K_1| = \omega$, $|K_2| \leq \omega - 2$.

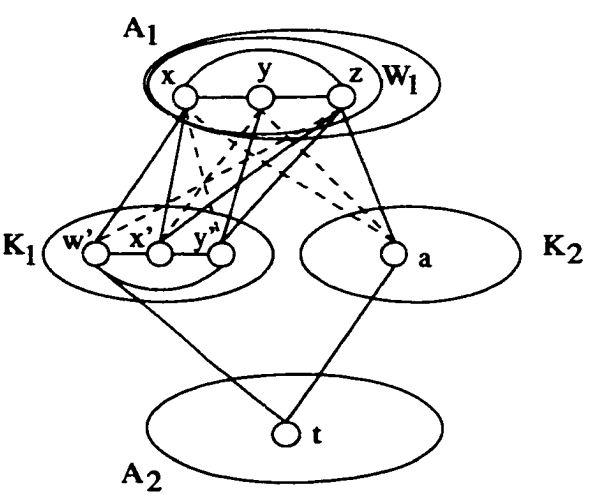
Similar to the structure of A_1 and depicted in Figure 4.7, is a structure on the vertices $xx, yy \in W_2$ of A_2 where xx, yy correspond to the vertices x, y in W_1. Let z have the most neighbours in K_1 of $\{x, y, xx, yy\}$ and WLOG let $x \in W_1$.

Pick any $c \in K_2$. By Claim 8, c must see z. Then, we know that there exists a $b \in K_1$ that is not adjacent to z or we would have a clique of size $\omega + 1$. Let $d \in W_2$ be a neighbour of b and $a \in K_1$ be a neighbour of z that d misses. The vertex a must exist since z has the most neighbours in K_1 and d sees b (a vertex that z misses).

Now, if c sees d, then $zcdbaz$ is a hole. So, cd cannot be an edge, and c must see some vertex $e \in W_2$ (see Figure 4.7). e must see b or by Fact 2, $cedb$ and some vertex in A_1 would induce a hole. Then, there must be a vertex $f \in K_1$ that sees z but misses e since z has the most neighbours in K_1 and e sees the vertex b that z misses (see Figure 4.7). However, then $zcebfz$ is a hole. Hence, it is impossible for any of the cliques that



Case 1: z misses y' and forms hole azyy't



Case 2: z sees y' but misses w' and forms hole azxw't

Figure 4.6: A hole is created if $a \in K_2$ misses x

make up the minimal cutset C to be of size ω.

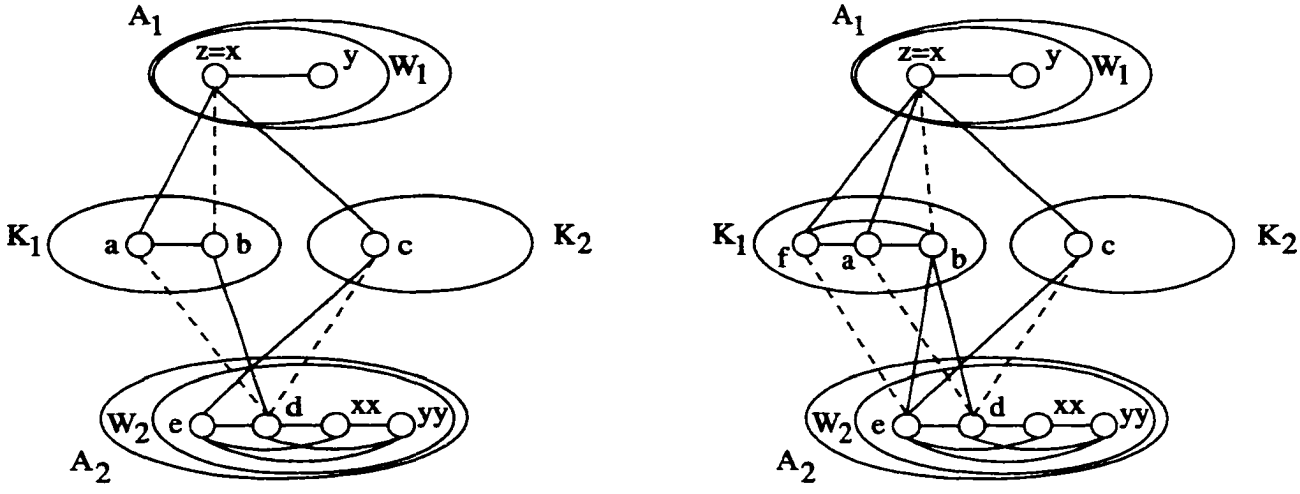


Figure 4.7: K_1 cannot be size ω

Next, we will show that when each of K_1, K_2 is of size less than ω then the graph G is not a minimal imperfect graph. We will show this by using a colouring argument to show that G can be coloured using ω colours.

First, we will look at $G_1 = G[A_1 \cup C]$ and $G_2 = G[A_2 \cup C]$. As G_1 and G_2 are proper induced subgraphs of the minimal imperfect graph G, they both are perfect, and hence can be optimally coloured with ω colours. Here, we will look at two cases.

Case 1: Suppose that colour ω is missing from the cutset C in both optimal colourings of G_1 and G_2. In an optimal colouring of any perfect graph, every colour class intersects every maximum clique. Since, G_1, G_2 are perfect, the set S of vertices with colour ω in both graphs forms a colour class and also intersect every maximum clique in their respective graphs. Then, if we remove all of the vertices of colour ω, the graph $G - S$ no longer contains a clique of size ω. $G - S$ is also perfect and can be optimally coloured in $\omega - 1$ colours since that is now the size of the largest clique. The set S is an independent

set of G and so G is ω-colourable.

Claim 9 *For all independent sets S of a minimal imperfect graph G, there exists a clique $K(S)$ of size ω such that $S \cap K(S) = \emptyset$.*

Proof: Suppose that the claim is false. Then $\omega(G - S) = \omega(G) - 1$ and $\chi(G - S) = \omega(G) - 1$. Therefore, $\chi(G) = \omega(G)$ which contradicts the definition of minimal imperfect graphs. □

Case 2: Suppose that the optimal colouring of G_1 (or G_2) has ω colours appearing in C. Then, we will show how to recolour each graph so that only $\omega - 1$ colours appear in C. Then Case 2 will revert back to Case 1 which shows a contradiction.

Assume in the optimal colouring of G_1 , ω colours appear in C. Since $|K_1| \leq \omega - 1$ and $|K_2| \leq \omega - 1$ and $C = K_1 \cup K_2$ uses all ω colours, each of K_1, K_2 must be missing a different colour. WLOG, say K_1 is missing colour 2 and K_2 is missing colour 1. Then, there exists a vertex $x \in K_1$ that is assigned colour 1 and a vertex $y \in K_2$ that is assigned colour 2. No other vertex in K_1 is assigned colour 1 and similarly, no other vertex in K_2 is assigned colour 2. Now, consider the subgraph H induced in G_1 by the vertices coloured with either colour 1 or colour 2.

Claim 10 *x and y belong to different components of H.*

Proof: Suppose that x and y belonged to the same component of H. Then, consider a shortest path P in H connecting x to y. Path P would be a colour-alternating-path where $|P| = 2k + 1$ for $k \geq 1$ and where $P \cap C = \{x, y\}$ (see Figure 4.8). Then, by Fact 2, P and some vertex in A_2 induce an odd hole in G, a contradiction. Hence, x and y must be in different components of H. □

Since, x and y belong to different components of H, we can recolour the component that x belongs to. By interchanging the colours 1 and 2 in this component, $x \in K_1$ is

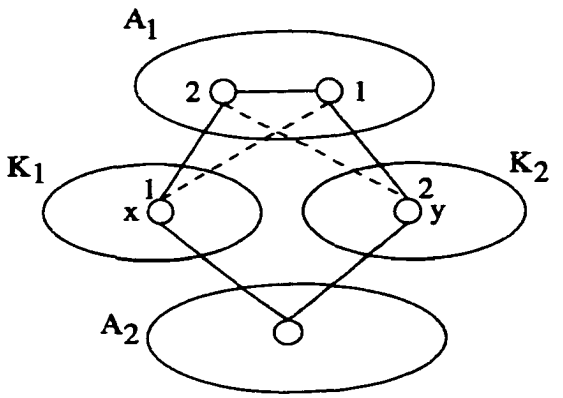


Figure 4.8: P and a common neighbour to x and y form a hole

recoloured with 2 and K_1 no longer has a vertex of colour 1. Also, K_2 remains unaffected by the recolouring, so, only $\omega - 1$ colours appear in C in an optimal colouring of G_1.

Similarly, this procedure can be performed on G_2 if C is coloured using all ω colours in an optimal colouring. Then, the colour classes of G_1, G_2 can be switched by simple relabelling so the colour ω is missing from C and Case 2 is then reduced to Case 1. Hence, G can be optimally coloured in ω colours which violates the hypothesis that G is a minimal imperfect graph. □

Chapter 5

Conclusions and Future Research

In this chapter, we discuss the work presented in this thesis along with future work for both short and long term goals.

5.1 One-In-One-Out Graphs

The problem of deciding whether one-in-one-out graphs can be recognized in polynomial time and finding a polynomial time recognition algorithm remains open. Determiniming this, is the main goal, however, it appears that this problem is very difficult and hence, is one of the long term goals. It would also be nice to find a subgraph characterization of this class and others created from the various types of P_4 orientations, such as P_4-simplicial, GCB, and P_4 comparability graphs.

In the short term, determining whether pyramid-free graphs with Type 2 and Type 3 orientations are acyclic would be a good interim result. If this is the case, it is likely that a polynomial time recognition algorithm could be developed for this class. The next step in the research would be to use a similar method to determine if a directed 4-cycle could be found and then to try and write a proof for the general case or a proof by induction on

an n-cycle. As with one-in-one-out graphs, pyramid-free graphs with Type 2 and Type 3 acyclic orientations would then be able to be coloured in polynomial time using the greedy colouring algorithm. By find a polynomial recognition algorithm, we could then also create a robust algorithm that would run in polynomial time.

5.2 Graph Generation

Graph generation has long been a problem discussed and researched by many. Currently, programs such as *nauty* by McKay that determine the automorphism group of a graph are being researched and developed. These programs are able to generate graphs with a larger number of vertices in a shorter amount of time. The problem of graph generation is a difficult one considering the rate at which the sequence of graphs on n vertices grows. The number of simple connected graphs on 1 vertex to 10 vertices are as follows 1, 1, 2, 6, 21, 112, 853, 11 117, 261 080, 11 716 571. Although being able to check all for certain properties in all of the graphs under say 20 vertices would be quite an asset, it does not appear that this will be possible for quite some time. In the case of one-in-one-out graphs, not finding a graph that has a directed cycle when looking at all of the graphs on less than 20 vertices would strongly indicate that a polynomial time algorithm exists. Finding counter examples to many conjectures could also be found using this method. Although the generation time of graphs is decreasing in terms of *big-O*, the hardware required for larger graphs is still not currently available.

5.3 Minimal Imperfect Graphs

The first theorem we proved on a hole-free graph G is that if G has a minimal cutset C that is the union of vertex-disjoint cliques, then each component of $G - C$ must have

a clique that sees all of C. This result was then used to prove another theorem that shows minimal imperfect graphs with a minimal cutset that is the union of two cliques must contain a hole. This theorem does not prove Fonlupt's conjecture that all minimal imperfect graphs with a minimal cutset that is the union of more than one disjoint clique, must be an odd hole. However, it does contribute to the instance of Fonlupt's conjecture when the minimal cutset is the union of two disjoint cliques. Also, the first theorem proves a property of hole-free graphs without specifying the number of disjoint cliques that make up C that could be of use in proving other instances or a general case of Fonlupt's conjecture.

Bibliography

[AH77a] K. Appel and W. Haken. Every planar map is 4-colorable-1: Discharging. *Ill. J. Math.*, 21:429–490, 1977.

[AH77b] K. Appel and W. Haken. Every planar map is 4-colorable-2: Reducibility. *Ill. J. Math.*, 21:491–567, 1977.

[BHJ79] R. G. Bland, H. C. Huang, and L. E. Trotter Jr. Graphical properties reated to minimal imperfection. *Discrete Math.*, 27:11–22, 1979.

[BL76] K. S. Booth and G. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *J. Computer and System Science*, 13:335–379, 1976.

[CH94] A. Cournier and M. Habib. A new linear algorithm for modular decomposition. *Lecture Notes in Computer Science*, 787:68–84, 1994.

[Chv84] V. Chvátal. Perfectly ordered graphs. In Berge C. and V. Chvátal, editors, *Topics on perfect graphs*, pages 63–65. North-Holland, Amsterdam, 1984.

[CPS85] D. G. Corneil, Y. Perl, and L. K. Stewart. A linear recognition algorithm for cograph. *SIAM J. Computing*, 14:926–934, 1985.

[CW90] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Computation*, 9:1–6, 1990.

[DGM97] E. Dahlhaus, J. Gustedt, and R. R. McConnell. Efficient and practical modular decomposition. In *Proceedings of the 8th Symposium on Discrete Algorithms*, pages 26–35, 1997.

[Dir61] G. A. Dirac. On rigid circuit graphs. *Abh. Math. Sem. Univ. Hamburg*, 25:71–76, 1961.

[FG65] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific J. Math.*, 15:835–855, 1965.

[FRRT99] J. F. Fouquet, F. Roussel, P. Rubio, and H. Thuillier. New classes of perfectly orderable graphs. University of Orleans, Technical Report. To appear in Discrete Math., 1999.

[FS90] J. Fonlupt and A. Sebő. On the clique rank and the coloration of perfect graphs. *Mathematical Programming Socienty*, 1990.

[Gal67] T. Gallai. Transitiv orientierbare Graphen. *Acta Math. Acad. Sci. Hungar.*, 18:25–66, 1967.

[GH64] P. C. Gilmore and A. J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canad. J. Math.*, 16:539–548, 1964.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman, San Fransisco, 1979.

[GM82] M. C. Golumbic and C. L. Monma. A generalization of intervals with tolerances. *Congressus Numerantium*, 35:321–331, 1982.

[GMJ84] M. C. Golumbic, C. L. Monma, and W. T. Trotter Jr. Tolerance graphs. *Discrete Applied Math.*, 9:157–170, 1984.

[Gol80] M. C. Golumbic. *Algorithmic graph theory and perfect graphs*. Academic Press, New York, 1980.

[Gol84] M. C. Golumbic. Algorithmic aspects of perfect graphs. In Berge C. and V. Chvátal, editors, *Topics on perfect graphs*. North Holland, Amsterdam, 1984.

[Hay85] R. Hayward. Weakly triangulated graphs. *J. Combin. Theory (B)*, 39:200–209, 1985.

[Her90] A. Hertz. Bipolarizable graphs. *Discrete Math.*, 81:25–32, 1990.

[HK88] C. T. Hoàng and N. Khouzam. On brittle graphs. *J. Graph Theory*, 12(3):391–404, 1988.

[HMN98] C. T. Hoàng, F. Maffray, and M. Noy. A characterization of P_4-indifference graphs. to appear in J. Graph Theory, 1998.

[HMOP92] C. T. Hoàng, F. Maffray, S. Olariu, and M. Preissmann. A charming class of perfectly orderable graphs. *Discrete Math.*, 102:67–74, 1992.

[HMP91] C. T. Hoàng, F. Maffray, and M. Preissmann. New properties of perfectly orderable graphs and strongly perfect graphs. *Discrete Math.*, 98:161–174, 1991.

[Hoà96] C. T. Hoàng. On the complexity of recognizing a class of perfectly orderable graphs. *Discrete Applied Math.*, 66:219–226, 1996.

[Hou91] S. Hougardy. Perfekte graphen diplomarbeit. Universität Bonn, Germany, 1991.

[HPV] M. Habib, C. Paul, and L. Viennot. Linear time recognition of P_4-indifference graphs. INRIA-Rocquencourt, Technical Report, 1999.

[HR89a] C. T. Hoàng and B. A. Reed. Recognizing P_4-comparability graphs. *Discrete Math.*, 74:173–200, 1989.

[HR89b] C. T. Hoàng and B. A. Reed. Some classes of perfectly orderable graphs. *J. Graph Theory*, 13(4):445–463, 1989.

[LB62] C. G. Lekkerkerker and J. C. Boland. Representation of a finite graph by a set of intervals in the real line. *Fund. Math.*, 51:45–64, 1962.

[Lov72] L. Lovász. A characterization of perfect graphs. *J. Combin. Theory*, 13:95–98, 1972.

[MK84] S. E. Markossian and I. A. Karapetian. On critically imperfect graphs. *Prikladnaia Matematika*, 1984. Erevan University.

[MP90] M. Middendorf and F. Pfeiffer. On the complexity of recognizing perfectly orderable graphs. *Discrete Math.*, 80(3):327–333, 1990.

[MS94] R. M. McConnell and J. P. Spinrad. Linear time modular decomposition and efficient transitive orientation of comparability graphs. In *5th ACM-SIAM Symposium of Discrete Algorithms*, pages 536–545, 1994.

[MS97] R. M. McConnell and J. P. Spinrad. Linear time transitive orientation. In *8th ACM-SIAM Symposium of Discrete Algorithms*, pages 19–25, 1997.

[MS99] R. M. McConnell and J. P. Spinrad. Modular decomposition and transitive orientation. 1999.

[Ola73] E. Olaru. Zur characterisierung perfekter graphen. *Elektronische Informationsverarbeitung und Kybernetik*, 9:543–548, 1973.

[Pad74] M. Padberg. Perfect zero-one matrices. *Math Programming*, 6:180–196, 1974.

[PdWM86] M. Preissmann, D. de Werra, and N. V. R. Mahadev. A note on super brittle graphs. *Discrete Math.*, 61:259–267, 1986.

[Ree] B. A. Reed. Unpublished manuscript.

[Ree86] B. A. Reed. A semi-strong perfect graph theorem. Ph.D. thesis, McGill University, Quebec, 1986.

[RS] T. Rasche and K. Simon. On the P_4-component of graphs. Institute of Theoretical Computer Science, ETH Zürich, Technical Report, 1998.

[RTL76] D.J. Rose, R.E. Tarjan, and G.S. Leuker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Computing*, 5:266–283, 1976.

[Sch91] A. A. Schäffer. Recognizing brittle graphs: remarks on a paper of hoàng and khouzam. *Discrete Applied Math.*, 31:29–35, 1991.

[Seb96] A. Sebő. The connectivity of minimal imperfect graphs. *J. Graph Theory*, 23:77–85, 1996.

[SJ] J. Spinrad and J. Johnson. Brittle and bipolarizable graph recognition. Vanderbilt University, Computer Science Department, Technical Report, 1998.

[Tuc84] A. Tucker. Uniquely colorable perfect graphs. *Discrete Math.*, 44:187–194, 1984.

[WP67] D. J. A. Welsh and M. B. Powell. An upper bound on the chromatic number of a graph and its application to timetabling problems. *Computer J.*, 10:85–87, 1967.