Diss. ETH No. 13119

Generalized Modular Decompositions and the Recognition of Classes of Perfectly Orderable Graphs

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZURICH

for the degree of
Doctor of Technical Sciences

presented by
THOMAS RASCHLE
Dipl. Informatik-Ing. ETH
born 1.5.1960
citizen of Bütschwil, Switzerland

accepted on the recommendation of
Prof. Dr. K. Simon, examiner
Prof. Dr. A. Hertz, co-examiner

1999

Acknowledgment

First of all, I would like to thank Professor Klaus Simon for introducing me to the fascinating theory of perfect graphs and for giving me the opportunity to do research in this field. Without his support and his encouragement, this thesis would not exist. I am also grateful to Professor Alain Hertz for his acceptance to be my co-examiner, for his careful reading and for the many suggestions which helped improving this text.

Many thanks go to Dr. Andrea Sterbini for introducing me to Ma's method of recognizing threshold dimension two and for the agreeable collaboration over the past years. Many thanks also go to Professor Uri Peled and Professor N.V.R. Mahadev for the enlightening discussions on threshold graphs and for their interest in my work.

It is also a pleasure to thank my officemate Jakob Magun: Discussing graph theory and many other things with him was always great fun. Further thanks go to all my friends in the Graph Theory Group at the ETH for creating an open, stimulating and relaxed atmosphere. This includes Davide Crippa, Daniele Degiorgi, Nicola Galli, Bernhard Seybold and Paul Trunz.

Last but not least, I wish to thank Doris for bearing with me when I simply could not stop talking about graph problems.

Abstract

A great many problems are naturally formulated in terms of objects and connections between them and are therefore best modeled as graphs. To solve these problems on a computer, efficient algorithms are required. Unfortunately, there are many graph problems for which no efficient algorithm has been found. Classical examples are the determination of the clique number and the calculation of the chromatic number. These examples are NP-complete, and it is widely believed that no NP-complete problem can be solved efficiently.

On the other hand, many graphs arising from real world problems have a special structure, which often makes solving the problem easier. For instance, the clique number and the chromatic number can be found in linear time if the graph is perfectly orderable and a perfect order is given. Recognizing perfectly orderable graphs, however, is NP-complete.

In this thesis, new algorithms for recognizing subclasses of perfectly orderable graphs are developed. To begin with, a recognition algorithm for triangulated graphs is presented which is linear in the size of the complement. Next, classical results on comparability graphs are reviewed. These results are then generalized in two ways.

First, modules are generalized such that divide and conquer methods are still applicable to solve graph problems. In particular, two types of generalized modules are further investigated. These investigations lead to a new unique graph decomposition, which refines the modular decomposition. Second, GALLAI's results on the P_3-structure are translated into analogous results on the P_4-structure. The arising theorems are then used to design efficient algorithms for recognizing and orienting P_4-comparability graphs and similar classes of perfectly orderable graphs.

Another part of this thesis deals with the recognition of graphs with threshold dimension two. In 1982, IBARAKI AND PELED conjectured that a graph has threshold dimension two if and only if its conflict graph is bipartite. A proof of this conjecture is given based on a theorem on generalized modules. Furthermore, a linear time algorithm for recognizing cobithreshold graphs is presented.

Zusammenfassung

Viele Probleme sind durch Beziehungen zwischen Objekten charakterisiert und lassen sich deshalb sehr gut als Graphen modellieren. Zur Lösung dieser Probleme auf dem Computer werden effiziente Algorithmen benötigt. Leider wurde für viele Graphenprobleme bis heute kein effizienter Algorithmus gefunden. Klassische Beispiele dafür sind die Bestimmung der Cliquenzahl und die Berechnung der chromatischen Zahl. Diese Beispiele sind NP-vollständig, und es wird angenommen, dass kein NP-vollständiges Problem effizient gelöst werden kann.

Viele sich aus praktischen Anwendungen ergebende Graphen haben allerdings eine spezielle Struktur, die das Lösen des Problems oft einfacher macht. Beispielsweise kann die Cliquenzahl und die Färbungszahl in linearer Zeit gefunden werden, falls der gegebene Graph perfekt orientierbar ist und eine perfekte Ordnung gefunden ist. Die Erkennung perfekt orientierbarer Graphen ist aber wiederum NP-vollständig.

In dieser Arbeit werden neue Algorithmen zur Erkennung von Unterklassen perfekt orientierbarer Graphen entwickelt. Zunächst wird ein Erkennungsalgorithmus für Dreiecksgraphen vorgestellt, dessen Laufzeit linear in der Grösse des Komplements ist. Danach werden klassische Resultate über transitiv orientierbare Graphen besprochen. Diese Resultate werden dann auf zwei Arten verallgemeinert.

Erstens werden Module so verallgemeinert, dass Teile-und-Herrsche-Methoden zur Lösung von Graphenproblemen immer noch anwendbar sind. Zwei Typen von verallgemeinerten Modulen werden genauer untersucht. Diese Untersuchungen führen auf eine neue eindeutige Graphenzerlegung, welche eine Verfeinerung der Modulzerlegung darstellt. Zweitens werden GALLAI's Resultate über die P_3-Struktur in analoge Resultate bezüglich der P_4-Struktur übersetzt. Die sich daraus ergebenden Sätze werden unter anderem zur Konstruktion effizienter Algorithmen zur Erkennung und Orientierung P_4-transitiv orientierbarer Graphen benutzt.

Ein weiterer Teil dieser Arbeit behandelt die Erkennung von Graphen mit Threshold Dimension zwei. Bereits 1982 äusserten IBARAKI UND PELED die Vermutung, dass ein Graph genau dann Threshold Dimension zwei hat, wenn sein Konfliktgraph zweifärbbar ist. Diese Vermutung wird basierend auf einem Satz über verallgemeinerte Module bewiesen. Auch wird ein linearer Algorithmus zur Erkennung von Cobithresholdgraphen vorgestellt.

Contents

Chapter 1

Introduction

Graph theory was founded by EULER in 1736 when he solved the Königsberger Bridge Problem, a famous problem of his days. In Köngisberg, there were two islands linked to each other and to the banks of the Pregel River by seven bridges as depicted in Figure 1.1. The problem was to start at a given land area, walk over each bridge precisely once and return to the starting point.

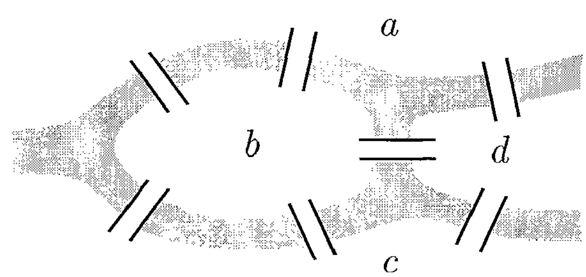


Figure 1.1: *The map of a park in Königsberg,* 1736.

Euler modeled the situation as a graph by replacing each land area with a vertex and each bridge with an edge that joined the corresponding vertices, see Figure 1.2. Rather than solving the problem for this specific graph, he developed a criterion for any given graph to be so traversable; namely, that the graph is connected and every vertex is incident to an even number of edges.

Since then, graphs have been studied intensively and graph theory has become a major branch of combinatorial mathematics. This is due

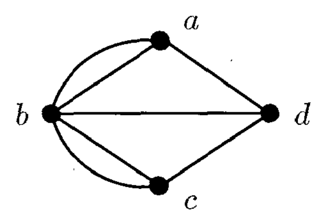


Figure 1.2: *The graph of the Königsberg Bridge Problem.*

to the fact that a great many problems are naturally formulated in terms of objects and connections between them and are therefore best modeled as graphs.

With the availability of computers, the interest in efficient algorithms for solving graph problems grew rapidly. The most common measure of the efficiency of an algorithm is the *worst case complexity*. It is a function in the size of the input and gives an upper bound for the number of operations that the algorithm performs on any input of the corresponding size.

The notion of complexity also led to a classification of problems into complexity classes [44, 63]. The most important complexity classes are P and NP, the class of problems for which polynomial algorithms exist on a deterministic and nondeterministic Turing machine, respectively. To this day, no proof of P $\neq$ NP has been found, although it is widely believed that P $\neq$ NP. Indeed, the security of most currently used cryptosystems is based on this assumption [70].

The hardest problems in NP are called NP-complete. They are defined to be those problems for which the existence of a polynomial algorithm would imply a polynomial algorithm for every other problem in NP. Unfortunately, many important graph problems are NP-complete. Classical examples are the calculation of the clique number or the chromatic number. Recent results have shown that even the approximation of these numbers up to certain factors is NP-complete [50, 4].

On the other hand, many graphs that arise from real world problems have a special structure. This special structure makes it often possible to solve problems in polynomial time that are NP-complete in general. Famous examples of such graphs are planar graphs and perfect graphs: Planar graphs can be drawn in the plane without crossings, and perfect graphs have only subgraphs whose clique number is equal to the chromatic number.

Whereas good recognition and optimization algorithms are known for planar graphs, the situation is less fortunate in case of perfect graphs. It is not even known whether the recognition of perfect graphs is in NP or not. Moreover, most NP-complete problems remain NP-complete when restricted to perfect graphs. A famous exception is the computation of the clique number and the chromatic number. In 1981, GRÖTSCHEL ET AL. [29] found a polynomial algorithm for computing a maximum clique and a minimum coloring for perfect graphs. Unfortunately their algorithm, the only known to date, makes use of the ellipsoid method and is therefore of mainly theoretical interest.

In search of certificates for perfection, BERGE [6, 7] made two conjectures concerning perfect graphs. The first one, proved by LOVÁSZ [48] in 1972 and since then called the *Perfect Graph Theorem*, states that a graph is perfect if and only if its complement is perfect. A slightly stronger version of this theorem, the *Semi-Strong Perfect Graph Theorem*, was proved by REED [69] in 1987 and asserts that the perfection of a graph solely depends on a derived hypergraph whose edges are the four element sets that induce a P_4 (chordless path on four vertices).

BERGE's second conjecture states that a graph is perfect if and only if it does not contain an odd hole or an odd antihole, that is, an odd chordless cycle of length greater than three or its complement. This conjecture, famous under the name *Strong Perfect Graph Conjecture*, is one of the most outstanding open problems in graph theory. The validity of the Strong Perfect Graph Conjecture, however, would not imply an easy method to recognize perfect graphs: BIENSTOCK [8] has shown that it is NP-complete to test whether a graph has an odd hole, so it might be difficult to test whether a graph or its complement has an odd hole.

One possible way to overcome the difficulty in recognizing and optimizing perfect graphs is to consider large classes of perfect graphs. The Strong Perfect Graph Conjecture suggests that promising candidates are graphs defined by properties not satisfied by graphs with odd holes or odd antiholes. Moreover, in view of REED's Semi-Strong Perfect Graph Theorem, natural classes of perfect graphs can be defined by properties associated with the P_4-structure. An example of such a class of graphs are perfectly orderable graphs.

Perfectly orderable graphs were introduced by CHVÁTAL [10] in 1984 as those graphs which admit a perfect orientation, i.e., an acyclic orientation such that no P_4 $abcd$ is oriented $a \to b$ and $c \leftarrow d$. He showed that a maximum clique and a minimum coloring can be found in linear time if a perfect orientation is given. This nice optimization behavior, however, is in stark contrast to the difficulties with the recognition. In 1990, MIDDENDORF AND PFEIFFER [56] proved that it is NP-complete to test whether a graph has a perfect orientation.

A class of perfectly orderable graphs that can be recognized in polynomial time are comparability graphs. They are defined as those graphs which admit an acyclic transitive orientation, i.e., an acyclic orientation such that no P_3 abc is oriented $a \to b$ and $b \to c$. Consequently, the orientation of one edge in a P_3 implies the orientation of the other edge in the same P_3. The equivalence classes of the transitive closure of this P_3-relation, called P_3-classes for short, were first studied by GHOUILA-HOURI [24]. He showed that a graph is a comparability graph if and only if its P_3-classes are transitively orientable. His proof relied on the fact that the set of vertices incident to edges in the same P_3-class is a module, that is, a vertex set such that vertices not in the set are adjacent to every or no vertex in the set.

A penetrating study of modules and the P_3-structure was conducted by GALLAI [23]. He showed that maximal nontrivial modules are disjoint whenever the given graph and its complement are connected. Based on this result, he proposed a unique graph decomposition, nowadays known as modular decomposition. Furthermore, he observed that if a graph and its complement are connected, then all edges not contained in maximal nontrivial modules belong to the same P_3-class. This observation leads to simple algorithms for computing the modular decomposition and for recognizing and orienting comparability graphs [57].

Besides its connection with comparability graphs, the modular decomposition is interesting because it allows the application of divide and conquer methods to solve graph problems [59, 58]. In Chapter 4, we generalize modules in a way that still admits the application of divide and conquer strategies. We then focus on two types of generalized modules, which we call *bipartite modules* and *split modules*. Those generalized modules are used to obtain a new unique decomposition which generalizes the decompositions found by BABEL AND OLARIU [5] and by RASCHLE AND SIMON [67].

In Chapter 5, our results on split modules are applied to P_4-comparability graphs. P_4-comparability graphs were introduced by HOÀNG AND REED [39] as those graphs which admit an acyclic orientation such that every P_4 is transitively oriented. From this definition, it follows that P_4-

comparability graphs are perfectly orderable and that the orientation of one edge in a P_4 implies the orientation of the other edges in the same P_4. Thus the crucial structure here are the P_4-classes, that is, the equivalence classes of the transitive closure of the relation between edges in which two edges are in relation if they belong to the same P_4.

Together with P_4-classes, we study the relation between P_4s in which two P_4s are in relation if they have three common vertices. In this thesis, the equivalence classes of the transitive closure of the above relation between P_4s are called strong P_4-components[1]. Several GALLAI-type results on P_4-components are obtained. In particular, we generalize CHVÀTAL's theorem [12] on 3-chains by showing that a graph without nontrivial modules and split modules has at most one strong P_4-component. Our findings are then used to compute the decomposition of a graph into maximal nontrivial split modules and to design an improved algorithm for orienting P_4-comparability graphs. As a further application, we show that a perfect orientation can be found by substituting two (marker) vertices for split modules. This substitution yields a general recognition algorithm for many classes of perfectly orderable graphs, including HERTZ' bipartable graphs [35].

An important subclass of bipartable graphs are graphs with threshold dimension two. The threshold dimension of a graph introduced by CHVÀTAL AND HAMMER [13] is the smallest integer k such that the graph can be written as the (edge-)intersection of k threshold graphs, and a threshold graph is a graph without induced P_4, C_4 and $2K_2$. Threshold graphs and the threshold dimension have applications in 0-1 programming, in psychology and in the synchronization of parallel processes [53].

In 1983, YANNAKAKIS [76] showed that it is NP-complete to test whether an arbitrary graph has threshold dimension k for all $k \geq 3$. The case $k = 2$ was open for over a decade. Indeed, it was widely believed that this problem is also NP-complete. Recently, however, RASCHLE AND SIMON [66] found an $O(|V|^4)$ time algorithm for recognizing graphs with threshold dimension two. Their algorithm represents a constructive proof of a conjecture made by IBARAKI AND PELED [41] which states that recognizing graphs with threshold dimension two is equivalent to testing whether an associated conflict graph is bipartite. In Chapter 6, we present an improved version of RASCHLE AND SIMON's algorithm based on a new structure theorem concerning special cobipartite and

[1] they are the connected components of the 3-overlap graph defined in [38]

split modules.

A large subclass of the complement of graphs with threshold dimension two are cobithreshold graphs. They are the union of two threshold graphs such that every clique in the union is a clique in one of the two threshold graphs. Cobithreshold graphs were first investigated by MAHADEV AND HAMMER [31] in connection with biregular boolean functions. MAHADEV AND HAMMER also found an $O(|E|^2)$ recognition algorithm for this class of graphs. In Chapter 7, we analyze the structure of cobithreshold graphs and give a linear time recognition algorithm.

Chapter 2

Preliminaries

This chapter provides the background for the subsequent chapters. In the first section, we introduce our graph theoretic terminology. Regarding undirected graphs, it is compatible with BONDY AND MURTY [9], GOLUMBIC [27] and MAHADEV AND PELED [53]. For directed graphs, however, we use a more intuitive notation. For instance, we write $\vec{G} = (V, \vec{E})$ for a directed graph and $v \rightarrow w$ or $w \leftarrow v$ for an edge from v to w.

In the second section, we present classical results on perfect graphs and discuss open problems in connection with the recognition and optimization of this class of graphs. We then focus on perfectly orderable graphs and review CHVÀTAL's result on their nice optimization behavior. Since the recognition of perfectly orderable graphs is NP-complete, one is naturally interested in finding classes of perfectly orderable graphs that can be recognized in polynomial time. As a first example of such a class of graphs, we discuss triangulated graphs.

Finally, the last section provides the algorithmical background which is needed to obtain the complexity results in the later chapters. Classical graph algorithms like BFS and LexBFS are modified such that they can be carried out on the complement in time proportional to the size of the graph. LexBFS on the complement is used to test whether the complement of a graph is triangulated. If the complement is not triangulated, we show how to find the complement of a chordless cycle of length greater than three in time proportional to the size of the graph.

2.1 Basic terminology

An *undirected graph* $G = (V, E)$ consists of a set of *vertices* V and a collection of *edges* E, and each edge is an unordered pair of vertices. We represent a graph $G = (V, E)$ by drawing the vertices as points and by drawing a line between the points v and w if and only if the edge vw exists, see for instance Figure 2.1. Unless stated otherwise, we do not allow loops and parallel edges, thus no edge has the form vv and no two edges in E denote the same unordered pair.

If $G = (V, E)$ is a graph and vw an edge, then v is *incident* to vw and *adjacent* to w. In this case, we also say that *v sees w*. Similarly, we say that *v misses w* if v and w are two nonadjacent vertices. A *dominating* vertex is a vertex that sees every other vertex, and an *isolated* vertex misses all other vertices. A vertex is said to be *covered* by an edge set $F \subseteq E$ if it is incident to at least one edge in F, and the set $V(F)$ of all vertices covered by F is called the *cover of F*.

The *neighborhood* $N(v)$ of a vertex v is defined to be the set of vertices adjacent to v, and $\deg(v) = |N(v)|$ is the *degree* of v. The *closed neighborhood* $N[v] = N(v) \cup \{v\}$ is the neighborhood including the vertex v, and the *non-neighborhood* $\overline{N}(v) = V - N[v]$ is the set of vertices missed by v. It is also common to use the term "neighborhood" for more than one vertex: The *neighborhood* $N(A)$ of a subset A of V is the set of vertices not in A but adjacent to at least one vertex in A, i.e.

$$N(A) = \bigcup_{a \in A} N(a) - A.$$

The *complement* of a graph $G = (V, E)$ is the graph $\overline{G} = (V, \overline{E})$ that arises from G by replacing edges with nonedges and vice versa. Consequently, the neighborhood of a vertex v becomes the non-neighborhood of v in the complement and vice versa.

A graph $H = (W, F)$ is a *subgraph* of $G = (V, E)$ if $W \subseteq V$ and $F \subseteq E$. Given a subset W of V and a subset F of E, special subgraphs are

- the *subgraph spanned by F*, that is, the graph $H = (V(F), F)$ where $V(F)$ denotes the set of vertices incident to some edges in F, and
- the *subgraph induced by W*, that is, the graph $G_W = (W, E(W))$ where $E(W)$ denotes the set of edges with both endpoints in W.

If a graph H is an induced subgraph of G, it is customary to say that H is contained in G. Furthermore, in this thesis, the term subgraph is always used for induced subgraphs.

The *union* of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the graph $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. If V_1 and V_2 are disjoint, we call this the *disjoint union* $G_1 + G_2 = (V_1 + V_2, E_1 + E_2)$. The graph that results from inserting every edge between V_1 and V_2 into the disjoint union is called the *join* of G_1 and G_2, denoted by $G_1 \oplus G_2$.

A *complete* graph is a graph in which every vertex is adjacent to every other vertex, and a subset C of V that induces a complete graph is called a *clique*. A clique is *maximal* if it is not a proper subset of another clique, and a clique is *maximum* if no other clique contains more vertices. The size of a maximum clique of a graph G is called the *clique number* $\omega(G)$.

If the subgraph induced by a subset S of V has no edges, we say that S is *stable*. A stable set that cannot be enlarged is *maximal* and a largest stable set is *maximum*. The size of a maximum stable set of G is called the *stability number* $\alpha(G)$.

A *k-coloring* of $G = (V, E)$ is an assignment of k colors to the vertices in V such that two adjacent vertices receive different colors. In other words, a *k-coloring* is a partition of the vertices $V = V_1 + V_2 + \cdots + V_k$ such that V_i is a stable set for $i = 1, \ldots, k$. The smallest number k for which a k-coloring exists is the *chromatic number* of G, denoted by $\chi(G)$, and a k-coloring is *minimal* if $k = \chi(G)$.

A *bipartite* graph $G = (V, E)$ is a graph that admits a 2-coloring, that is, a bipartition $V = V_1 + V_2$ exists such that every edge has one endpoint in V_1 and the other in V_2. Similarly, a graph $G = (V, E)$ is *split* if a *split partition* $V_1 + V_2$ exists, that is, a bipartition $V = V_1 + V_2$ such that V_1 is a clique and V_2 is a stable set (in this order). In case of a split graph, we often write $G = (V_1, V_2, E)$ to indicate that $V_1 + V_2$ is a split partition.

A *path of length k* is a sequence of vertices $v_0, v_1, \ldots, v_{k-1}$ such that two consecutive vertices v_i and v_{i+1} are joined by an edge. A path is *simple* if every vertex in the sequence appears precisely once, and a path is *chordless* if it is simple and there are no other edges between the vertices in the path except for those between two consecutive vertices.

Similarly, a *cycle of length k* is a sequence of vertices $v_0, v_1, \ldots, v_{k-1}$ such that v_i and v_{i+1} are adjacent (indices modulo k). A cycle is *simple*

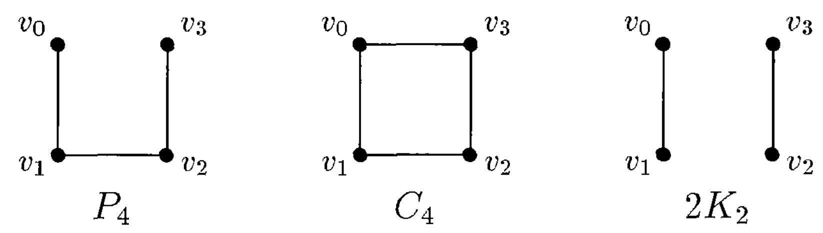


Figure 2.1: *Some special graphs with four vertices.*

if $k \geq 3$ and every vertex in the sequence appears precisely once, and a cycle is *chordless* if it is a simple cycle and there are no other edges between vertices in the cycle except for those between v_i and v_{i+1}. A chordless cycle of length $2k+1$ and $k > 1$ is also called an *odd hole* and its complement an *odd antihole.*

Some special graphs occur frequently in this work, so it is convenient to have names for some of them.

P_k: The chordless path graph on k vertices.
C_k: The chordless cycle graph on k vertices.
K_n: The complete graph on k vertices.
mK_n: The disjoint union of m copies of the K_n.

Furthermore, we often write $v_0v_1v_2\cdots v_{k-1}$ for a P_k that consists of a chordless path $v_0, v_1, \ldots, v_{k-1}$. The vertices v_0 and v_{k-1} are said to be the *endpoints* and the vertices $v_1, \ldots, v_{k-2}$ the *midpoints* of the P_k. If $v_0v_1v_2v_3$ is a P_4, then the edges v_0v_1 and v_2v_3 are called the *wings* and the edge v_1v_2 the *rib* of the P_4.

A graph is *connected* if a path exists between every pair of vertices, otherwise it is *disconnected.* The *connected components* of a graph are its maximal connected subgraphs. We usually do not distinguish between the vertices in a connected component and the connected component itself. If the complement $\overline{G}$ of a graph G is connected, we say that G is *coconnected.*

A connected graph without simple cycles is a *tree.* Given a graph $G = (V, E)$, a tree $T = (V, F)$ with $F \subseteq E$ is called a *spanning tree* of $G = (V, E)$. A spanning forest of graph $G = (V, E)$ is the disjoint union of spanning trees of the connected components of G (one per connected component).

A *directed graph* $\vec{G} = (V, \vec{E})$ consists of a set of *vertices* V and a set of directed edges $\vec{E}$ where a directed edge is an ordered pair of vertices. We write $v \rightarrow w$ for a directed edge with *starting point* v and *endpoint* w and, in the drawing of a directed graph, the edge $v \rightarrow w$ is represented by an arrow from v to w. Furthermore, we sometimes omit repeating a vertex when we want to express that certain directed edges exist. For instance, instead of $v_0 \rightarrow v_1, v_1 \leftarrow v_2, v_2 \rightarrow v_3$, we simply write $v_0 \rightarrow v_1 \leftarrow v_2 \rightarrow v_3$.

A *cycle* in a directed graph is a sequence of vertices $v_0, v_1, \ldots, v_{k-1}$ such that $v_0 \rightarrow v_1 \rightarrow \cdots \rightarrow v_{k-1} \rightarrow v_0$. A directed graph $\vec{G}$ is called *cyclic* if it contains such a cycle, otherwise it is called *acyclic*.

A *topological ordering* of a directed graph $\vec{G} = (V, \vec{E})$ is a linear order $v_1 < v_2 < \cdots < v_n$ of the vertices such that $v_i \rightarrow v_j$ in $\vec{E}$ implies $i < j$. It is easy to see that a topological ordering of a directed graph exists if and only if the graph is acyclic. Moreover, a topological ordering can be computed in linear time by *topological sorting*, see [27].

The directed graph $\vec{G} = (V, \vec{E})$ that arises from an undirected graph $G = (V, E)$ by assigning a direction to each edge in E is an *orientation* of $G = (V, E)$. Thus, an acyclic orientation is a directed acyclic graph; hence every acyclic orientation implies a topological ordering.

2.2 Perfect graphs

A graph G is *perfect* if $\omega(H) = \chi(H)$ holds for every induced subgraph H of G, and a graph G is *minimal imperfect* if $\omega(G) < \chi(G)$ and every proper induced subgraph is perfect. BERGE observed that odd holes and odd antiholes are minimal imperfect graphs. This observation led him to make the following conjecture.

Conjecture 2.2.1 (Strong Perfect Graph Conjecture) *A graph is perfect if it does not contain an odd hole or an odd antihole.*

Although the Strong Perfect Graph Conjecture is still open, partial results towards it have been obtained by LOVÁSZ and REED. In 1972, LOVÁSZ proved[1]

Theorem 2.2.2 (Perfect Graph Theorem) *The complement of a perfect graph is perfect.*

[1] An elegant proof can be found in [49]

A slightly stronger theorem was proved by REED in 1987. It asserts that the perfectness of a graph solely depends on the structure of the P_4s. In its original version [69], this theorem was expressed in terms of P_4-isomorphism: Two graphs G and H are *P_4-isomorphic* if they have the same vertex set and if every set of four vertices that induces a P_4 in G induces a P_4 in H.

Theorem 2.2.3 (Semi-Strong Perfect Graph Theorem) *If a graph G is P_4-isomorphic to a perfect graph, then G is perfect.*

Since the complement of a P_4 is again a P_4, the Semi-Strong Perfect Graph Theorem implies the Perfect Graph Theorem. Similarly, the validity of the Strong Perfect Graph Conjecture implies the Semi-Strong Perfect Graph Theorem [11].

The above theorems, however, have not led to a polynomial time algorithm to recognize perfect graphs. To date, it is not even known whether the recognition of perfect graphs is in NP or not. Moreover, it seems unlikely that the validity of the Strong Perfect Graph Conjecture would make the problem tractable: BIENSTOCK [8] has shown that it is NP-complete to test whether an arbitrary graph has an odd hole.

The situation does not look much better if we consider optimization problems on perfect graphs. In fact, most optimization problems remain NP-complete when restricted to perfect graphs. An exception is GRÖTSCHEL, LOVÁSZ AND SCHRIJVER's polynomial algorithm for computing a maximum clique and a minimum coloring in a perfect graph. Their algorithm, however, is based on the Ellipsoid method and therefore of mainly theoretical interest. For this reason, subclasses of perfect graphs with fast combinatorial optimization algorithms have been investigated. A famous example of such a graph class are perfectly orderable graphs.

2.2.1 Perfectly orderable graphs

To define perfectly orderable graphs, we first have to discuss the *greedy coloring algorithm.* This algorithm scans the vertices of a graph in a given linear order $v_1 < v_2 < \cdots < v_n$ and assigns to v_i the least color different from that of its already colored neighbors. A graph is *perfectly orderable* if it admits a linear order of its vertices such that, for every induced subgraph, the coloring computed by the greedy coloring algorithm using this order is minimal.

A linear order $v_1 < v_2 < \cdots < v_n$ is said to be *perfect* if no P_4 *abcd* satisfies $a < b$ and $c > d$. Similarly, an acyclic orientation is *perfect* if it contains no *obstruction*, that is, no P_4 *abcd* which is oriented $a \rightarrow b$ and $c \leftarrow d$. Therefore every topological ordering of a perfect orientation is a perfect order, and the orientation that arises from a perfect order by directing $v_i \rightarrow v_j$ if $v_i < v_j$ is perfect.

Since the greedy coloring algorithm computes a 3-coloring for a P_4 *abcd* if $a < b$ and $c > d$, every perfectly orderable graph admits a perfect order. CHVÁTAL showed that the converse holds as well.

Lemma 2.2.4 *Given a perfect order, the greedy coloring algorithm using the order computes a minimal coloring.*

Proof. Let $v_1 < v_2 < \cdots < v_n$ denote the perfect order and suppose that G is k-colored by the greedy algorithm. To prove our lemma, it suffices to show that a clique of size k exists. This is done by induction: We claim that every clique C of size $j < k$ with vertices of colors $k-j+1, k-j+2, \ldots, k$ can be enlarged with a vertex of color $k-j$.

Let $c_1 < c_2 < \ldots < c_j$ denote the vertices in such a clique C, and let W be the set of vertices w with color $k-j$ such that w sees a maximal number of consecutive vertices $c_i, c_{i+1}, \ldots c_j$ and $w < c_i$. Choose $w \in W$ minimal with respect to the perfect order. If w sees every vertex in C, then we are done. So let x be the largest vertex in C that misses w.

Since x is not colored $k-j$ by the greedy algorithm, there is a vertex u with color $k-j$ that sees x and satisfies $u < x$. Moreover, such a vertex u sees every vertex $y \in \{c_i, c_{i+1}, \ldots, c_j\}$, for otherwise $uxyw$ would be a P_4 with $u < x$ and $y > w$. Hence u belongs to W.

But every vertex in W either misses c_{i-1} or is greater than c_{i-1}. Clearly $u < x \leq c_{i-1}$, hence $x < c_{i-1}$. On the other hand, x is the largest vertex missed by w, thus w sees c_{i-1} and $w > c_{i-1}$. This implies $u < w$, a contradiction to our choice of w. □

In the above proof, we have also shown that a perfectly orderable graph has a maximum clique of size $\chi(G)$, thus perfectly orderable graphs are perfect[2]. Furthermore, many other problems that are NP-complete in general can be solved in polynomial time if a perfect orientation is given [36, 3]. To find a perfect orientation, however, is much

[2] A detailed implementation of a linear algorithm for computing $\chi(G)$ can be found in [15]

harder: MIDDENDORF AND PFEIFFER [56] proved that the recognition of perfectly orderable graphs is NP-complete. Therefore research focused on subclasses of perfectly orderable graphs that can be recognized in polynomial time.

One way to obtain candidates for such subclasses of perfectly orderable graphs is to restrict the number of ways a P_4 may be oriented. Classical examples of such graphs are triangulated graphs and comparability graphs: They admit an acyclic orientation such that no P_3 abc is oriented $a \rightarrow b \leftarrow c$ and $a \rightarrow b \rightarrow c$, respectively.

Another way to define subclasses of perfectly orderable graphs is by graph decompositions. OLARIU's stitch decomposition [60] and the modular decomposition are such examples. Conversely, graph decompositions can be used to recognize subclasses of perfectly orderable graphs. Triangulated graphs, for instance, are recognized by splitting off simplicial vertices.

2.2.2 Triangulated graphs

A graph G is *triangulated* if it does not contain an induced chordless cycle of length greater than three. A *simplicial* vertex is a vertex whose neighborhood induces a clique, and a *perfect elimination scheme* is an order of the vertices $v_1 < v_2 < \cdots < v_n$ such that the vertex v_i is simplicial in $G_{\{v_i, v_{i+1}, \ldots, v_n\}}$. It is well–know that a graph is triangulated if and only if it admits a perfect elimination scheme [27].

To see that the above definition matches the definition in the previous section, we first observe that a simplicial vertex cannot be the midpoint of a P_3. Therefore the reverse of a perfect elimination scheme induces an acyclic orientation such that no P_3 abc is oriented $a \rightarrow b \leftarrow c$. Conversely, if G admits such an orientation, then the smallest vertex in the implied order must be simplicial, thus a perfect elimination scheme can be constructed by repeatedly taking the smallest vertex.

The first algorithm for recognizing triangulated graphs in linear time is due to ROSE ET AL. [71]. In a first step, a linear algorithm called *lexicographic breath first search* is executed which provides a *LexBFS-ordering* of the vertices. ROSE ET AL. showed that every LexBFS-ordering of a triangulated graph is also a perfect elimination scheme. To recognize triangulated graphs, it therefore suffices to test whether a given vertex order is a perfect elimination scheme. Both algorithms are given in Section 2.3.

If a graph is not triangulated, a LexBFS-ordering can be used to find a chordless cycle of length greater than three in linear time. Let v_i denote the largest vertex which is not simplicial in $G' = G_{\{v_i, v_{i+1}, \dots, v_n\}}$ (It is explained in Section 2.3 how to find such a vertex in linear time). Therefore nonadjacent vertices x_1 and x_2 in $N(v_i) \cap \{v_{i+1}, \dots, v_n\}$ exist. Following the proof of Theorem 4.3 [27], we choose x_1 and x_2 such that x_2 is as large as possible. Then there is a chordless cycle $x_1, v_i, x_2, \dots$ in G' and this cycle can be found in linear time by computing a shortest path between x_1 and x_2 in $G'_{\{x_1,x_2\} \cup \overline{N}(v_i)}$. We formulate this result as a theorem.

Theorem 2.2.5 *If a graph is not triangulated, a chordless cycle of length greater than three can be found in linear time.*

2.3 Graph algorithms on the complement

The purpose of this section is to provide basic linear time algorithms for the complement of a graph. For instance, we present a linear algorithm for computing a LexBFS-ordering of the complement and a linear algorithm for recognizing cotriangulated graphs. Those algorithms are then used in Chapter 7 to recognize cobithreshold graphs in linear time.

As usual, it is assumed that the input graph $G = (V, E)$ is given by its adjacency lists, i.e., the vertices of the graph are stored in an array and the neighborhood $N(v)$ of a vertex v is a doubly linked list attached to the array element that contains v. Therefore the removal of a vertex can be carried out in constant time.

For every problem, we first present the classical linear algorithm. We then discuss the changes that must be made to achieve a running time of $O(|V| + |\overline{E}|)$ if the input is the complement $\overline{G} = (V, \overline{E})$. The first problem considered is graph search.

2.3.1 Breath first search

A graph search algorithm takes a graph $G = (V, E)$ and a vertex $v_0 \in V$ and computes the vertices "reachable" from v_0, that is, the set of vertices in the connected component of v_0. The graph search algorithm known as *breath first search*, BFS for short, works as follows.

breath first search

input: a graph $G = (V, E)$ and a vertex $v_0 \in V$
output: in B the vertices in the connected component of v_0

(1) initialize list W with V;
(2) let Q and B be empty lists;
(3) remove v_0 from W and append it to Q;
(4) **while** Q is not empty **do**
(5) let v be the first vertex in Q;
(6) remove v from Q and append it to B;
(7) remove $W \cap N(v)$ from W and append it to Q
(8) **od**

Algorithm 2.1

It is easy to see that every vertex belongs to precisely one of the lists W, Q or B: List W contains the not-reached vertices, list Q the reached but not visited vertices, and B the reached and visited vertices. The list Q serves as "queue data structure".

Line (7) of Algorithm 2.1 can be implemented to run in time proportional to $|N(v)|$: In a first step, the list W is divided into two lists $W_1 = W \cap N(v)$ and the "remainder" $W_2 = W - N(v)$. In a second step, W_2 becomes the new list W and the vertices in W_1 are appended to Q. Therefore Line (7) of Algorithm 2.1 can be replaced by Line (7.1) and Line (7.2) below.

(7.1) split W into $W_1 = W \cap N(v)$ and $W_2 = W - N(v)$;
(7.2) let $W = W_2$ and append the vertices in W_1 to Q;

Except for the initialization in Line (1), the running time of Algorithm 2.1 is proportional to the number of vertices and edges in the connected component of v_0 provided that each vertex stores the information to which list it belongs.

The order of the vertices as they are visited by BFS is called a *BFS-ordering*. Therefore, in the above algorithm, the sequence of the vertices as they appear in B is a BFS-ordering of the vertices in the connected component of v_0.

Algorithm 2.2 is a straight-forward generalization of Algorithm 2.1 to visit every vertex in the graph. As before, Line (5) to Line (10) compute the connected component of the vertex v_0 chosen in Line (4), thus Algorithm 2.2 implicitly computes the connected components of G.

connected components

input: a graph $G = (V, E)$
output: a BFS-ordering B

(1) initialize W with V;
(2) let Q and B be empty lists;
(3) **while** W is not empty **do**
(4) remove an arbitrary vertex v_0 from W and append it to Q;
(5) **while** Q is not empty **do**
(6) let v be the first vertex in Q;
(7) remove v from Q and append it to B;
(8) split W into $W_1 = W \cap N(v)$ and $W_2 = W - N(v)$;
(9) let $W = W_2$ and append the vertices in W_2 to Q
(10) **od**
(11) **od**

Algorithm 2.2

Now assume the input of the above algorithm is the complement graph $\overline{G} = (V, \overline{E})$. Then the adjacency list of a vertex v consists of the non-neighborhood $\overline{N}(v)$. We only have to consider Line (8) as the adjacency lists appear in no other line. But $W \cap N(v) = W - \overline{N}(v)$ and $L - N(v) = L \cap \overline{N}(v)$, so Line (8) can be replaced with

(8.1) split W into $W_1 = W - \overline{N}(v)$ and $W_2 = W \cap \overline{N}(v)$;

Since the execution of Line (8.1) takes time proportional to $|\overline{N}(v)|$, we have derived an algorithm that runs in $O(|V| + |\overline{E}|)$. In other words, we have

Lemma 2.3.1 *Given a graph $G = (V, E)$, a BFS-ordering and the connected components of its complement $\overline{G} = (V, \overline{E})$ can be computed in $O(|V| + |E|)$.*

Remark 1: A similar approach for depth first search can be found in the SODA'97 paper by DAHLHAUS ET AL. [21]. Remark 2 and 3 have to be seen in connection with their work.

Remark 2: Suppose the input graph is given in a "mixed representation", that is, the adjacency list of a vertex v contains either the vertices in $N(v)$ or the vertices in $\overline{N}(v)$. Depending on which case applies, we can either execute Line (8) or Line (8.1) in Algorithm 2.2. The result is an algorithm that is linear in the size of the input of the mixed representation.

Remark 3: It is sometimes useful to have a so-called BFS-forest: An edge $x \to y$ belongs to the BFS-forest if and only if the vertex y was appended to Q while visiting x. One way to compute those edges is to implement Q as a list of lists as follows.

In Line (9), list W_2 is appended (as a list) to Q, and in Line (6), a vertex v from the first list in Q is chosen. Furthermore, we store v in the head of W_2 before appending W_2 to Q. When removing a vertex v from the first list L_0 in Q, we insert the edge $w \to v$ in our BFS-forest where w stands for the vertex in the head of L_0.

2.3.2 Lexicographic breath first search

In connection with the recognition of triangulated graphs, we are interested in a lexicographical breath first search ordering, *LexBFS-ordering* for short. A LexBFS-ordering is a special BFS-ordering computed by a refined BFS algorithm.

As mentioned in Remark 3 of the previous section, the data structure Q in Algorithm 2.2 can be implemented as a list of lists. In ordinary BFS, it does not matter in which order the vertices in the same list in Q are visited. In LexBFS, however, vertices adjacent to the first already visited vertices are preferred. In fact, every time a vertex v is visited, a list L in Q is replaced with two lists $L_1 = L \cap N(v)$ immediately followed by the remainder $L_2 = L - N(v)$. Since the position of L_1 and L_2 relative to the other lists in Q remains the same, a LexBFS-ordering is a special BFS-ordering.

In the implementation of LexBFS given as Algorithm 2.3, we have assumed that every list in Q is not empty, thus no empty lists are inserted in Q and, whenever a list becomes empty, it is immediately removed from Q. With this assumption, Line (8) to Line (11) can be executed in time proportional to $|N(v)|$ as follows.

The vertices w in $N(v)$ are scanned and, if w belongs to a list L in Q but the list in front of L is not empty, an empty list L_1 is inserted immediately before $L_2 = L$. In a second scan of the vertices w in $N(v)$, every vertex w in a list L_2 in Q is moved to the list L_1 immediately

LexBFS

input: a connected graph $G = (V, E)$ and a vertex $v_0 \in V$
output: a LexBFS-ordering B

```
(1)  initialize list W with V;
(2)  let B be an empty list;
(3)  let Q be an empty list of lists;
(4)  remove v_0 from W and append a list consisting of v_0 to Q;
(5)  while Q is not empty do
(6)    let v be a vertex in the first list L_0 of Q;
(7)    remove v from L_0 and append it to B;
(8)    forall lists L in Q do
(9)      split L into lists L_1 = L ∩ N(v) and L_2 = L − N(v);
(10)     replace L in Q with L_1 followed by L_2;
(11)   od
(12)   split W into lists W_1 = W ∩ N(v) and W_2 = W − N(v);
(13)   let W = W_2 and append W_1 to Q
(14) od
```

Algorithm 2.3

before L_2 (and L_2 is removed from Q if it becomes empty). Thus the overall running time of Algorithm 2.3 is linear.

Now assume that the input of the above algorithm is the complement graph $\overline{G} = (V, \overline{E})$. Again changes affect only Line (9) and Line (12) because adjacency lists appear in no other line. As in Algorithm 2.2, we can replace Line (9) with

```
(9.1)      split L into lists L_1 = L − N̄(v) and L_2 = L ∩ N̄(v);
```

and Line (12) with

```
(12.1)   split W into lists W_1 = W − N̄(v) and W_2 = W ∩ N̄(v);
```

With the technique described above, Line (8) to Line (11) can be executed in time proportional to $|\overline{N}(v)|$, and the overall running time of Algorithm 2.2 is proportional to $|V| + |\overline{E}|$. Thus

Theorem 2.3.2 *Given a graph $G = (V, E)$, a LexBFS-ordering of $\overline{G} = (V, \overline{E})$ can be computed in $O(|V| + |E|)$.*

2.3.3 Testing a perfect elimination scheme

In this section, we address the problem of testing whether a given vertex order $v_1 < v_2 < \cdots < v_n$ is a perfect elimination scheme. To simplify our notation, let $V_i = \{v_i, v_{i+1}, \ldots, v_n\}$, let $N_i = N(v_i) \cap V_{i+1}$ and let $\min(N_i)$ denote the least vertex in N_i (if N_i is not empty). Thus, $v_1 < v_2 < \cdots < v_n$ is a perfect elimination scheme if

$$\forall i : N_i \text{ is a clique.} \tag{2.1}$$

We claim that this is equivalent to

$$\forall i : \exists v_j = \min(N_i) : N_i - v_j \subseteq N_j, \tag{2.2}$$

which reads *for all i for which the vertex $v_j = \min(N_i)$ exists, the property $N_i - v_j \subseteq N_j$ holds.* The proof of this claim is by induction: Suppose that (2.1) and (2.2) hold for $i = 2, \ldots n$. If N_1 is empty, then there is nothing to prove. Otherwise the vertex $v_j = \min(N_1)$ exists, hence our induction hypothesis asserts that N_j is a clique. Therefore $N_1 - v_j \subseteq N_j$ if and only if N_1 is a clique.

To verify (2.2) efficiently, we scan the vertices v_i in ascending order and collect the vertices $N_i - v_j$ in A_j where v_j denotes the smallest vertex in N_i (if such a vertex exists), thus

$$A_j = \bigcup_{\forall i : \exists v_j = \min(N_i)} N_i - v_j. \tag{2.3}$$

At the time when v_j is reached, the computation of A_j is complete and the test $A_j \subseteq N_j$ can be performed.

In the implementation given as Algorithm 2.4, the vertices in $N_i - v_j$ are simply appended to the list A_j, so A_j can contain the same vertex multiple times. The test whether $A_j \subseteq N_j$ is done in time proportional to the sum of the length of list A_j and N_j by using an array as described in [27]. Consequently, the running time of Algorithm 2.4 is $O(|V| + |E|)$.

If $v_1 < v_2 < \cdots < v_n$ is not a perfect elimination scheme, we are interested in the largest vertex v_i that is not simplicial in $G_{\{v_i, v_{i+1}, \ldots, v_n\}}$. To find this vertex in linear time, we store with each vertex w inserted in list A_j the vertex $w' = v_i$ responsible for the insertion of w. Then w' is nonsimplicial for every $w \in A_i - N_i$ in Line (9), thus we just have to find the largest vertex among those vertices w'.

is_perfect

input: a graph $G = (V, E)$ and a vertex order $v_1 < v_2 < \cdots < v_n$
output: true if $v_1 < \cdots < v_n$ is a perfect elimination scheme of G

```
(1)  for j = 1 to n do
(2)    let A_j be an empty list
(3)  od;
(4)  for i = 1 to n do
(5)    if N_i ≠ ∅ then
(6)       let v_j = min(N_i);
(7)       append N_i − v_j to A_j
(8)    fi
(9)    if A_i ⊈ N_i then
(10)      return "false"
(11)   fi
(12) od;
(13) return "true"
```

Algorithm 2.4

Theorem 2.3.3 *Let $G = (V, E)$ be a graph and $v_1 < v_2 < \cdots < v_n$ a linear order of its vertices. If this order is no perfect elimination scheme, then the largest vertex v_i not simplicial in $G_{\{v_i, v_{i+1}, \cdots, v_n\}}$ can be found in linear time.*

Now assume that the input is the complement graph $\overline{G} = (V, \overline{E})$. Since $N_i \subseteq V_{i+1}$ and $A_i \subseteq V_{i+1}$, it is quite natural to work with the complement of those sets in V_{i+1}. So let $\overline{N}_i = V_{i+1} - N_i$ and let $\overline{A}_j = V_{j+1} - A_j$. Therefore the test $A_i \not\subseteq N_i$ in Line (9) translates into $\overline{N}_i \not\subseteq \overline{A}_i$. Furthermore, according to (2.3), we have

$$\overline{A}_j = V_{j+1} \cap \bigcap_{\forall i : \exists v_j = \min(N_i)} \overline{N_i - v_j} = \bigcap_{\forall i : \exists v_j = \min(N_i)} \overline{N}_i \cap V_{j+1}.$$

Note that $\overline{A}_j = V_{j+1}$ if no index i exists for which $v_j = \min(N_i)$, that is, N_i is empty. Therefore every $\overline{A}_j$ has to be initialized with V_{j+1}, which results in an $O(|V|^2)$ running time.

To obtain a linear running time, we maintain lists C_j consisting of those vertices v_i for which $v_j = \min(N_i)$. Then

$$v \in \overline{A}_j \iff \forall v_i \in C_j : v \in \overline{N}_i \cap V_{j+1}.$$

So if B_j stands for the concatenation of the lists that represent the sets $\overline{N}_i \cap V_{j+1}$, $i \in C_j$, then the vertices in $\overline{A}_j$ are precisely those vertices which appear $|C_j|$ times in B_j.

In Algorithm 2.5, the lists B_j are computed from the empty lists by appending $\overline{N}_i \cap V_{j+1}$ whenever $N_i \neq \emptyset$. In Line (11), we verify that every vertex in $\overline{N}_i$ appears $|C_i|$ times in B_i. We write $\overline{N}_i \subseteq_{\times |C_i|} B_i$ if this is true and $\overline{N}_i \not\subseteq_{\times |C_i|} B_i$ otherwise. Therefore Algorithm 2.5 is correct.

is_complement_perfect

input: a graph $\overline{G} = (V, \overline{E})$ and a vertex order $v_1 < v_2 < \cdots < v_n$
output: true if $v_1 < \cdots < v_n$ is a perfect elimination scheme of G

(1) **for** $j = 1$ **to** n **do**
(2)   let B_j be an empty list;
(3)   let C_j be an empty list
(4) **od**;
(5) **for** $i = 1$ **to** n **do**
(6)   **if** $N_i \neq \emptyset$ **then**
(7)   let $v_j = \min(N_i)$;
(8)   append $\overline{N}_i \cap V_{j+1}$ to B_j
(9)   append v_i to C_j
(10)   **fi**;
(11)   **if** $|C_i| > 0$ and $\overline{N}_i \not\subseteq_{\times |C_i|} B_i$ **then**
(12)   **return** "false"
(13)   **fi**
(14) **od**;
(15) **return** "true"

Algorithm 2.5

The test $N_i \neq \emptyset$ in Line (6) can be implemented as $|\overline{N}_i| \neq |V_{i+1}|$. Furthermore, we may assume that the adjacency lists of $\overline{G}$ are sorted according to $v_1 < v_2 < \cdots < v_n$ (sorting the adjacency lists of a graph is linear, see [27]). Thus Line (7) can be executed in $O(|\overline{N}_i|)$ as v_j is the smallest vertex in V_{i+1} not contained in $\overline{N}_i$. By using an array, the running time of Line (11) is proportional to the length of the lists B_i and $\overline{N}_i$; hence Algorithm 2.5 is in $O(|V| + |\overline{E}|)$.

Theorem 2.3.4 *For a graph $G = (V, E)$, the test whether a linear order $v_1 < v_2 < \cdots < v_n$ is a perfect elimination scheme of $\overline{G}$ can be performed in $O(|V| + |E|)$.*

We conclude this section with the problem of finding the largest vertex v_i that is not simplicial in $G_{\{v_i, v_{i+1}, \ldots, v_n\}}$. To begin with, each vertex w inserted into the list B_j has to store the vertex $w' = v_i$ responsible for the insertion of w. If $|C_i| > 0$ in Line (11), we perform the test $\overline{N}_i \not\subseteq_{\times |C_i|} B_i$ by computing $\overline{N}_i - \overline{A}_i$ with an array of initially empty lists $T(v)$, $v \in V$.

(11.1) **forall** w in list B_i **do**
(11.2)   append w' to $T(w)$
(11.3) **od**
(11.4) **forall** $w \in \overline{N}_i$ **do**
(11.5)   **if** $T(w) \not\subseteq C_i$ **then**
(11.6)   (* the vertices in $T(w) - C_i$ are nonsimplicial *)
(11.7)   **return** "false"
(11.8)   **fi**
(11.9) **od**
(11.10) **forall** w in list B_i **do**
(11.11)   let $T(w)$ be an empty list
(11.12) **od**

It is assumed that the forall-statement scans the vertices in the sequence as they appear in the given list. Therefore $T(w)$ is sorted according to $v_1 < v_2 < \cdots < v_n$. Since C_i is sorted in the same way, Line (11.5) can be carried out in $O(|T(w)|)$; hence the running time of the above code is $O(|B_i| + |\overline{N}_i|)$.

Note that every vertex $x \in T(w) - C_i$ is nonsimplicial because $x < v_i < w$ and x sees v_i and w but v_i and w are nonadjacent. Moreover, if we scan $T(w)$ in reverse order, the first vertex in $T(w)$ but not in C_i is the largest nonsimplicial vertex in $C_i - T(w)$. Clearly, the largest vertex v_i that is nonsimplicial in $\overline{G}_{\{v_i, v_{i+1}, \ldots, v_n\}}$ is found this way.

Theorem 2.3.5 *Let $G = (V, E)$ be a graph and $v_1 < v_2 < \cdots < v_n$ a linear order of its vertices. If this order is no perfect elimination scheme of $\overline{G}$, the largest vertex v_i that is not simplicial in $\overline{G}_{\{v_i, v_{i+1}, \ldots, v_n\}}$ can be found in $O(|V| + |E|)$.*

Given the vertex v_i of the above theorem, we can calculate a chordless cycle in $\overline{G}$ of length greater than three in $O(|V| + |E|)$ with the method described in Section 2.2.2. Together with Theorem 2.3.2, we have the following.

Corollary 2.3.6 *Let $G = (V, E)$ be a graph whose complement is not triangulated. Then the complement of a chordless cycle of length greater that three can be found in $O(|V| + |E|)$.*

Chapter 3

Comparability graphs

In this chapter, we present historical results in connection with comparability graphs. On the one hand, most of these results are needed in the subsequent chapters. On the other hand, their presentation allows us to demonstrate the methods and proof techniques used in the rest of this thesis. We shall therefore often refer to the theorems and proofs of this chapter to point out the analogy.

In the first section, we introduce P_4-free graphs and show that they are precisely those graphs for which every subgraph is either disconnected or codisconnected. The arising decomposition is then generalized to what is nowadays known as the modular decomposition. The uniqueness of the decomposition comes from the fact that the union of two intersecting modules that are not contained in one another induces a disconnected or codisconnected graph.

The modular decomposition of a graph is closely related to its P_3-structure. In the third section, we therefore analyze this structure and use the obtained results to compute the modular decomposition and to develop algorithms for recognizing comparability graphs.

Finally, in the last section, we review Hoàng and Reed's result on induced subgraphs which exist in prime graphs that are not triangulated. We show that those subgraphs can be found in linear time by applying the theorems of Section 2.3.

3.1 Cographs

A graph is called *cograph* if it does not contain a P_4. Clearly, a P_4-free graph cannot contain an obstruction, hence cographs (and their complements) are perfectly orderable. The following lemma was found by SEINSCHE [73] in 1974.

Lemma 3.1.1 (Seinsche) *A nontrivial, connected and coconnected graph contains a* P_4.

Proof. Let G be a smallest counterexample, i.e. G is nontrivial, P_4-free, connected and coconnected but every nontrivial induced subgraph is disconnected or codisconnected. Let v be an arbitrary vertex of G and suppose that G_{V-v} is disconnected.

Since G is connected, every connected component of G_{V-v} contains a vertex that sees v. But v is not isolated in $\overline{G}$; hence a connected component G_1 of G_{V-v} exists with a vertex that misses v. Following a path in G_1 from this vertex to a vertex that sees v, we encounter an edge ab with $av \notin E$ and $bv \in E$. Thus $abvx$ is a P_4 for any vertex x adjacent to v in a connected component of G_{V-v} different from G_1, a contradiction to our assumption.

If G_{V-v} is codisconnected, the above argumentation applied to the complement leads to a P_4 in $\overline{G}$, again a contradiction to our assumption because the complement of a P_4 is again a P_4. □

Since a P_4 is connected and coconnected, cographs are precisely those graphs which are completely decomposed by the following algorithm.

---

if G is trivial **then**
  stop
if G is disconnected **then**
  decompose the connected components of G
if $\overline{G}$ is disconnected **then**
  decompose the connected components of $\overline{G}$

An arbitrary nontrivial disconnected graph is coconnected, so

Fact 3.1.2 *Every graph is connected or coconnected.*

Consequently no graph is disconnected and codisconnected at the same time, which proves the uniqueness of the above decomposition.

Furthermore, the decomposition can be represented by a tree in which the decomposition operations are distinguished by 0 and 1-nodes and, if the graph is trivial, by an empty node labeled v where v stands for the only vertex in G. The computation of this unique decomposition tree called *cotree* is given below.

buildCotree(G)

input: a graph $G = (V, E)$
output: the root of the cotree of G

(1) **if** $|V| = 1$ **then**
(2) let v be the vertex in V;
(3) return an empty node labeled v;
(4) **else if** G is disconnected **then**
(5) let $G_1, \ldots, G_t$ be the connected components of G;
(6) let $r_i =$ buildCoTree(G_i) for $i = 1, \ldots, t$;
(7) return a 0-node with children $r_1, r_2, \ldots, r_t$
(8) **else if** $\overline{G}$ is disconnected **then**
(9) let $\overline{G}_1, \overline{G}_2, \ldots, \overline{G}_t$ be connected components of $\overline{G}$;
(10) let $r_i =$ buildCoTree(G_i) for $i = 1, \ldots, t$;
(11) return a 1-node with children $r_1, r_2, \ldots, r_t$
(12) **else**
(13) **stop** (* G is no cograph *)
(14) **fi**

Algorithm 3.1

If we return a 2-node instead of stopping at Line (13), the above algorithm computes a decomposition tree for an arbitrary graph. The original graph can then be reconstructed from the decomposition tree if the graph G in Line (13) is stored in the corresponding 2-node.

In the next Section, we discuss a generalization of the above decomposition, the so-called modular decomposition. Since the modular decomposition tree can be found in linear time, the same holds for the above decomposition tree, thus cographs can be recognized in linear time.

3.2 The modular decomposition

The modular decomposition was found by GALLAI [23] in 1967 while investigating comparability graphs. To discuss the modular decomposition, we need the following definitions.

Given a graph $G = (V, E)$ and a subset A of V, a vertex $v \notin A$ is called *A-null* if v misses every vertex in A. Similarly, $v \notin A$ is *A-universal* if it sees every vertex in A. Vertices not in A that are neither A-universal nor A-null are called *A-partial.*

A *module* is a nonempty vertex set H such that no H-partial vertex exists. A module H with $1 < |H| < |V|$ is a *homogeneous set.* The following properties of modules are important to prove the results of this section.

Fact 3.2.1 *If modules H_1 and H_2 intersect, then $H_1 \cup H_2$ is again a module.*

Fact 3.2.2 *If intersecting modules H_1 and H_2 do not contain each other, then $G_{H_1 \cup H_2}$ is either disconnected or codisconnected.*

Proof. The first fact is obvious. To prove the second, let H_1 and H_2 be two intersecting modules such that none is a subset of the other. If G_{H_1} is connected, then an edge between a vertex in $H_1 \cap H_2$ and a vertex $v_1 \in H_1 - H_2$ exists. But H_2 is a module, so v_1 sees every vertex in H_2. Moreover, since every vertex in $H_2 - H_1$ sees v_1 and H_1 is a module, every vertex in $H_2 - H_1$ sees every vertex in H_1. Hence $\overline{G}_{H_1 \cup H_2} = \overline{G}_{H_1} + \overline{G}_{H_2 - H_1}$, thus $\overline{G}$ is disconnected.

If G_{H_1} is disconnected, then H_1 is coconnected and the above argumentation applies to the complement, thus $G = G_{H_1} + G_{H_2 - H_1}$ and G is disconnected. □

A homogeneous set H is *connected* if G_H is connected, and it is *coconnected* if G_H is coconnected. Furthermore a homogeneous set H is called *maximal* if no other homogeneous set is a superset of H.

The modular decomposition is based on the following theorem.

Theorem 3.2.3 *The maximal homogeneous sets of a connected and coconnected graph are disjoint.*

Proof. Let $G = (V, E)$ be connected and coconnected and suppose that different maximal homogeneous sets H_1 and H_2 intersect. Then $H_1 \cup H_2 = V$ because $H_1 \cup H_2$ is a module.

Furthermore, H_1 and H_2 are homogeneous sets, so H_1 and H_2 are proper subsets of V, thus $H_1 \not\subseteq H_2$ and $H_2 \not\subseteq H_1$. By Fact 3.2.2, $G = G_{H_1 \cup H_2}$ is disconnected or codisconnected, a contradiction to our assumption. □

The *modular decomposition* is given in Algorithm 3.2. It combines the decomposition into connected components of G and $\overline{G}$ with the decomposition into maximal homogeneous sets, thus the uniqueness of the modular decomposition tree follows immediately from Fact 3.1.2 and Theorem 3.2.3.

buildModTree(G)

input: a graph $G = (V, E)$
output: the root of the modular decomposition tree of G

```
if |V| = 1 then
   let v be the vertex in V;
   return an empty node labeled v;
else if G is disconnected then
   let G_1, G_2, ..., G_t be the connected components of G;
   let r_i = buildModTree( G_i ) for i = 1, ..., t;
   return a 0-node with children r_1, r_2, ..., r_t
else if co-G is disconnected then
   let co-G_1, co-G_2, ..., co-G_t be the connected components of co-G;
   let r_i = buildModTree( G_i ) for i = 1, ..., t;
   return a 1-node with children r_1, r_2, ..., r_t
else (* G and co-G are connected and |V| > 1 *)
   let H_1, H_2, ..., H_t be the maximal homogeneous sets of G;
   let r_i = buildModTree( G_{H_i} ) for i = 1, ..., t;
   return a 2-node with children r_1, r_2, ..., r_t
fi
```

Algorithm 3.2

A nontrivial graph that cannot be decomposed by the above algorithm is called *prime*, thus a nontrivial graph is prime if it is connected and coconnected and if it has no homogeneous sets.

If h is a vertex in a homogeneous set H, we say that G_{V-H+h} is derived from G by substituting the marker vertex h for the homogeneous set H. The prime graph that arises from substituting marker vertices for all maximal homogeneous sets of G is called the *characteristic graph* of G. To reconstruct G from its modular decomposition tree, it suffices to store the characteristic graphs in the 2-nodes of the tree.

If a graph has a nontrivial modular decomposition tree, this tree can be used to apply divide and conquer methods to solve optimization problems like maximum clique, see [59, 58] for details. Thus the question arises how fast the modular decomposition of a graph can be computed. A simple $O(|V|^3)$ algorithm is described in Section 3.3. In recent years, however, linear time algorithms for the modular decomposition have been found [54, 21]. Unfortunately, those algorithms are rather complicated.

3.3 Comparability graphs

A graph is a *comparability graph* if it admits a *transitive orientation*, i.e., an acyclic orientation such that no P_3 abc is directed $a \to b \to c$. Since a transitive orientation cannot contain obstructions, an orientation that is transitive is also perfect. Furthermore, the orientation of one edge in a P_3 in a transitive orientation determines the orientation of the other edge in the same P_3. This observation gives rise to the following definition.

Definition 3.3.1 *Two edges are* P_3-adjacent *if they belong to the same* P_3*, and a* P_3-class *is an equivalence class of the transitive closure of the* P_3*-adjacency relation.*

Obviously, the orientation of one edge in a P_3-class forces the orientation of all other edges in the same P_3-class. Therefore every P_3-class of a comparability graph can be transitively oriented in precisely two ways. GHOUILA-HOURI [24] showed that the converse holds as well.

Theorem 3.3.2 (GHOUILA-HOURI) *A graph is a comparability graph if and only if each of its* P_3*-classes admits a transitive orientation.*

We prove of the above theorem in the same way we shall prove our results on P_4-comparability graphs in Chapter 5. First, we study the P_3-classes of arbitrary graphs.

3.3.1 P_3-classes

In the rest of this section, C^* stands for a P_3-class and $C^*(vw)$ for the P_3-class that contains the edge vw. Given a set H of vertices, a P_k is said to be *H-partial* if it is not contained in G_H but has at least one edge in $E(H)$.

Theorem 3.3.3 *Let C^* denote an arbitrary P_3-class. Then no $V(C^*)$-partial P_3 exists.*

Proof. Let abc be a $V(C^*)$-partial P_3. Without loss of generality, we may assume that $a \in V - V(C^*)$ and $b, c \in V(C^*)$. Since c is covered by C^*, an edge $cd \in C^*$ exists. Clearly $b \neq d$, b sees d and a misses d, for otherwise the contradiction $C^* = C^*(ab)$ would arise. We claim that, for every edge $xy \in C^*$, b sees x and y and a misses x and y. It follows that b cannot be covered by C^*, a contradiction.

The proof of our claim is by induction. Since it holds for cd, the basis is settled. The inductive step consists of showing our claim for an edge yz in a P_3 xyz on the assumption that it holds for xy. If b misses z, then aby and byz are P_3s, hence $C^* = C^*(ab)$, a contradiction. If a sees z, then yza is a P_3, hence $az \in C^*$, another contradiction. Thus b misses y and z and a sees y and z as claimed. □

Suppose that a $V(C^*)$-partial vertex v exists. Since $G_{V(C^*)}$ is connected, there is a path in $G_{V(C^*)}$ from a vertex that misses v to a vertex that sees v. Following this path, we must encounter an edge ab with $av \notin E$ and $bv \in E$. But abv is a $V(C^*)$-partial P_3, a contradiction to Theorem 3.3.3. Therefore no $V(C^*)$-partial vertex exists, thus

Corollary 3.3.4 *The cover of a P_3-class is a module.*

Conversely, assume that an edge xy has both endpoints in a module H. If $H \subset V(C^*(xy))$, then a P_3 abc in C^* with $a, b \in H$ and $c \in V - H$ exists. But this is impossible because c is H-partial, hence

Corollary 3.3.5 *If both endpoints of an edge xy belong to a module H, then $V(C^*(xy)) \subseteq H$.*

The above corollary applied to G and $\overline{G}$ implies that every minimal homogeneous set is the cover of a P_3-class of G or $\overline{G}$. By Theorem 3.2.3, the maximal homogeneous sets can therefore be computed "bottom up" from the covers of the P_3-classes of G or $\overline{G}$.

The following theorem states that P_3-classes can be uniquely identified by their covers.

Theorem 3.3.6 *Two different P_3-classes have different covers.*

We prepare the proof of this theorem with the following lemma.

Lemma 3.3.7 (Triangle Lemma) *Let $\{a, b, c\}$ be a clique such that $C^*(ab)$ and $C^*(ac)$ are different from $C^*(bc)$. Then a is not in the cover of $C^*(bc)$.*

Proof. We prove the lemma by showing that, for every edge $xy \in C^*(bc)$, the edges ax and ay exist but do not belong to $C^*(bc)$. Clearly this holds for $xy = bc$.

For the inductive step, we have to prove our claim for an edge yz in a P_3 xyz on the assumption that it already holds for xy. If $az \notin E$, then the P_3 ayz implies $ay \in C^*(bc)$, a contradiction to our assumption. Therefore $az \in E$ and xaz is a P_3, thus $C^*(az) = C^*(xa) \neq C^*(bc)$ as claimed. □

Proof of Theorem 3.3.6. Suppose that two different P_3-classes C_1^* and C_2^* have the same cover and let b denote an arbitrary vertex in $V(C_1^*) = V(C_2^*)$. Then edges ab in C_1^* and bc in C_2^* exist. Furthermore, a sees c and and either $ac \notin C_1^*$ or $ac \notin C_2^*$.

Without loss of generality, let $ac \notin C_2^*$. Then $C_2^* = C^*(bc)$ is different from $C_1^* = C^*(ab)$ and $C^*(ac)$, thus Lemma 3.3.7 implies that $a \notin V(C_2^*)$, a contradiction to our assumption. □

The next theorem constitutes the main part of GALLAI's decomposition theorem. Together with Theorem 3.2.3, it is considered as one of the deepest results in connection with comparability graphs [45].

Theorem 3.3.8 (GALLAI) *Let $G = (V, E)$ be a nontrivial connected and coconnected graph and let $H_1, \dots, H_k$ be the maximal homogeneous sets of G. Then $E - E(H_1) - \cdots - E(H_k)$ is a P_3-class that covers G.*

Proof. Since G is connected and, by Theorem 3.2.3, the maximal homogeneous sets of G are disjoint, there is an edge vw in $E - E(H_1) - \cdots - E(H_k)$. Furthermore $C^* = C^*(vw)$ covers G, as otherwise $V(C^*)$ would be homogeneous and therefore be contained in a maximal homogeneous set H_i, a contradiction. By Theorem 3.3.6, there is only one such P_3-class, hence $E - E(H_1) - \cdots - E(H_k)$ is a subset of C^*. But no other edge belongs to C^* because of Corollary 3.3.5. □

The above theorem leads to a very simple $O(|V|^3)$ time algorithm for the modular decomposition of a graph. In a first step, we compute the P_3-classes $C_1^*, \ldots, C_k^*$ of G as well as their covers.

The P_3-classes are precisely the vertices in the connected components of $\tilde{G} = (\tilde{V}, \tilde{E})$ where $\tilde{V} = E$ and two vertices are adjacent in $\tilde{G}$ if the corresponding edges belong to the same P_3 in G. Since the connected components of $\tilde{G}$ can be found in $O(|\tilde{V}| + |\tilde{E}|)$ and every edge in G can be in at most $|V| - 2$ different P_3s, we have $|\tilde{E}| \leq |E| \cdot (|V| - 2)$. Thus the P_3-classes can be computed in $O(|V| \cdot |E|)$.

At each stage of Algorithm 3.2, we test whether G or $\overline{G}$ is disconnected. If so, we recursively compute the modular decomposition tree of the connected components of G or $\overline{G}$. Otherwise, if G is connected and coconnected, we scan the edges in G until we find an edge vw whose P_3-class $C^*(vw)$ satisfies $|V(C^*(vw))| = |V|$. This can be done in $O(|V|^2)$. By Theorem 3.3.8, the maximal connected homogeneous sets are the connected components of $G' = (V, E - C^*(vw))$.

The same procedure applied to the complement computes the maximal coconnected homogeneous sets in $O(|V|^2)$. From the maximal connected and the maximal coconnected homogeneous sets, the maximal homogeneous sets are easily found in $O(|V|^2)$. The overall running time of our algorithm is therefore $O(|V|^3)$.

3.3.2 Recognition and orientation algorithms

A necessary condition for a graph to be a comparability graph is that each of its P_3-classes can be transitively oriented. If a graph has no or precisely one P_3-class, then a transitive orientation is easy to calculate because the orientation of one edge in a P_3-class forces the orientation of all other edges in the same P_3-class. We show that the other cases can be reduced to this one.

Suppose that a graph $G = (V, E)$ has at least two P_3-classes. By Theorem 3.3.6, one P_3-class, say C^*, does not cover the whole graph, thus G has a homogeneous set $V(C^*)$. If a graph has a homogeneous set H, we proceed as follows.

(*i*) Replace H with a marker vertex h.

(*ii*) Compute a transitive orientation of G_H and G_{V-H+h}.

(*iii*) Construct a transitive orientation of G by directing
vw with $v, w \in H$ as in G_H,
vw with $v, w \in V - H$ as in G_{V-H+h},
vw with $v \in V - H$ and $w \in H$ as vh in G_{V-H+h}.

If G has a transitive orientation, the same holds for G_H and G_{V-H+h} as they are induced subgraphs. Surprisingly, the converse holds as well.

Lemma 3.3.9 *If the orientation of G_H and G_{V-H+h} is transitive, then (iii) gives a transitive orientation of G.*

Proof. To begin with, we show that no P_3 abc is oriented $a \to b \to c$. This is obvious for a P_3 with all its vertices in H and a P_3 with at most one vertex in H because a corresponding P_3 exists in G_H or G_{V-H+h}. The remaining P_3s have precisely two vertices in H. Since H is homogeneous, such a P_3 has $a, c \in H$ and $b \notin H$. It is therefore oriented $a \to b \leftarrow c$ or $a \leftarrow b \to c$.

Now suppose the constructed orientation $\vec{G}$ is cyclic. As the orientations of G_H and G_{V-H+h} are acyclic, every cycle in $\vec{G}$ contains vertices in $V - H$ and edges with both endpoints in H. Consider a shortest cycle in $\vec{G}$ and let $v \to \cdots \to w$ denote a longest part of it with vertices in H. Furthermore, let u be the predecessor of v in this cycle. Since H is homogeneous, the edge uw exists and, by construction, $u \to w$ in $\vec{G}$. Therefore our cycle can be shortened by substituting $u \to w$ for $u \to v \to \cdots \to w$, a contradiction. □

Note that the above lemma proves Theorem 3.3.2 because (*a*) if the P_3-classes of G can be transitively oriented, the same holds for the P_3-classes of $G_{V(C^*)}$ and $G_{V-V(C^*)+h}$, and (*b*) this division into subproblems can be repeated until the graph has at most one P_3-class.

Instead of explicitly performing the substitution of marker vertices for homogeneous sets, GOLUMBIC [27] proposed an algorithm that does this implicitly by removing P_3-classes from the graph. His algorithm for computing the transitive orientation of a graph is given below.

orient(G)

input: a graph $G = (V, E)$
output: a transitive orientation of G (if such an orientation exists)

(1) **while** $E \neq \emptyset$ **do**
(2) choose an edge vw in E;
(3) orient the P_3-class $C^*(vw)$ of $G = (V, E)$;
(4) $E \leftarrow E - C^*(vw)$;
(5) **od**

Algorithm 3.3

The complexity of the above algorithm is $O(|V| \cdot |E|)$. To prove its correctness, let $H = V(C^*(vw))$. From Lemma 3.3.9 follows that a transitive orientation of G exists such that the orientation of the P_3-classes in G_H is independent of the orientation of the other P_3-classes. The only restriction imposed on the orientation of the P_3-classes not in G_H is that edges between vertices in H and a vertex in $V - H$ are directed in the same way. This constraint is satisfied because G_H is coconnected after the removal of $C^*(vw)$.

Now consider the orientation of G_H. Again Lemma 3.3.9 guarantees that we can orient G_H by orienting the P_3-classes in a maximal homogeneous set of G_H independently from the other P_3-classes of G_H. So it remains to show that the P_3-classes not contained in maximal homogeneous sets of G_H are oriented properly. We do this by showing that $C^*(vw)$ is the only P_3-class not in a maximal homogeneous set.

Note that G_H is connected. If G_H is coconnected, then Theorem 3.3.8 guarantees that all edges not in a maximal homogeneous set belong to the same P_3-class. If G_H is codisconnected, then it is easy to see that G_H is the join of two coconnected graphs G_{H_1} and G_{H_2}. Hence H_1 and H_2 are maximal homogeneous sets and every edge between H_1 and H_2 belongs to $C^*(vw)$, thus Algorithm 3.3 is correct.

At this point, it should be mentioned that there is a vast literature on the recognition and orientation of comparability graphs, and that faster but much more complicated algorithms for the recognition of comparability graphs are known. The best results are due to McConnell and Spinrad [55]. In 1997, they presented the first linear time algorithm for computing a transitive orientation of a comparability graph.

To recognize comparability graphs, however, it has to be tested whether the computed orientation is transitive. This problem can be reduced to (boolean) matrix multiplication, for which the fastest algorithms run in $O(|V|^{2.38})$ [16].

3.4 Special prime graphs

The purpose of this section is to provide further results on prime graphs. We start with split graphs, which play a key role in the generalized modular decomposition given in the next chapter.

Theorem 3.4.1 (FÖLDES AND HAMMER) *For a graph G, the following conditions are equivalent.*

(*i*) *G is a split graph.*

(*ii*) *G and $\overline{G}$ are triangulated.*

(*iii*) *G contains no $2K_2$, C_4 or C_5.*

In Section 2.3, we have shown that we can test in $O(|V| + |E|)$ whether a graph or its complement is triangulated, thus split graphs can be recognized in linear time. Furthermore, we claim that every split graph G admits a split partition $V^1 + V^2$ such that V^1 consists of the first $\omega(G)$ vertices in descending degree order; thus the split partition can also be calculated in linear time.

To prove our claim, let $V_1 + V_2$ denote a split partition such that V_1 is a maximum clique. Clearly the vertices with degree greater than $|V_1| - 1$ belong to V_1 and the vertices with degree less than $|V_1| - 1$ belong to V_2. Let $v_1 \in V_1$ and $v_2 \in V_2$ be vertices with $\deg(v_1) = \deg(v_2) = |V_1| - 1$. Since v_1 misses every vertex in V_2, we find that $V_1 - v_1 + v_2$ is a clique and $V_2 - v_2 + v_1$ is a stable set, so $V_1' = V_1 - v_1 + v_2$ and $V_2' = V_2 - v_2 + v_1$ is again a split partition such that V_1' is a maximal clique. Thus our claim follows by induction.

Now suppose that a prime graph contains a C_4. In [40], HOÀNG AND REED showed that such a graph must also contain one of the graphs F_1, F_2 or F_3 in Figure 3.1. The next theorem provides the corresponding complexity result.

Theorem 3.4.2 *If a C_4 in a prime graph $G = (V, E)$ is given, then an F_1, F_2 or F_3 can be found in $O(|V| + |E|)$.*

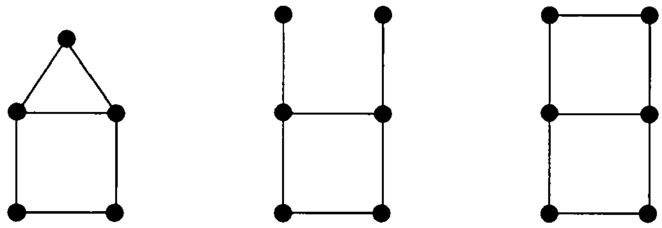


Figure 3.1: *The graphs contained in a prime graph with C_4s.*

Proof. Let v_0, v_1, v_2, v_3 denote the given C_4 in G. We proceed as HOÀNG AND REED in the proof of Claim 3.5 [40].

Step 1: Compute the set A of all vertices that see both v_1 and v_3.

Step 2: Compute the vertices in the connected component A_1 of v_0 and v_2 in $\overline{G}_A$.

Step 3: Find an A_1-partial vertex x in $V - A$. Since A_1 is not homogeneous, such a vertex exists.

Step 4: Find two nonadjacent vertices $w_1, w_2 \in A_1$ such that x sees w_1 and misses w_2. Since x belongs to $V - A$, it cannot see v_1 and v_3. If x sees precisely one of the two vertices v_1 and v_3, then $\{x, v_1, v_3, w_1, w_2\}$ induces an F_1 and we are done. So suppose that x misses v_1 and v_3.

Step 5: Compute the set B of all vertices that see w_1 and w_2 and miss x.

Step 6: Compute the connected component B_1 of v_1 and v_3 in $\overline{G}_B$.

Step 7: Find a W_1-partial vertex y in $V - W$. Since W_1 is not homogeneous, such a vertex exists.

Step 8: Find two nonadjacent vertices $u_1, u_2 \in W_1$ such that y sees u_1 and misses u_2. If y sees w_1 and w_2, then y sees x as well and $\{x, y, w_1, u_2, w_2\}$ induces an F_1. Similarly, if y sees precisely one of the two vertices w_1 and w_2, then $\{y, u_1, u_2, w_1, w_2\}$ induces an F_1. Finally, if y misses w_1 and w_2, then $\{x, y, u_1, u_2, w_1, w_2\}$ induces an F_2 or an F_3, depending on whether x sees y.

Clearly A and G_A can be computed in linear time. By Lemma 2.3.1, the connected components of $\overline{G}_A$ are obtained in $O(|V| + |E|)$, hence A_1 and x can be found in linear time. In Section 2.3 Remark 3, we have explained how to compute a BFS-forest of the complement in $O(|V|+|E|)$. A spanning tree of $\overline{G}_{A_1}$ is readily obtained from a BFS-tree of $\overline{G}_A$ by making directed edges undirected. So a path from a vertex that sees x to a vertex that misses x is available and, following this path, w_1 and w_2 can be computed in linear time. Step 5 to 8 are analog to Step 1 to 4 and have therefore the same complexity. □

Now suppose that $\overline{G} = (V, \overline{E})$ is given. Step 1 to 8 in the above proof can still be done in $O(|V| + |\overline{E}|)$. Since the complement of a C_4 is a $2K_2$, Theorem 3.4.2 translates into

Corollary 3.4.3 *If a $2K_2$ in a prime graph $G = (V, E)$ is given, then an $\overline{F}_1$, $\overline{F}_2$ or $\overline{F}_3$ can be found in $O(|V| + |E|)$.*

By Theorem 2.2.5, a cycle in a graph that is not triangulated is obtained in linear time. Furthermore, by Corollary 2.3.6, the same complexity result holds for the complement. By observing that an F_3 and a cycle of length greater than 5 contains a P_5, that the complement of a C_5 is again a C_5 and that an F_1 is the complement of a P_5, we derive

Theorem 3.4.4 *Let G be a prime graph that is not split. Then a C_5, P_5, $\overline{P}_5$, F_2 or $\overline{F}_2$ can be found in linear time.*

Chapter 4

Generalizations of the modular decomposition

In the first section of this chapter, we propose a straight-forward generalization of modules and discuss which measures have to be taken in order to obtain a unique decomposition that generalizes the modular decomposition. We then restrict ourselves to generalized modules that induce bipartite graphs or split graphs, which is why we call them bipartite modules and split modules, respectively.

In the second section, we show that our bipartite modules imply a unique decomposition of nonbipartite prime graphs and we briefly discuss how this decomposition can be computed. In the third section, we prove similar theorems for split modules and nonsplit prime graphs. As it turns out, the arising decomposition generalizes BABEL AND OLARIU's separable-homogeneous decomposition [5] as well as the decomposition found by RASCHLE AND SIMON [67]. Computational aspects of this decomposition, however, are only discussed in the next chapter when the required results on the P_4-structure are available.

In the last section, we show that the decomposition into bipartite modules, split modules and the complement of bipartite modules can be combined to obtain a new unique decomposition. We do this by proving that bipartite modules, split modules and the complement of bipartite modules do not intersect if the given graph is prime.

4.1 Generalized modules

A module of a graph $G = (V, E)$ as defined in Section 3.2 is a nonempty vertex set H such that no H-partial vertex exists, that is, no vertex in $V - H$ distinguishes between vertices in H. This special neighborhood relation between the vertices in $V - H$ and those in H makes it possible to solve optimization problems with divide and conquer methods. For instance, a maximum weighted clique of G can be found by computing a maximum weighted clique in G_{V-H+h} where G_{V-H+h} denotes the graph after replacing H with a marker vertex h and h has the weight of a maximum weighted clique in G_H.

The substitution of marker vertices for modules can also be used to test isomorphism between graphs. For this purpose, some modules have to be identified which yield a unique decomposition tree (isomorphism between trees can be tested in polynomial time [2]). Clearly, those modules must be nontrivial and maximal with respect to set inclusion, i.e., those modules must be maximal homogeneous sets. These requirements are already sufficient for connected and coconnected graphs because

(*i*) the union of intersecting modules is a module (Fact 3.2.1), and

(*ii*) the union of intersecting modules that do not contain each other induces a disconnected or codisconnected graph (Fact 3.2.2).

The above statement guarantees that the maximal nontrivial modules of a connected and coconnected graph are disjoint: From (*i*), it follows that maximal modules are disjoint, and (*ii*) implies that the union of intersecting nontrivial modules is again a nontrivial module if the given graph is connected and coconnected.

A straightforward generalization of modules is to allow vertices in $V - H$ to distinguish vertices in H.

Definition 4.1.1 *A nonempty vertex set H of a graph $G = (V, E)$ is a k-module if a partition $H = H^1 + H^2 + \cdots H^k$ exists such that no vertex in $V - H$ is H^i-partial for $i = 1, \ldots, k$.*

According to Definition 4.1.1, classical modules are 1-modules. In this chapter, only 2-modules are considered, that is, vertices in $V - H$ distinguish at most two types of vertices in H. In the following, we usually write H if we refer to a 1-module and $W = W^1 + W^2$ if we refer to a 2-module.

To replace a 2-module $W = W^1 + W^2$, (at least) two marker vertices are required, one for W^1 and another for W^2. A trivial 2-module therefore contains less than three or all vertices of the graph. In analogy to 1-modules, we call nontrivial 2-modules 2-homogeneous sets.

Note that the special neighborhood relation between vertices in a 2-module W and vertices in $V - W$ still allows us to solve optimization problems with divide and conquer strategies. For instance, a maximum weighted clique of $G = (V, E)$ can be found by computing a maximum weighted clique in $G_{V-W+w_1+w_2+w_3}$ where w_1 stands for a maximum weighted clique in G_{W_1}, w_2 for a maximum weighted clique in G_{W_2} and w_3 for a maximum weighted clique in G_W.

To obtain a unique decomposition tree, we only consider maximal 2-homogeneous sets. Maximal 2-homogeneous sets, however, need not be disjoint: Given two intersecting 2-modules $A = A^1 + A^2$ and $B = B^1 + B^2$, it is possible that there are vertices x and y in $V - A - B$ such that x is A-partial but not B-partial whereas y is B-partial but not A-partial, hence x and y are $A \cup B$ partial but do not distinguish the same vertices in $A \cup B$, thus $A \cup B$ is not a 2-module. To avoid the above counterexample, it is necessary to require that

$$\text{If } A^1 \cap B^1 \neq \emptyset \text{ then } A^2 \cap B \neq \emptyset \text{ or } A \cap B^2 \neq \emptyset \tag{4.1}$$

for every labeling of the partition $A^1 + A^2$ and $B^1 + B^2$. It is also easy to see that if 2-modules A and B satisfy (4.1), then their union is indeed a 2-module, thus (4.1) is equivalent to (i) for 2-modules. So we are looking for constraints on 2-modules that imply (4.1).

If we allow vertices in W^1 not to be W^2-partial, then intersecting 2-modules A and B could satisfy $A \cap B = A^1 \cap B^1$ and no vertex in $A^1 \cap B^1$ is A^2-partial or B^2-partial. In this scenario, it seems to be hard to find constraints that guarantee (4.1). We therefore require that, in a 2-module W,

$$\text{every vertex in } W^1 \text{ must be } W^2\text{-partial} \tag{4.2}$$

and vice versa. The next lemma proves that (4.2) is indeed sufficient.

Lemma 4.1.2 *The union of intersecting 2-modules that satisfy (4.2) is again a 2-module.*

Proof. Let $v \in A^1 \cap B^1$ and suppose that $A^2 \cap B = \emptyset = A \cap B^2$. Since v is A^2-partial, vertices $x, y \in A^2$ exist such that v sees x and misses y. Furthermore, $A^2 \cap B = \emptyset$ and B is a 2-module, so x sees every vertex in B^1 and y misses every vertex in B^1, hence every vertex in B^1 is A^2-partial and therefore $B^1 \subseteq A^1$. The symmetric argument asserts $A^1 \subseteq B^1$, thus x sees every vertex in A^1, a contradiction to our assumption that every vertex in A^2 is A^1-partial. □

To study (4.2) in more detail, we define an AC_4 (*alternating cycle of length* 4) to be a sequence of four distinct vertices x, v, w, y such that vw and xy are edges whereas xv and wy are nonedges. We write $vw \parallel xy$ if x, v, w, y is an AC_4 and $vw \parallel yx$ if y, v, w, x is an AC_4.

Lemma 4.1.3 *If $W = W^1 + W^2$ satisfies (4.2), then there is an AC_4 $ab \parallel cd$ with $a, d \in W^1$ and $b, c \in W^2$.*

Proof. Suppose that a vertex v in W does not belong to an AC_4 $ab \parallel cd$ with $b, c \in W^1$ and $a, d \in W^2$. Without loss of generality, we may assume that v belongs to W^2. Then v partitions W^1 into nonempty sets $A^1 = W^1 \cap N(v)$ and $B^1 = W^1 \cap \overline{N}(v)$.

Let $A^2 = W^2 \cap \overline{N}(A^1)$ and $B^2 = W^2 \cap N(B^1)$. Since every vertex in W^1 is W^2-partial, the vertex sets A^2 and B^2 are nonempty. Furthermore there are no edges between vertices in A^2 and vertices in B^1, for otherwise v would belong to an AC_4. Similarly, every edge between vertices in B^2 and vertices in A^1 exists. It is now easy to verify that $W - v = W^1 + (W^2 - v)$ still satisfies (4.2).

By repeatedly removing vertices that do not belong to an AC_4 $ab \parallel cd$ with $a, d \in W^1$ and $b, c \in W^2$, we end up with a vertex set $W = W^1 + W^2$ (not necessarily a 2-module) that satisfies (4.2) and every vertex belongs to an AC_4. □

By requiring (4.2) for 2-modules, we established an equivalent statement of (*i*) for 2-modules. Regarding (*ii*), however, this is not so easy: For every graph $G = (V, E)$ and every vertex $v \in V$, the set $V - v$ is 2-homogeneous, and $V - v$ satisfies (4.2) for almost every graph. To make the decomposition unique for a large number of graphs, we have to find further constraints on 2-modules.

In the rest of this chapter, we discuss decompositions that are unique for prime graphs which are not split, not bipartite or not cobipartite, respectively. So we are looking for constraints on 2 modules which imply that the union of 2-modules which do not contain each other induces a split graph, a bipartite graph or a cobipartite graph, respectively. In the following, we require that the 2-modules themselves induce split graphs, bipartite graphs or cobipartite graphs. In other words, we require that W^1 (W^2) is a clique or a stable set.

4.2 Bipartite modules

In this section, we consider 2-modules W for which W^1 and W^2 are stable sets and for which (4.2) holds. To simplify our terminology, we call those 2-modules bipartite modules:

Definition 4.2.1 *A vertex set W of a graph $G = (V, E)$ is a* bipartite module *if a partition $W = W^1 + W^2$ (called* bipartition*) exists such that*

(*i*) *W^1 and W^2 are nonempty stable sets,*

(*ii*) *every vertex in W is W^1-partial or W^2-partial, and*

(*iii*) *every vertex in $V - W$ is neither W^1-partial nor W^2-partial.*

A bipartite module W is called bipartite-homogeneous if W is a proper subset of V.

Clearly nontrivial bipartite modules are bipartite-homogeneous sets and vice versa. Furthermore, note that the bipartition of a bipartite module is unique.

In the following, we show that the maximal bipartite-homogeneous sets of a nonbipartite prime graph are disjoint. The next lemma prepares this proof.

Lemma 4.2.2 *Let A and B be bipartite modules with bipartitions $A = A^1 + A^2$ and $B = B^1 + B^2$. If $A^1 \cap B^1 \neq \emptyset$ and neither A nor B is a 1-module, then*

(*i*) *$A^2 \cap B^2 \neq \emptyset$ and*

(*ii*) *$A^1 \cap B^2 = \emptyset = A^2 \cap B^1$.*

Proof. We prove (*i*) first. Suppose the contrary, that is, $A^2 \cap B^2 = \emptyset$. Let b denote a vertex in $A^1 \cap B^1$. Since b is A^2-partial and B^2-partial, there are vertices $a \in A^2$ and $c \in B^2$ which see b. By our assumption, c cannot be in A^2, hence c belongs to $V - A$, thus c is A^1-universal. Furthermore, because B^1 is stable, a is in $V - B$ and is therefore B^1-universal.

Now a is A^1-partial, so there is a vertex d in A^1 that is missed by a. On the one hand, d cannot belong to B^1 because a is B^1-universal. On the other hand, d cannot belong to B^2 because $c \in B^2$ sees d.

So $d \notin B$, hence d is B^2-universal, thus $A \cap B^2 = \emptyset$. Since b is B^2-partial, a vertex $e \in B^2$ exists which is missed by b. Now e is a vertex in $V - A$ that sees d but misses b, a contradiction because no vertex in $V - A$ may be A^1-partial.

It remains to prove (ii). Because of symmetry, it suffices to show that $A^1 \cap B^2 = \emptyset$. Suppose the contrary. Then there are vertices $a \in A^1 \cap B^1$, $b \in A^1 \cap B^2$ and $c \in A^2 \cap B^2$ (the latter because of (i)). Since b is B^1-partial, there is a vertex $d \in B^1$ which sees b. But d cannot be in $V - A$, for otherwise d would see a, a contradiction because B^1 is stable. Hence $d \in B^1 \cap A^2$.

Now every A-partial vertex is B^1-partial and B^2-partial, so it must belong to B. But this is impossible because B^1 and B^2 are stable sets. Therefore A is a 1-module, a contradiction to our assumption. □

Let A and B be two intersecting bipartite modules of a prime graph $G = (V, E)$. Without loss of generality, we may assume that the bipartitions $A = A^1 + A^2$ and $B = B^1 + B^2$ are labeled such that $A^1 \cap B^1 \neq \emptyset$. Since neither A nor B is a 1-module, it follows from Lemma 4.2.2 that $(A^1 \cup B^1) + (A^2 \cup B^2)$ is a partition of $A \cup B$ and that vertices $v \in A^1 \cap B^1$ and $w \in A^2 \cap B^2$ exist. Clearly every vertex in $A^1 \cup B^1$ is $A^2 \cup B^2$-partial and vice versa. We claim that $A^1 \cup B^1$ and $A^2 \cup B^2$ are stable sets.

If two vertices a and b in $A^1 \cup B^1$ are adjacent, then a and b do not belong to $A^1 \cap B^1$. Because of symmetry, we may assume that $a \in A^1 - B^1$ and $b \in B^1 - A^1$. But a misses v and therefore every vertex in B^1, a contradiction. So $A^1 \cup B^1$ is a stable set. By symmetry, the same holds for $A^2 \cup B^2$.

Since $A^1 \cap B^1 \neq \emptyset \neq A^2 \cap B^2$, a vertex in $V - (A \cup B)$ is A^1-universal (A^1-null, A^2-universal, A^2-null) if and only if it is B^1-universal (B^1-null, B^2-universal, B^2-null). Therefore the following analog of Fact 3.2.1 holds.

buildBipartiteModTree(G)

input: a graph $G = (V, E)$
output: the root of the bipartite modular decomposition tree of G

---

if $|V| = 1$ **then**
  let v be the vertex in V;
  return an empty node labeled v;
elsif G is disconnected **then**
  let $G_1, G_2, \ldots, G_t$ be the connected components of G;
  let $r_i =$ buildBipartiteModTree(G_i) for $i = 1, \ldots, t$;
  return a 0-node with children $r_1, r_2, \ldots, r_t$
elsif $\overline{G}$ is disconnected **then**
  let $\overline{G}_1, \overline{G}_2, \ldots, \overline{G}_t$ be the connected components of $\overline{G}$;
  let $r_i =$ buildBipartiteModTree(G_i) for $i = 1, \ldots, t$;
  return a 1-node with children $r_1, r_2, \ldots, r_t$
else (* G and $\overline{G}$ are connected and $|V| > 1$ *)
  let $G' = (V', E')$ be the characteristic graph of G;
  if G' is a bipartite graph **then**
    let $H_1, \ldots, H_t$ be the maximal proper modules of G;
    let $r_i =$ buildBipartiteModTree(G_{H_i}) for $i = 1, \ldots, t$;
    return a 2-node with children $r_1, \ldots, r_t$
  else (* G' is not bipartite *)
    let $B_1, \cdots, B_k$ be the vertex sets of G that correspond
      to maximal bipartite-homogeneous sets of G';
    let $b_i =$ buildBipartiteModTree(G_{B_i}) for $i = 1, \ldots, t$;
    let $H_1, \ldots, H_t$ be those maximal proper modules of G
      which are not contained in $B_1, \ldots, B_k$;
    let $r_i =$ buildBipartiteModTree(G_{H_i}) for $i = 1, \ldots, t$;
    return a 3-node with children $b_1, \ldots, b_k, r_1, \ldots, r_t$
  fi
fi

Algorithm 4.1

Fact 4.2.3 *If bipartite modules $A = A^1 + A^2$ and $B = B^1 + B^2$ of a prime graph intersect, then $A \cup B = (A^1 \cup B^1) + (A^2 \cup B^2)$ is again a bipartite module.*

The uniqueness of the decomposition of nonbipartite prime graphs into maximal bipartite-homogeneous sets now follows immediately.

Theorem 4.2.4 *The maximal bipartite-homogeneous sets of a prime nonbipartite graph are disjoint.*

Proof. Suppose that two maximal bipartite-homogeneous sets A and B intersect. Then $A \cup B$ is a bipartite module, hence $A \cup B = V$. But this is a contradiction because a bipartite module induces a bipartite graph. □

The corresponding decomposition is given in Algorithm 4.1. In the rest of this section, we briefly discuss some aspects of bipartite modules with respect to the computation of maximal bipartite-homogeneous sets. For this purpose, the following definition is useful.

Definition 4.2.5 *Two $2K_2$s are* adjacent *if they have three common vertices, and $2K_2$-components are the equivalence classes of the transitive closure of the adjacency relation between $2K_2$s.*

Let W be bipartite module. By Lemma 4.1.3, W contains a $2K_2$. Furthermore, it is easy to see that if two $2K_2$s are adjacent, then either both belong to W or none of them is in W. By induction, this holds for all $2K_2$s in the same $2K_2$-component.

Let C^* denote a $2K_2$-component and let $V(C^*)$ stand for the set of vertices which belong to some $2K_2$ in C^*. If a $2K_2$ in C^* belongs to W, then $V(C^*) \subseteq W$ as mentioned above. Moreover, in this case, it is easy to see that no vertex in $W^1 - V(C^*)$ is $W^2 \cap V(C^*)$-partial. Similarly, no vertex in $W^1 - V(C^*)$ is $W^1 \cap V(C^*)$-partial. Therefore $V(C^*)$ is a bipartite-homogeneous set.

Fact 4.2.6 *If a $2K_2$ in a $2K_2$-component C^* belongs to a bipartite-homogeneous set W, then $V(C^*) \subseteq W$ and $V(C^*)$ is also bipartite-homogeneous.*

To compute the maximal bipartite-homogeneous sets of a nonbipartite prime graph, we can proceed as follows. First, we compute the $2K_2$-components and test whether they induce bipartite-homogeneous sets. Second, we select those bipartite-homogeneous sets which are maximal with respect to set inclusion. Third, we take the union if some of those sets intersect (by Fact 4.2.3, the union is bipartite-homogeneous). Fourth, we take the union of disjoint sets if the union is again bipartite-homogeneous.

If a maximal homogeneous set W is not one of those computed so far, then W contains vertices W' that do not belong to any $2K_2$ in W. Consider again the proof of Lemma 4.1.3. It should be clear that A and B are bipartite modules if W is a bipartite module. So we know that $W - W'$ consists of precisely two disjoint bipartite homogeneous sets $A = A^1 + A^2$ and $B = B^1 + B^2$ and every vertex in A and B is in a $2K_2$ in A and B, respectively. In other words, A and B belong to the already computed bipartite-homogeneous sets.

Again following the proof of Lemma 4.1.3, it is easy to see that every vertex in W' must be $A^1 \cup B^1$-partial or $A^2 \cup B^2$-partial. To find the maximal homogeneous sets, it therefore suffices to consider all pairs of bipartite-homogeneous sets A and B and to compute the set W' of vertices that are $A^1 \cup B^1$-partial or $A^2 \cup B^2$-partial. It then remains to test whether $A \cup B \cup W'$ is bipartite-homogeneous.

Since all these steps can be carried out in polynomial time, the bipartite-modular decomposition can be computed in polynomial time. In fact, a more detailed analysis reveals that the bipartite-modular decomposition is in $O(|V|^5)$.

4.3 Split modules

In this section, we consider 2-modules W for which W^1 is a clique and W^2 is a stable set and for which (4.2) holds. In other words, W induces a split graph $G_W = (W^1, W^2, E(W))$.

For this type of 2-modules, a statement similar to Fact 4.2.3 does not hold. For instance, we can choose $A^1 + A^2 = \{b, c\} + \{a, d\}$ and $B^1 + B^2 = \{c, d\} + \{b, e\}$ of a C_5 a, b, c, d, e, so A and B are 2-modules of the required type but $A \cup B$ does not induce a split graph.

As it turns out, the above problem appears only if the partitions $A = A^1 + A^2$ and $B = B^1 + B^2$ are unrelated to the A-partial and B-partial vertices. So we additionally require that every W-partial vertex must be W^1-universal and W^2-null.

Definition 4.3.1 *A vertex set W of a graph $G = (V, E)$ is a* split module *if a partition $W = W^1 + W^2$ (called* split-partition*) exists such that*

(i) W^1 is a nonempty clique and W^2 is a nonempty stable set,

(ii) every vertex in W is W^1-partial or W^2-partial, and

(iii) $V - W$ *can be partitioned into sets* P, Q *and* R *where*

the vertices in P *are* W*-universal,*

the vertices in Q *are* W*-null and*

the vertices in R *are* W_1*-universal and* W_2*-null.*

A split module W *is* strict *if no edges between* Q *and* R *exist. Furthermore, a (strict) split module* W *is called (strict) split homogeneous if* W *is a proper subset of* W.

First, we observe that the split-partition $W^1 + W^2$ of a split module W is unique. This is clear if every vertex in W belongs to a P_4 in W, for every P_4 in W must have its midpoints in W^1 and its endpoints in W^2. On the other hand, Lemma 4.1.3 guarantees that every split module contains a P_4. The uniqueness of the split partition now follows from the proof of Lemma 4.1.3 as we can uniquely determine to which set the vertex v belongs given we know the split partition of $W - \{v\}$.

Second, note that split modules are split modules in the complement. This, however, does not hold for strict split modules: A strict split module of $\overline{G}$ is a split module of G such that all edges between P and R exist.

In the following, we show that the union of intersecting split modules is again a split module. Lemma 4.3.2 prepares this proof.

Lemma 4.3.2 *Let* A *and* B *be intersecting split modules with split-partitions* $A^1 + A^2$ *and* $B^1 + B^2$. *Then*

(i) $A^1 \cap B^1 \neq \emptyset \neq A^2 \cap B^2$, *and*

(ii) $A^1 \cap B^2 = \emptyset = A^2 \cap B^1$.

Proof. We prove *(ii)* first. Because of symmetry, it suffices to show that $A^1 \cap B^2 = \emptyset$. Suppose the contrary and let b denote a vertex in $A^1 \cap B^2$. Since b is A^2-partial, there is a vertex $a \in A^2$ that sees b. Furthermore, a is A^1-partial, so a vertex $c \in A^1$ exists which is missed by a.

If a belongs to B, then $a \in B^1$ because B^2 is stable. Since c sees $b \in B^2$ and misses $a \in B^1$, we infer that c belongs to B. But this is impossible because B^1 is a clique and B^2 a stable set.

So we know that a is not in B, hence a is B-universal. Therefore $c \notin B$ and no vertex in A^2 belongs to B. Since b is B^1-partial, there is a vertex $d \in B^1$ that misses b. Then $d \notin A^2$ and, as A^1 is a clique, $d \notin A^1$. So d misses c, a contradiction to the fact that c is B-universal.

It remains to prove (i). Suppose that $A^1 \cap B^1 = \emptyset$. Then $A \cap B = A^2 \cap B^2 \neq \emptyset$ and, by (ii), $A^1 \cap B^2 = \emptyset = A^2 \cap B^1$. Let $b \in A^2 \cap B^2$. Since b is A^1-partial, there are vertices a and c in A^1 such that b sees a and misses c. Then a is B^2-universal and c is B^2-null. So every vertex in B^2 is A^1-partial, which implies $B^2 \subseteq A^2$. This is a contradiction because every vertex in B^1 is B^2-partial but it must not be A^2-partial (as such a vertex does not belong to A). □

Let A and B be intersecting split modules. Then Lemma 4.3.2 implies that $(A^1 \cup B^1) + (A^2 \cup B^2)$ is a partition of $A \cup B$ and that vertices $v \in A^1 \cap B^1$ and $w \in A^2 \cap B^2$ exist. Clearly every vertex in $A^1 \cup B^1$ is $A^2 \cup B^2$-partial and vice versa. We claim that $A^1 \cup B^1$ is a clique and that $A^2 \cap B^2$ is a stable set.

If two vertices a and b in $A^1 \cup B^1$ are not adjacent, then a and b do not belong to $A^1 \cap B^1$. Because of symmetry, we may assume that $a \in A^1 - B^1$ and $b \in B^1 - A^1$. But a sees v and therefore every vertex in B^1, a contradiction. Similarly, if two vertices a and b in $A^2 \cup B^2$ are adjacent, we may assume that $a \in A^2 - B^2$ and $b \in B^2 - A^2$. But a misses w and therefore every vertex in B^2, again a contradiction.

Since $A^1 \cap B^1 \neq \emptyset$ and $A^2 \cap B^2 \neq \emptyset$, a vertex in $V - (A \cup B)$ is A-universal if and only if it is B-universal, and it is A-null if and only if it is B-null. Therefore the following analog of Fact 3.2.1 and Fact 4.2.3 holds.

Fact 4.3.3 *If (strict) split modules $A = A^1 + A^2$ and $B = B^1 + B^2$ intersect, then $A \cup B = (A^1 \cup B^1) + (A^2 \cup B^2)$ is again a (strict) split module.*

The uniqueness of the decomposition of prime nonsplit graphs is established by the next theorem, the analog of Theorem 3.2.3.

Theorem 4.3.4 *The maximal (strict) split-homogeneous sets of a nonsplit graph are disjoint.*

Proof. Suppose that two maximal (strict) split-homogeneous sets A and B intersect. Then $A \cup B$ is a (strict) split module, hence $A \cup B = V$. But this is a contradiction because a (strict) split module induces a split graph. □

buildSplitModTree(G)

input: a graph $G = (V, E)$
output: the root of the split modular decomposition tree of G

```
if |V| = 1 then
   let v be the vertex in V;
   return an empty node labeled v;
elsif G is disconnected then
   let G_1, G_2, ... , G_t be the connected components of G;
   let r_i = buildSplitModTree( G_i ) for i = 1, ... , t;
   return a 0-node with children r_1, r_2, ... , r_t
elsif co-G is disconnected then
   let co-G_1, co-G_2, ... , co-G_t be the connected components of co-G;
   let r_i = buildSplitModTree( G_i ) for i = 1, ... , t;
   return a 1-node with children r_1, r_2, ... , r_t
else (* G and co-G are connected and |V| > 1 *)
   let G' = (V', E') be the characteristic graph of G;
   if G' is a split graph then
      let H_1, ... , H_t be the maximal proper modules of G;
      let r_i = buildSplitModTree( G_{H_i} ) for i = 1, ... , t;
      return a 2-node with children r_1, ... , r_t
   else (* G' is not split *)
      let S_1, ··· , S_k be the vertex sets of G that correspond
         to maximal split-homogeneous sets of G';
      let s_i = buildSplitModTree( G_{S_i} ) for i = 1, ... , t;
      let H_1, ... , H_t be those maximal proper modules of G
         which are not contained in S_1, ... , S_k;
      let r_i = buildSplitModTree( G_{H_i} ) for i = 1, ... , t;
      return a 3-node with children s_1, ... , s_k, r_1, ... , r_t
   fi
fi
```

Algorithm 4.2

The decomposition derived so far is given in Algorithm 4.2. It generalizes BABEL AND OLARIU's separable-homogeneous decomposition [5] for nonsplit prime graphs as their "maximal separable-homogeneous sets" correspond to those maximal split-homogeneous sets W in the characteristic graph in which every vertex belongs to a P_4 in G_W. Independently, RASCHLE AND SIMON [67] proposed the decomposition of prime graphs into "P_4-split graphs", which are strict split-homogeneous sets in the characteristic graph or its complement. To prove the uniqueness of the latter decomposition, we need the following lemma.

Lemma 4.3.5 *Let G be a prime graph and let A and B be strict split-homogeneous sets of G and $\overline{G}$, respectively. If A intersects B, then $A \cup B$ is a strict split module of G and $\overline{G}$ and either*

(i) $V = A \cup B$ or

(ii) there is precisely one vertex v in $V - A - B$, and v does not belong to any P_4 of G.

Proof. Let $A = A^1 + A^2$ and $B = B^1 + B^2$ be the split-homogeneous sets of G and $\overline{G}$, respectively. By Fact 4.3.3, $A \cup B$ is a split module. Furthermore no edges between A-partial and A-null vertices exist whereas all edges between B-partial and B-universal vertices are present, hence $A \cup B$ is a strict split module of G and $\overline{G}$.

Let R denote the set of $A \cup B$-partial vertices. Then $R \cup A \cup B$ is a module. But G is prime, thus $R \cup A \cup B = V$. Now R is a module, hence $|R| \leq 1$, thus either (i) or (ii) holds. □

Remark: Lemma 4.3.5(ii) can be used to decompose prime split graphs because the vertex v is unique. In fact, the modular decomposition together with this decomposition of prime split graphs is precisely JAMISON AND OLARIU's "homogeneous decomposition" [43]. BABEL AND OLARIU [5] further refined the decomposition of prime split graphs. Those results are discussed in Section 5.3 of the next chapter.

If Lemma 4.3.5 applies, then G is a split graph. Thus a strict split-homogeneous set of G cannot intersect a strict split-homogeneous set of $\overline{G}$ given G is prime nonsplit. Together with Theorem 4.3.4, this establishes the uniqueness of RASCHLE AND SIMON's decomposition.

Theorem 4.3.6 *If a prime graph G is not split, then the maximal strict split-homogeneous sets of G and $\overline{G}$ are disjoint.*

We conclude this section with discussing the similarities between modules and (strict) split modules. Since modules are modules in the complement, it seems at first glance that split modules are closer related to modules than strict split modules. On the other hand, given a homogeneous set H and a marker vertex $h \in H$, every P_4 of G has a corresponding P_4 either in G_{V-H+h} or in G_H. We show that a similar result holds for strict split-homogeneous sets of G and $\overline{G}$ but not for split-homogeneous sets.

Let $W = W^1 + W^2$ be a strict split module. Then every P_4 of G with at least one but not all its vertices in W is of the following type.

type (1)	wpq_1q_2	where $w \in W$, $p \in P$, $q_1 \in Q$, $q_2 \in Q$
type (2)	p_1wp_2q	where $p_1 \in P$, $w \in W$, $p_2 \in P$, $q \in Q$
type (3)	$p_1w_2p_2r$	where $p_1 \in P$, $w_2 \in W^2$, $p_2 \in P$, $r \in R$
type (4)	$w_2pr_1r_2$	where $w_2 \in W^2$, $p \in P$, $r_1 \in R$, $r_2 \in R$
type (5)	rw_1pq	where $r \in R$, $w_1 \in W^1$, $p \in P$, $q \in Q$
type (6)	rw_1pw_2	where $r \in R$, $w_1 \in W^1$, $p \in P$, $w_2 \in W^2$

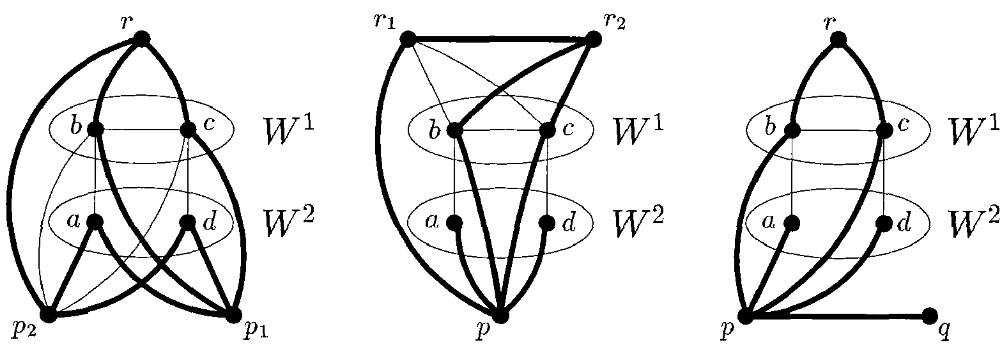


Figure 4.1: *The subgraphs induced by a P_4 of types (3) to (5).*

The graphs induced by a P_4 $abcd$ in G_W together with a P_4 of type (3) to (5) are depicted in Figure 4.1, (bold lines indicate edges in P_4s with vertices in $V - W$). The existence of a P_4 of type (3) to (5) implies a P_4 of type (6), and a P_4 of type (6) together with $abcd$ induces a graph called pyramid, see Figure 4.2.

A *pyramid abcdrp* is of a P_4 $abcd$ together with an $\{a, b, c, d\}$-universal vertex p and an $\{a, b, c, d\}$-partial vertex r which sees the midpoints of $abcd$ and misses its endpoints. The complement of a pyramid is a *net*, thus a net $abcdrq$ consists of a P_4 $abcd$ together with an $\{a, b, c, d\}$-null vertex q and an $\{a, b, c, d\}$-partial vertex r which sees the midpoints of $abcd$ and misses its endpoints, see Figure 4.2.

Given a strict split-homogeneous set $W = W^1 + W^2$, we can replace $W^1 + W^2$ with two nonadjacent marker vertices $w_1 \in W^1$ and $w_2 \in W^2$.

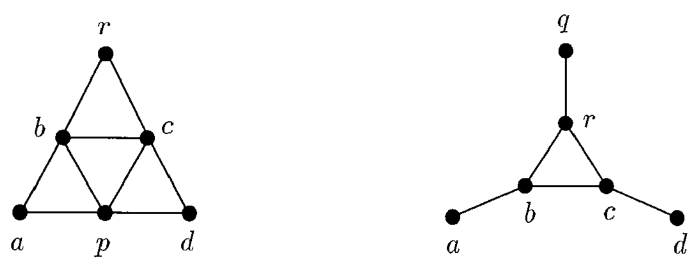


Figure 4.2: *A pyramid abcdrp and a net abcdrq.*

Then every P_4 of G has a corresponding P_4 either in $G_{V-W+w_1+w_2}$ or in G_W. Figure 4.3 illustrates the substitution of marker vertices for strict split-homogeneous sets of G or $\overline{G}$. The graph depicted in Figure 4.3(a) has a strict split-homogeneous set $A = \{d_1, d_2\} + \{e_1, e_2\}$ and a strict split-homogeneous set $B = \{a_1, a_2\} + \{b_1, b_2\}$ in the complement. Figure 4.3(b) shows the graph after the substitution of adjacent marker vertices a_1, b_1 for A and of nonadjacent marker vertices d_1, e_2 for B.

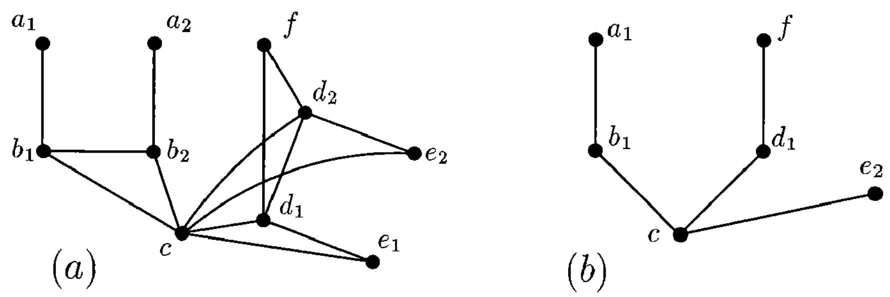


Figure 4.3: *The substitution of marker vertices for strict split-homogeneous sets of G and $\overline{G}$.*

If $W = W^1 + W^2$ is a split module, then a P_4 with at least one but not all its vertices in W is of type (1) to (6), or one of its edges has an endpoint in Q and the other in R. In the latter case, the following additional P_4s are possible.

type (7)	$w_1 r q_1 q_2$	where	$w_1 \in W^1$, $r \in R$, $q_1 \in Q$, $q_2 \in Q$
type (8)	$r_1 w_1 r_2 q$	where	$r_1 \in R$, $w_1 \in W^1$, $r_2 \in R$, $q \in Q$
type (9)	$w_2 p r q$	where	$w_2 \in W^2$, $p \in P$, $r \in R$, $q \in Q$
type (10)	$w_2 p q r$	where	$w_2 \in W^2$, $p \in P$, $q \in Q$, $r \in R$
type (11)	$p w_1 r q$	where	$p \in P$, $w_1 \in W^1$, $r \in R$, $q \in Q$

type (12) $w_2 w_1 r q$ where $w_2 \in W^2$, $w_1 \in W^1$, $r \in R$, $q \in Q$

Note that the following pairs of P_4 are complementary: type (1) and (2), type (3) and (7), type (4) and (8), type (5) and (9), type (6) and (12), type (10) and (11).

If a split module $W = W_1 + W_2$ is not a strict split module of G or $\overline{G}$, then substituting two marker vertices for W^1 and W^2 does not satisfy the desired property regarding the P_4s: There are vertices $q \in Q$ adjacent to some $r_1 \in R$ and vertices $p \in P$ nonadjacent to some $r_2 \in R$, thus for every P_4 $abcd$ in G_W either qr_1ba or r_2cpa has no corresponding P_4 in $G_{V-W+w_1+w_2}$.

To ensure that every P_4 of G has a corresponding P_4 in G_W or in the graph after the substitution, we replace W with a marker P_4. Figure 4.4 illustrates this substitution. The prime nonsplit graph of Figure 4.4(a) has a split-homogeneous set $A = \{c_1, c_2, c_3\} + \{b_1, b_2, b_3\}$, which is replaced with the marker P_4 $b_1c_1c_3b_3$ in Figure 4.4(b).

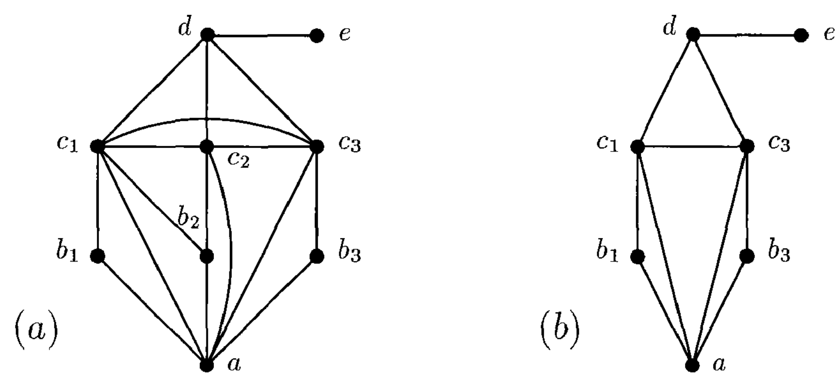


Figure 4.4: *The substitution of a marker P_4 for a split-homogeneous set.*

The substitution of maker P_4s was proposed by BABEL AND OLARIU in [5] whereas the substitution of two marker vertices for strict split-homogeneous sets was given by RASCHLE AND SIMON in [67]. Consequently, BABEL AND OLARIU do not perform the substitution shown in Figure 4.3 and RASCHLE AND SIMON fail to substitute the split-homogeneous set of Figure 4.4. Of course, both approaches can be combined in a natural way by substituting marker P_4s only if the split-homogeneous sets are neither strict in the graph nor strict in the complement, otherwise we use two marker vertices as described before.

4.4 The combined decomposition

In this section, we show that the decompositions of the previous two sections can be combined. To begin with, note that the bipartite-modular decomposition can also be applied to the complement of a graph. We call the complement of bipartite modules and bipartite-homogeneous sets *cobipartite modules* and *cobipartite-homogeneous sets*, respectively.

Lemma 4.4.1 *Let $A = A^1 + A^2$ be a bipartite module and $B = B^1 + B^2$ be a cobipartite module of a prime graph. Then $A \cap B = \emptyset$.*

Proof. First, we show that $|B^1 \cap A| \leq 1$. Suppose the contrary. Then B^1 consists of two adjacent vertices $a \in A^1$ and $b \in A^2$. Since A is not a 1-module, an A-partial vertex c exists. Without loss of generality, assume that c is A^1-universal and A^2-null. Because c is B^1-partial, c belongs to B^2.

Since a is A^2-partial, there is a vertex $d \in A^2$ which misses a. Moreover, $d \notin B^2$ because $c \in B^2$ misses d, hence $d \notin B$, thus d is B-null. On the other hand, there is a vertex $e \in B^2$ which sees b and misses a. If $e \notin A$, then e sees d, a contradiction as d is B-null. So $e \in A^1$. Since b is A^1-partial, there is a vertex $f \in A^1$ which misses b. Now f sees c and misses e, hence f is B^2-partial and it must belong to B. But this is impossible for f misses $e \in B^2$ and misses $b \in B^1$.

So far, we have show that $|B^1 \cap A| \leq 1$. By symmetry, we also know that $|B^2 \cap A| \leq 1$. Now suppose that $|B^1 \cap A| = 1$. Without loss of generality, assume that $b \in B^1 \cap A^1$. Since $|A^1| \geq 2$ and $|B^1| \geq 2$, vertices $a \in A^1 - B^1$ and $c \in B^1 - A$ exist. Furthermore, c is A^1-universal, thus a sees c and misses b, hence a is B^1-partial, thus $a \in B^2$. By our assumption $A^2 \cap B = \emptyset$. Since b is A^2-partial, there are vertices $d, e \in A^2$ such that b sees d and misses e. Therefore d is B^1-universal and e is B^1-null, a contradiction because $c \in B^1$ is not A^2-partial.

So $|B^1 \cap A| = \emptyset$ and, by symmetry, $B^2 \cap A = \emptyset$, which proves our lemma. □

Lemma 4.4.2 *Let $A = A^1 + A^2$ be a bipartite module and $B = B^1 + B^2$ a split module of a prime graph. Then $A \cap B = \emptyset$.*

Proof. In a fist step, we show that the assumption $A \cap B \neq \emptyset$ and $A \cap B^2 = \emptyset$ leads to a contradiction. Without loss of generality, let

$a \in A^1 \cap B^1$. Since a is B^2-partial, there are vertices $b, c \in B^2$ such that a sees b and misses c. Hence b is A^1-universal and c is A^1-null, thus every vertex in A^1 is B^2-partial, therefore $A^1 \subseteq B^1$. This is a contradiction because A^1 is a stable set consisting of at least two vertices whereas B^1 is a clique.

To show our lemma, it remains to prove that the assumption $A \cap B^2 \neq \emptyset$ also leads to a contradiction. Without loss of generality, let $b \in A^1 \cap B^2$. Since b is A^2-partial, there is a vertex $a \in A^2$ that sees b. Then $a \notin B^2$ because B^2 is stable.

Case 1: $a \notin B$. Then a is B-universal. Since a is A^1-partial, there is a vertex $c \in A^1$ that misses a. Moreover, $c \notin B$. Since b is B^1-partial, there are vertices $d, e \in B^1$ such that b sees d and misses e. Now a sees d and e, hence $d, e \notin A^2$, thus $d \notin A$. But d sees c, so c is B^1-universal, hence $e \notin A$. But this is a contradiction because e is A^1-partial.

Case 2: $a \in B$. Then $a \in B^1$.

Case 2.1: $B^1 \not\subseteq A$. Let c be a vertex in $B^1 - A$. Then c is A^2-universal. Since a is A^1-partial, there is a vertex $d \in A^1$ that misses a. Furthermore, $d \notin B^1$ and, since d is A^2-partial, a vertex $e \in A^2$ exists which sees d. Now e misses a and sees c, thus e is B^1-partial and therefore $e \in B^2$. So d is B^2-partial, a contradiction as d does not belong to B^1.

Case 2.2: $B^1 \subseteq A$. Then B^1 consists of $a \in A^2$ and another vertex $c \in A^1$, thus every vertex in B^2 distinguishes between a and c. Since both types of vertices in B^2 constitute modules and our graph is prime, B^2 consists of $b \in A^1$ and another vertex $d \in A^2$, i.e. the graph induced by B is the P_4 $bacd$. If this P_4 constituted the whole graph, then A would not be a bipartite module. So we may assume that a B-partial vertex e exists. Then e sees c and misses b, hence e is A^1-partial and therefore $e \in A^2$. But this is a contradiction because e sees $a \in A^2$ and A^2 is a stable set. □

A split module is a split module in the complement, thus the following corollary holds.

Corollary 4.4.3 *Let $A = A^1 + A^2$ be a cobipartite module and $B = B^1 + B^2$ a split module of a prime graph. Then $A \cap B = \emptyset$.*

The above results imply the uniqueness of the combined decomposition given in Algorithm 4.3

buildExtModTree(G)

input: a graph $G = (V, E)$
output: the root of the extended modular decomposition tree of G

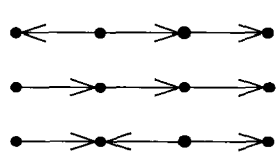
if $|V| = 1$ **then**
  let v be the vertex in V;
  return an empty node labeled v;
elsif G is disconnected **then**
  let $G_1, G_2, \dots, G_t$ be the connected components of G;
  let r_i = buildExtModTree(G_i) for $i = 1, \dots, t$;
  return a 0-node with children $r_1, r_2, \dots, r_t$
elsif $\overline{G}$ is disconnected **then**
  let $\overline{G}_1, \overline{G}_2, \dots, \overline{G}_t$ be the connected components of $\overline{G}$;
  let r_i = buildExtModTree(G_i) for $i = 1, \dots, t$;
  return a 1-node with children $r_1, r_2, \dots, r_t$
else (* G and $\overline{G}$ are connected and $|V| > 1$ *)
  let $G' = (V', E')$ be the characteristic graph of G;
  if G' is bipartite, split or cobipartite **then**
    let $H_1, \dots, H_t$ be the maximal proper modules of G;
    let r_i = buildExtModTree(G_{H_i}) for $i = 1, \dots, t$;
    return a 2-node with children $r_1, \dots, r_t$
  else (* G' is not bipartite, split or cobipartite *)
    let $B_1, \cdots, B_{k_B}$ be the vertex sets of G that correspond
      to maximal bipartite-homogeneous sets of G';
    let b_i = buildExtModTree(G_{B_i}) for $i = 1, \dots, k_B$;
    let $C_1, \cdots, C_{k_C}$ be the vertex sets of G that correspond
      to maximal cobipartite-homogeneous sets of G';
    let c_i = buildExtModTree(G_{C_i}) for $i = 1, \dots, k_C$;
    let $S_1, \cdots, S_{k_S}$ be the vertex sets of G that correspond
      to maximal split-homogeneous sets of G';
    let s_i = buildExtModTree(G_{S_i}) for $i = 1, \dots, k_S$;
    let $H_1, \dots, H_t$ be those maximal proper modules of G
      which are not contained in $B_1, \dots, B_{k_B}$,
        $C_1, \dots, C_{k_C}$ and $S_1, \dots, S_{k_S}$;
    let r_i = buildExtModTree(G_{H_i}) for $i = 1, \dots, t$;
    return a 3-node with children $b_1, \dots, b_{k_B}$,
      $c_1, \dots, c_{k_C}$, $s_1, \dots, s_{k_S}$ and $r_1, \dots, r_t$;
  fi
fi

Algorithm 4.3

Chapter 5

P_4-comparability graphs

To obtain subclasses of perfectly orderable graphs that can be recognized in polynomial time, HOÀNG AND REED [40] suggested restricting the number of ways a P_4 may be oriented. Since a perfect order is obstruction-free, perfectly orderable graphs are precisely those graphs which admit an acyclic orientation such that every P_4 is oriented as one of the P_4s in Figure 5.1 (up to symmetry). Six classes of graphs are obtained by permitting any nonempty proper subset of these P_4s in an acyclic orientation.



type 1

type 2 (indifferent)

type 3 (transitive)

Figure 5.1: *All obstruction-free orientations of a P_4.*

If only P_4s of type 1 and 2 are permitted, the corresponding class of graphs is a subclass of brittle graphs, and brittle graphs can be recognized in $O(|E|^2)$ [72]. If only P_4s of type 1 and 3 are permitted, the recognition of the corresponding class of graphs is NP-complete [37]. The remaining graphs admit an acyclic orientation with P_4s of type 2 and 3. We call these graphs *wing-comparability* and the corresponding orientation *wing-transitive*. To date, it is not known whether wing-comparability graphs can be recognized in polynomial time.

P_3	P_4	
nontrivial P_3-class	**P_4-component**	**strong P_4-component**
Corollary 3.3.4	Corollary 5.1.11	
Theorem 3.3.3	Theorem 5.1.12	
Corollary 3.3.5	Corollary 5.1.13	
Theorem 3.3.6	Theorem 5.2.1	
Lemma 3.3.7	Lemma 5.2.6	Lemma 5.2.8
Theorem 3.3.8	Theorem 5.2.2	

Table 5.1: *Analogous results on the P_3- and P_4-structure.*

If only P_4s of type 3 are permitted, the corresponding graphs are called *P_4-comparability* and their orientation *P_4-transitive* (every P_4 is transitively oriented). In [40] and [39], HOÀNG AND REED presented an $O(|V|^4)$ algorithm to recognize P_4-comparability graphs and an $O(|V|^5)$ algorithm to compute a P_4-transitive orientation.

In [67], RASCHLE AND SIMON investigated the P_4-analog of P_3-classes and developed an $O(|V|^2 \cdot |E|)$ recognition and orientation algorithm for P_4-comparability graphs. Another relation between P_4s was studied by BABEL AND OLARIU [5]. In the next two sections, we extend both RASCHLE AND SIMON's and BABEL AND OLARIU's results by conducting a rigorous study of the P_4-structure. As it turns out, most properties of the P_3-structure translate smoothly into similar properties of the P_4-structure. An overview of the correspondence between those results is given in Table 5.1. We also prove a stronger version of a theorem by CHVÀTAL [12] on P_4-chains.

In Section 5.3, we analyze the P_4-structure of split graphs and use the obtained results to decompose prime split graphs. In Section 5.4, we give an $O(|V|^4)$ algorithm to compute the split-modular decomposition and, in Section 5.5, two algorithms for recognizing and orienting P_4-comparability graphs are proposed. The first algorithm runs in $O(|E|^2)$ time and $O(|V| \cdot |E|)$ space and the other runs in $O(|V|^2 \cdot |E|)$ time and $O(|V| + |E|)$ space.

Finally, in the last section, we propose a new algorithm that uses the split-modular decomposition to recognize classes of perfectly orderable graphs. For instance, HERTZ' bipartable graphs can be recognized this way.

5.1 P_4-components

In analogy to the P_3-classes of Section 3.3, we define *P_4-classes* as the equivalence classes of the transitive closure of the P_4-adjacency relation where two edges are P_4-adjacent if they belong to the same P_4. In [40], HOÀNG AND REED proved the following analog of Theorem 3.3.2.

Theorem 5.1.1 (HOÀNG AND REED) *A graph is P_4-comparability if and only if each of its P_4-classes admits a P_4-transitive orientation.*

We prove the above theorem in Section 5.5. To obtain more general results, however, we investigate relations between P_4s rather than relations between the edges in P_4s. The following relations between P_4s are considered. (Note that nontrivial P_4-classes correspond to the weak P_4-components defined below.)

Definition 5.1.2 *Two P_4s are*
(1) weak-adjacent *if they have a common edge, and*
(2) adjacent *if two wings or a rib and a wing coincide, and*
(3) strong-adjacent *if they have three common vertices.*
The equivalence classes of the transitive closure of the above (weak, strong) adjacency relation are called (weak, strong) P_4-components.

In the rest of this chapter, C^* stands for a P_4-component and D^* for a strong P_4-component. Furthermore, we use F^* to indicate that a statement holds for P_4-components and strong P_4-components. The *cover* of a (strong) P_4-component F^*, denoted by $V(F^*)$, is the set of vertices which belong to some P_4s in F^*. Similarly, $E(F^*)$ denotes the set of edges which belong to some P_4s in F^*. Given a P_4 $abcd$, we write $F^*(abcd)$ for the (strong) P_4-component that contains $abcd$. Furthermore, we write $abcd \sim a'b'c'd'$ if the P_4s $abcd$ and $a'b'c'd'$ are strong-adjacent.

Consider again the relations between P_4s given in Definition 5.1.2. Clearly two adjacent P_4s are also weak-adjacent. We claim that two strong-adjacent P_4s are also adjacent. To prove this claim, we examine the graphs induced by a P_4 $abcd$ and a fifth vertex v. Up to symmetry, all possibilities are enumerated in Figure 5.2 (bold lines indicate edges in P_4s). Now it is easy to infer that every strong-adjacent P_4 is one of the types given in Table 5.2 (up to symmetry), thus strong-adjacent P_4s are indeed adjacent.

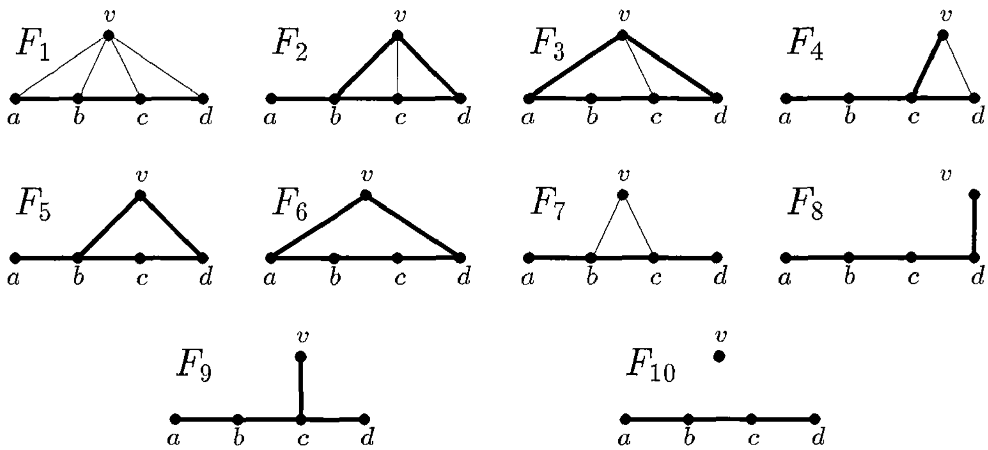


Figure 5.2: *All possibilities of a P_4 together with a fifth vertex v.*

type	strong-adjacent P_4	graph in Figure 5.2
(a)	$abvd$	F_2, F_5
(b)	$abcv$	F_4, F_9
(c)	$bcdv$	F_6, F_8
(d)	$bavd$	F_6, F_3

Table 5.2: *All types of P_4s strong-adjacent to a P_4 abcd.*

The converse, however, does not hold: Weak adjacent P_4s need not be adjacent and adjacent P_4s need not be strong-adjacent. A weak form of the converse are the following lemmas.

Lemma 5.1.3 *Two different P_4s with a common rib are connected by a sequence of strong-adjacent P_4s.*

Proof. Let $abcd$ and $a'bcd'$ denote the two P_4s with common ribs. If $abcd$ and $a'bcd'$ are not strong-adjacent, then $|\{a, a', b, c, d, d'\}| = 6$.

If a misses d', then $abcd \sim abcd' \sim a'bcd'$. The analogous argument applies if a' misses d, so it remains to discuss the case $ad', a'd \in E$.

If a sees a', then $abcd \sim aa'dc \sim a'ad'c \sim a'bcd'$. Otherwise, if a misses a', we find that $abcd \sim aba'd \sim a'bad' \sim a'bcd'$. □

Lemma 5.1.4 *If the rib of a P_4 is the wing of another P_4, then those P_4s are connected by a sequence of strong-adjacent P_4s.*

Proof. Because of symmetry, we may assume that $abcd$ and $bcef$ denote the two P_4s. If $abcd$ and $bcef$ are not strong-adjacent, then $|\{a, b, c, d, e, f\}| = 6$.

If a misses e, then $abcd \sim abce \sim bcef$. Similarly, if a sees f, then $abcd \sim fabc \sim bcef$. So suppose that a sees e but misses f.

If d sees e, then $abcd \sim baed \sim baef \sim bcef$. If d misses e and d misses f, then $abcd \sim aecd \sim fecd \sim fecb$. Otherwise, if d misses e and d sees f, then $abcd \sim bcdf \sim bcef$. □

Two weak-adjacent P_4s that are not adjacent have a common rib, hence it follows from Lemma 5.1.3 that

Corollary 5.1.5 *The P_4-components and the weak P_4-components are identical.*

The above corollary implies that P_4-components correspond to non-trivial P_4-classes. The P_4s to which an edge vw belongs are therefore contained in the same P_4-component, that is, for every edge vw, there is at most one P_4-component C^* with $vw \in E(C^*)$. For this reason, we do not always distinguish between C^* and $E(C^*)$. So we write $vw \in C^*$ instead of $vw \in E(C^*)$ and $C^*(vw)$ for the P_4-component that contains the edge vw.

Regarding P_4-components and strong P_4-components, a result similar to Corollary 5.1.5 is impossible because the net of Figure 4.2 is a counterexample: It has only one P_4-component but consists of three strong P_4-components. The next lemma shows that, in some sense, the net is the only exception.

Lemma 5.1.6 *If two adjacent P_4s do not belong to the same strong P_4-component, then these two P_4s induce a net.*

Proof. Let $abcd$ and $a'b'c'd'$ be those two adjacent P_4s. Since they are adjacent but in different strong P_4-components, we may assume that $|\{a, b, c, d, a', b', c', d'\}| = 6$. Furthermore, by Lemma 5.1.3 and Lemma 5.1.4, those P_4s have a common wing. Without loss of generality, let $ab = a'b'$, thus either $a' = a$ and $b' = b$ or $a' = b$ and $b' = a$.

Case 1: $a' = b$ and $b' = a$. We show that $abcd$ and $bac'd' = a'b'c'd'$ belong to the same strong P_4-component, hence this case is impossible. If c misses c', then $abcd \sim c'abc \sim d'c'ab$. So suppose that c sees c'.

If c sees d', then $abcd \sim abcd' \sim bac'd'$ is a sequence of strong P_4-components. If c misses d', then $bac'd' \sim bcc'd'$. Furthermore Lemma 5.1.4 implies that $bcc'd$ and $abcd$ belong to the same strong P_4-component, so we are done.

Case 2: $a' = a$ and $b' = b$. If d sees c', then $abcd \sim abc'd \sim abc'd'$, a contradiction. So we may assume that d misses c' and, because of symmetry, that d' misses c.

If c misses c', then $abcd \sim c'bcd$ and Lemma 5.1.4 applies to $c'bcd$ and $abc'd'$, hence $abcd$ and $abc'd'$ are in the same strong P_4-component, a contradiction. Therefore c sees c'.

Finally, if d sees d', then $abcd \sim bcdd' \sim bc'd'd \sim abc'd'$, again a contradiction. So d misses d' and the induced subgraph is a net as claimed. □

In the rest of this section, we relate the cover of P_4-components and strong P_4-components to strict split modules and split modules. For that purpose, we need the notion of separable (strong) P_4-components.

Definition 5.1.7 *A (strong) P_4-component F^* is separable if its cover $V(F^*)$ can be partitioned into vertex sets $V^1 + V^2$ such that every P_4 in F^* has its midpoints in V^1 and its endpoints in V^2.*

The following lemma exhibits the fundamental structure of separable (strong) P_4-components.

Lemma 5.1.8 *Given a separable (strong) P_4-component with vertex partition $V^1 + V^2$. Then neither a P_3 abc with $a \in V^1$ and $b, c \in V^2$ nor a $\overline{P}_3$ abc with $a, b \in V^1$ and $c \in V^2$ exists.*

Proof. Let F^* stand for the (strong) P_4-component. In a first step, we show that no P_3 or $\overline{P}_3$ as described in our lemma has edges in $E(F^*)$. Assume a P_3 abc with $a \in V^1$ and $b, c \in V^2$. Since F^* is separable, bc cannot belong to $E(F^*)$, so suppose that $ab \in E(F^*)$. Then a P_4 $bade$ in F^* exists with $d \in V_1$ and $e \in V^2$. If $ce \in E$, then $bade \sim abce$ and $abce$ contradicts the separability of F^*. Hence $ce \notin E$. But $dc \in E$ implies *bade* $\sim$ *bcde*, and $dc \notin E$ implies *bade* $\sim$ *dabc*. This is a contradiction because the P_4s *bcde* and *dabc* violate the separability of F^*.

Now assume a $\overline{P}_3$ with $a, b \in V^1$, $c \in V^2$ and a P_4 *cade* exists in F^*. Then $d \in V^1$ and $e \in V^2$. If b sees d, then *cade* $\sim$ *cadb* and *cadb* violates the separability of F^*; hence b misses d. If b sees e, then *cade* $\sim$ *adeb* and *adeb* would violate the separability of F^*; thus b misses e. In fact, we have shown that if b misses the vertices incident to one wing of a P_4 in F^*, then the same holds for the vertices incident to the other wing. But Corollary 5.1.5 and the separability of F^* imply that (strong-) adjacent P_4s in F^* have a common wing. So by induction on the P_4s in F^*, no wing is incident to b, a contradiction to our assumption that b belongs to the cover of F^*.

The remainder of the proof is based on what we have already shown, namely that an edge in a P_3 or a $\overline{P}_3$ as defined in our lemma does not belong to a P_4 in F^*. We call those P_3 and $\overline{P}_3$ *forcing* because every P_4 with an edges in such a P_3 or $\overline{P}_3$ is forced out of F^*. Next, we show that no forcing $\overline{P}_3$ *abc* can exist.

Since F^* covers b, there is a P_4 *dbef* in F^* and therefore $d \in V^2$. If $cd \in E$, then *bdc* is a forcing P_3, and if $ad \notin E$, then *bad* is a forcing $\overline{P}_3$; in both cases a contradiction to $bdef \in F^*$. Therefore $cd \notin E$ and $ad \in E$; thus *cadb* is a P_4. Since F^* is separable, $cadb \notin F^*$. Moreover *cadb* and *dbef* are adjacent but do not induce a net, hence *cadb* $\sim$ *dbef*, thus *cadb* contradicts the separability of F^*.

It remains to prove that no forcing P_3 *abc* exists. Since F^* covers c, there is a P_4 $cdef \in F^*$, hence $d \in V^1$. Moreover $bd \in E$, for otherwise the forcing P_3 *dcb* would contradict $dcef \in F^*$. We say that an edge $vw \in F^*$ with $v \in V^2$ and $w \in V^1$ is

type1 if b sees v and a forcing P_3 *wbu* exists, and

type2 if b sees w and a forcing P_3 *ubv* exists.

Figure 5.3 illustrates this definition. (Solid lines indicate edges that must exist whereas dotted lines indicate edges that must not exist.)

Obviously *cd* is type2. We claim that every wing of a P_4 in F^* is either type1 or type2. From this follows immediately that F^* cannot cover b, a contradiction to our assumption.

The proof of the above claim is by induction on the P_4s in F^*. Since *cd* is type2, we have already settled the basis. For the inductive step, it suffices to show that one wing in a P_4 in F^* is type1 or type2 on the

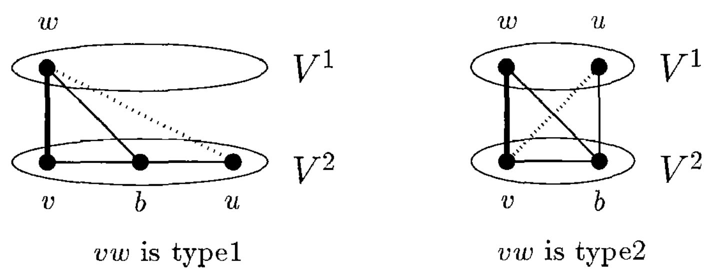


Figure 5.3: *A type1 and type2 edge as defined in the proof of Lemma 5.1.8.*

assumption that this already holds for the other wing in the same P_4. So let $vwxy$ denote an arbitrary P_4 in F^* and assume that vw is type1 or type2.

Case 1: vw is type1. Then v misses u, for otherwise the forcing P_3 wvu would contradict $vw \in F^*$. We distinguish the following two subcases.

Case 1.1: $u = y$. If b misses x, then xyb is a forcing P_3, a contradiction to $vwxy \in F^*$. Therefore b sees x; thus b sees y and xbv is a forcing P_3, i.e. xy is type1.

Case 1.2: $u \neq y$. Then $|\{b, u, v, w, x, y\}| = 6$. Furthermore, both $bx \notin E$ and $by \notin E$ cannot hold, as otherwise $vwxy \sim bwxy$ but bw cannot belong to a P_4 in F^*. If $bx \notin E$ and $by \in E$, then xyb is a forcing P_3, a contradiction to $bwxy \in F^*$. If $bx \in E$ and $by \notin E$, then $vwxy \sim vbxy$, a contradiction because $vbxy$ violates the separability of F^*. Therefore $bx \in E$ and $by \in E$ holds; thus b sees x and wby is a forcing P_3, i.e. xy is type2.

Case 2: vw is type2. Then u sees w, for otherwise the forcing $\overline{P}_3$ wuv would contradict $vwxy \in F^*$. Again we distinguish two subcases.

Case 2.1: $x = u$. If b misses y, then $vwxy \sim vbxy$ and $vbxy$ contradicts the separability of F^*. Therefore b sees y and xbv is a forcing P_3; thus xy is type1.

Case 2.2: $x \neq u$. Then $|\{b, u, v, w, x, y\}| = 6$. Assume that b misses x. Then b misses y as well, for otherwise the forcing P_3 xyb would contradict $vbxy \in F^*$. If u misses y, then either $vwxy \sim uwxy$ and $uwxy$ contradicts the separability of F^* or $buxy \sim bwxy \sim vwxy$, a contradiction to $buxy \notin F^*$ because of the forcing P_4 buv. So u sees

y and $vbuy \sim vwuy \sim vwxy$, a contradiction as $vbuy \notin F^*$ because of the forcing P_4 ubv.

Therefore our assumption was wrong; so b sees x. Moreover b sees y, as otherwise $vwxy \sim vbxy$ and $vbxy$ would violate the separability of F^*. Thus b sees x and wby is a forcing P_3, i.e. xy is type2. □

Suppose that a vertex v is not covered by a (strong) P_4-component $F^*(abcd)$. Then the only possible graphs are the F_1, the F_7 and the F_{10} of Figure 5.2, i.e. v is either $\{a,b,c,d\}$-universal, $\{a,b,c,d\}$-null or it sees the midpoints but misses the endpoints of the P_4 $abcd$. We use this observation to proof the next lemma.

Lemma 5.1.9 *Let F^* be a (strong) P_4-component and v a vertex not covered by F^*. If v and a P_4 in F^* induces an F_7, then the graph induced by v and any P_4 in F^* is an F_7.*

Proof. Our proof is by induction on the P_4s in a (strong) P_4-component F^*. For the inductive step, we show that a P_4 $a'b'c'd'$ together with v induces an F_7 on the assumption that an adjacent P_4 $abcd$ together with v induces an F_7. We distinguish the following cases:

Case 1: Two wings coincide. Without loss of generality, we may assume that the wing ab coincides with the wing $a'b'$; thus either $a' = a$ and $b' = b$ or $a' = b$ and $b' = a$. The latter, however, is impossible because $a'b'c'd'$ and v would not induce an F_1, F_7 or F_{10}. In the former case, the only possible induced graph is the F_7 as claimed.

Case 2: A wing coincides with a rib. A wing of $abcd$ cannot coincide with $b'c'$ as otherwise the graph induced by $a'b'c'd'$ and v would not be an F_1, F_7 or F_{10}. Therefore, a wing of $a'b'c'd'$ must coincide with bc. This implies that the graph induced by $a'b'c'd'$ and v is an F_1; thus $|\{a,d,a',b',c',d'\}| = 6$.

Without loss of generality (symmetry), let $b = a'$ and $c = b'$. Then d' sees a and d, for otherwise $abvd'$ or $dcvd'$ would be a P_4 in F^* that covers v. So $ad'dc$ is a P_4 in F^*, a contradiction because $ad'dc$ and v induce an F_5. □

If a $V(F^*)$-partial vertex r exists, then there is a P_4 $abcd$ in F^* such that r is $\{a,b,c,d\}$-partial, hence r together with $abcd$ induces an F_7. By Lemma 5.1.9, the vertex r sees the midpoints of every P_4 in F^* and misses its endpoints; thus F^* is separable.

Furthermore, if C^* is a P_4-component, then r cannot be adjacent to a $V(C^*)$-null vertex q, as otherwise every P_4 $abcd$ in F^* would imply a P_4 $qrba$ in C^*, a contradiction to our assumption that r is not covered by F^*. The next corollary summarizes our findings.

Corollary 5.1.10 *Let F^* be a (strong) P_4-component whose cover is not a module. Then F^* is separable and every $V(F^*)$-partial vertex is V^1-universal and V^2-null. Moreover, if C^* is a P_4-component, then no edge between a $V(C^*)$-partial and a $V(C^*)$-null vertex exists.*

Next, we investigate the relation between separable (strong) P_4-components and modules. Let F^* denote a separable (strong) P_4-component and consider an edge vw with both endpoints in V^2. From Lemma 5.1.8, it follows that no vertex in V^1 is $\{v, w\}$-partial, hence v and w have the same neighborhood relative to $V - V^2$. By induction, this holds for every pair of vertices in the same connected component of G_{V^2}, thus a connected component of G_{V^2} is a module. Since the analogous argumentation applies to V^1 and $\overline{G}$, we have the following analog of Corollary 3.3.4.

Corollary 5.1.11 *In a prime graph, the cover of a separable P_4-component is a strict split module and the cover of a separable strong P_4-component is a split module.*

Recall that every P_4 not contained in a strict split-homogeneous set $W = W^1 + W^2$ has a corresponding P_4 in the graph after the substitution of two nonadjacent marker vertices for W^1 and W^2. But such a P_4 either has all its vertices in $V - W$ or is of type (1) to (6) listed on Page 52. In each case, this P_4 is not W-partial, thus the P_4-analog of Theorem 3.3.3 holds.

Theorem 5.1.12 *Let C^* denote an arbitrary P_4-component. Then no $V(C^*)$-partial P_4 exists.*

Let W be a strict split module and $abcd$ a P_4 in G_W. If $W \subset V(C^*(ab))$, then a W-partial P_4 would exist, a contradiction to Theorem 5.1.12. Hence $V(C^*(ab)) \subseteq W$.

Similarly let W be a split module and D^* a strong P_4-component that contains a P_4 with all its vertices in W. If $W \subset V(D^*)$, then strong-adjacent P_4s $abcd \sim a'b'c'd'$ in D^* exist such that $\{a, b, c, d\} \subseteq W$ and

$\{a', b', c', d'\} \not\subseteq W$. But every possibility of Table 5.2 contradicts the definition of a split module, thus the analog of Corollary 3.3.4 holds.

Corollary 5.1.13 *Let W be a vertex set and $abcd$ be a P_4 of G_W. If W is a split module, then $V(D^*(abcd)) \subseteq W$. Similarly, if W is a strict split module, then $V(C^*(ab)) \subseteq W$.*

The above corollary together with Lemma 4.1.3 implies that every minimal strict split module is the cover of some P_4-component and that every minimal split module is the cover of some strong P_4-component.

5.2 GALLAI-type theorems

In this section, we prove the P_4-analogs of GALLAI's decomposition theorem. The key theorem is the following P_4-analog of Theorem 3.3.6. It states that (strong) P_4-components can be uniquely identified by their covers.

Theorem 5.2.1 *Two different (strong) P_4-components have different covers.*

The proof of the above theorem is rather lengthy, which is why we moved it to the end of this section. Given Theorem 5.2.1 holds, however, it is quite easy to show the following GALLAI-type theorem for (strong) P_4-components.

Theorem 5.2.2 *Let $G = (V, E)$ be a prime graph that is not split. Then the P_4s not contained in one of the maximal strict split-homogeneous sets of G constitute a P_4-component that covers G, and the P_4s not contained in one of the maximal split-homogeneous sets of G constitute a strong P_4-component that covers G.*

Proof. If no strong P_4-component covers G, then, by Corollary 5.1.10 and Corollary 5.1.11, the cover of every strong P_4-component induces a split graph. Since G is prime nonsplit, Theorem 3.4.4 implies a C_5, P_5, $\overline{P}_5$, F_2 or $\overline{F}_2$, where the F_2 and $\overline{F}_2$ refer to the graphs in Figure 3.1. But each of those graphs contains two strong-adjacent P_4s that induce a C_4, C_5 or $2K_2$, hence the corresponding strong P_4-component does not induce a split graph, a contradiction.

So we know that a strong P_4-component, say D^*, covers G. Let $abcd$ be a P_4 not contained in one of the maximal split-homogeneous sets of G. Then $D^*(abcd)$ covers the whole graph, for otherwise $V(D^*(abcd))$ would be split-homogeneous and therefore be contained in a maximal split-homogeneous set, a contradiction to our assumption. But Theorem 5.2.1 implies that $D^* = D^*(abcd)$, thus we have shown the second part of our theorem.

To prove the first part, let C^* be the P_4-component that contains all P_4s in D^*. Then C^* covers G and the above argumentation remains valid if we replace strong P_4-components and split-homogeneous sets with P_4-components and strict split-homogeneous sets. □

Corollary 5.1.10 together with Corollary 5.1.11 implies that the cover of a strong P_4-component that does not cover the whole graph is either homogeneous or split-homogeneous in the characteristic graph. Therefore every strong P_4-component in a graph without homogeneous and split-homogeneous sets covers the whole graph. By Theorem 5.2.1, there is at most one such component, thus we have

Corollary 5.2.3 *If a graph G has neither homogeneous sets nor split-homogeneous sets, then every P_4 in G belongs to the same strong P_4-component.*

A *star-cutset* of a graph $G = (V, E)$ is a vertex set S such that G_{V-S} is disconnected and G_S contains a dominating vertex. In [12], CHVÀTAL showed that if neither G nor its complement has a star-cutset, then every two P_4s are "3-chained", that is, every two P_4s belong to the same strong P_4-component. We claim that the above corollary is a stronger version of CHVÀTAL's theorem. We do this by proving that a graph with homogeneous or split-homogeneous sets has a star-cutset in the graph or its complement but not vice versa. The latter is easy as the P_5 is an example of a graph that has star-cutset but has neither homogeneous nor split-homogeneous set.

Now suppose that a graph $G = (V, E)$ has a homogeneous set H. If there are H-null vertices, then $(N(h) \cap V - H) + h$ is a star-cutset for every vertex $h \in H$. If no H-null vertices exist, then $\overline{G} = \overline{G}_H + \overline{G}_{V-H}$, hence every vertex $h \in H$ is a star-cutset of $\overline{G}$. Next, suppose that $G = (V, E)$ has a split-homogeneous set $W = W^1 + W^2$. Then $S = W^1 \cup R \cup P$ is a star-cutset as every vertex in W^1 is dominating in G_S and $G_{Q \cup W^2}$ is disconnected (even if $Q = \emptyset$).

In the remainder of this section, we prove Theorem 5.2.1. In a first step, we show the theorem for P_4-components. The following lemmas prepare this part of the proof.

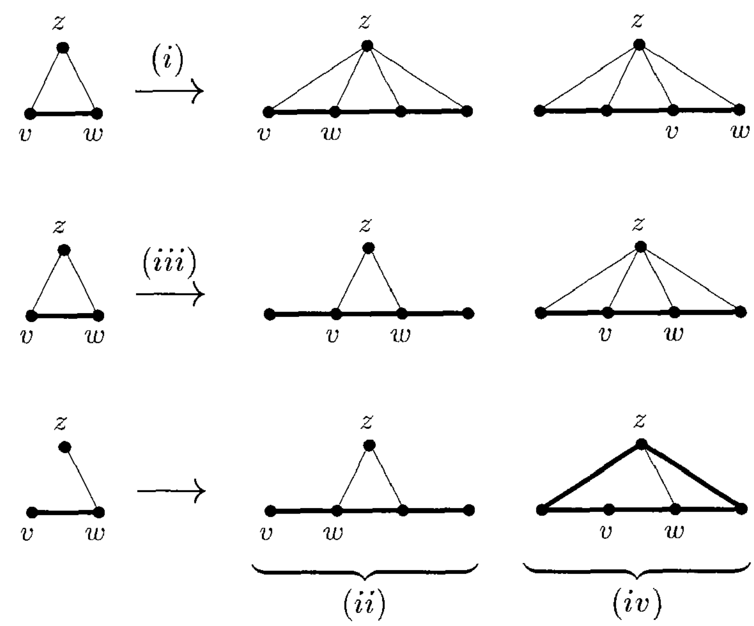


Figure 5.4: *Lemma 5.2.4 illustrated.*

Lemma 5.2.4 *Let vw be an edge of a P_4 and z a vertex different from v and w.*

(i) If vw is a wing and $vz, wz \in E - C^*(vw)$, then z sees all the vertices in the P_4.

(ii) If vw is a wing, z misses v and $wz \in E - C^*(vw)$, then the P_4 can be labeled $vwxy$ and z sees x but misses y.

(iii) If vw is a rib and $vz, wz \in E - C^*(vw)$, then the P_4 can be labeled $uvwx$ and either z misses u and x or z sees u and x.

(iv) If vw is a rib, z misses v and $wz \in E - C^*(vw)$, then P_4 can be labeled $uvwx$ and $uz, xz \in C^*(vw)$.

Proof. *(i)* Without loss of generality, let $vwxy$ be the P_4 in question. From Figure 5.2 follows that only the F_1 is possible.

(ii) The P_4 can be labeled $xyvw$ or $vwxy$. Again from Figure 5.2 follows that the former case is impossible whereas in the latter case only an F_7 does not contradict $wz \in E - C^*(vw)$.

(iii) A P_4 $xvwy$ implies an F_1, F_2 or F_7. But an F_2 cannot satisfy both $vz \notin C^*(vw)$ and $wz \notin C^*(vw)$.

(iv) In this case, only the F_3 does not contradict $wz \in E - C^*(vw)$, see Figure 5.2. □

Lemma 5.2.5 *Let vw be a rib of a P_4 and z a vertex that sees w but misses v. If $|C^*(wz)| > 1$, then $C^*(wz) = C^*(vw)$.*

Proof. Suppose the contrary $C^*(wz) \neq C^*(vw)$. From Lemma 5.2.4(iv) follows that the P_4 in which vw is the rib can be labeled $uvwx$ with $uz, xz \in C^*(vw)$. Moreover, as $|C^*(wz)| > 1$, the edge wz belongs to a P_4 as well.

Case 1: wz is a wing. Then Lemma 5.2.4(ii) applies to wz and u; hence the P_4 with the wing wz can be labeled $wzab$. The same lemma also applies to zw and v; therefore the same P_4 can be labeled $zwde$. But no P_4 can be labeled in both ways.

Case 2: wz is a rib. Then Lemma 5.2.4(iv) applied to wz and u and zw and v respectively guarantees a P_4 $awzb$ with $ua, ub, va, vb \in C^*(wz)$. Thus either $bvwx$ or $ubxw$ is a P_4; in both cases a contradiction to $C^*(wz) \neq C^*(vw)$. □

The next lemma deals with the pyramid, see Figure 4.2. It is the analog of Lemma 3.3.7 for P_4-components.

Lemma 5.2.6 *Let $abcdrp$ be a pyramid. If $C^*(ab)$ is different from $C^*(rb)$ and $C^*(rc)$, then r and p are not covered by $C^*(ab)$.*

Proof. If $\{ab, bc, cd\} = C^*(ab)$, there is nothing to prove. Therefore, assume a P_4 $a'b'c'd'$ weak-adjacent to $abcd$. Note that the P_4s $rbpd$ and $rcpa$ guarantee that all edges in the pyramid different from ab, bc and cd do not belong to $C^*(ab)$.

In the following case analysis, we show that $a'b'c'd'pr$ is another pyramid which satisfies $C^*(rb') \neq C^*(ab)$ and $C^*(rc') \neq C^*(ab)$. By induction, this holds for every P_4 in $C^*(ab)$; thus r is incident to no edge in $C^*(ab)$ as claimed.

Case 1: A wing of $abcd$ coincides with a wing of $a'b'c'd'$. Without loss of generality, let $a'b'$ be the common edge. Then Lemma 5.2.4(ii) applies to $a'b'$ and r; hence $a' = a$, $b' = b$ and r sees c' but misses d'; thus $C^*(rb') = C^*(rb) \neq C^*(ab)$. Similarly, Lemma 5.2.4(i) applies to $a'b'$ and p; hence p sees c' and d'; thus $a'b'c'd'rp$ is a pyramid. Moreover $C^*(rc') = C^*(rc) \neq C^*(ab)$ because of the P_4s $rcpa$ and $rc'pa$.

Case 2: A wing of abcd coincides with the rib of $a'b'c'd'$. Then Lemma 5.2.5 applies to $b'c'$ and r; thus $C^*(ab) = C^*(rb)$ or $C^*(ab) = C^*(rc)$, a contradiction to the premise of our lemma.

Case 3: The rib of abcd coincides with a wing of $a'b'c'd'$. Without loss of generality, let $a' = b$ and $b' = c$. From Lemma 5.2.4(i) applied to $a'b'$ and r follows that r sees c' and d'. But the same Lemma also applies to $a'b'$ and p; so p sees c' and d'. Thus $|\{a', b', c', d', d, r, p\}| = 7$. Furthermore d sees d', as otherwise the P_4 $dcrd'$ would contradict $C^*(ab) \neq C^*(rc)$. So $bcdd'$ and $dd'rb$ are P_4s; hence $C^*(ab) = C^*(rb)$, a contradiction to our assumption. □

Corollary 5.2.7 *Let abcdrp denote a pyramid. Then $V(C^*(rb)) = V(C^*(ab))$ implies $C^*(rb) = C^*(ab)$.*

Proof. Suppose $V(C^*(rb)) = V(C^*(ab))$ and $C^*(rb) \neq C^*(ab)$. Then $C^*(rc) = C^*(ab)$, as otherwise a contradiction to Lemma 5.2.6 would arise. Therefore $C^*(ab) = C^*(rc)$ is different from $C^*(rb)$, thus Lemma 5.2.6 applies to the pyramid $rbpdac$; hence a cannot be covered by $C^*(rb)$, a contradiction to our assumption. □

Proof of Theorem 5.2.1 for P_4-components. Suppose the contrary, i.e. two different P_4-components C_1^* and C_2^* satisfy $V(C_1^*) = V(C_2^*)$. Then C_1^* (and C_2^*) cannot be trivial and a P_4 $abcd$ in C_1^* exists. Clearly, each vertex in $\{a, b, c, d\}$ is incident to at least one edge in C_2^*. Therefore, the vertices $\{a, b, c, d\}$ together with the other endpoint of such an edge, say v, induce one of the graphs depicted in Figure 5.2. Moreover $C_1^* \neq C_2^*$, which leaves the graphs F_1, F_2, F_3, F_4 and F_7. We show that each of these graphs is impossible.

***F_3*:** Then $vc \in C_2^*$ and Lemma 5.2.5 applies to bc and v; hence $C^*(bc) = C^*(vc)$, a contradiction to $C_1^* \neq C_2^*$.

***F_4*:** Then $vd \in C_2^*$. Since the situation is symmetric relative to v and d, we may assume that vw denotes another edge in a P_4 that contains vd. Hence dvw is a P_3 and $|\{a, b, c, d, v, w\}| = 6$.

Suppose w misses c. Then w sees b, as otherwise the P_4 $bcvw$ would imply $C_1^* = C_2^*$. Hence $bwvd$ is a P_4 in C_2^*, Lemma 5.2.5 applies to wv and c; thus $C^*(wv) = C^*(cv)$, a contradiction to $C_1^* \neq C_2^*$. Therefore our supposition was wrong, so w sees c.

Furthermore w misses a, for otherwise the P_4s $awvd$ and $awcd$ would imply $C_1^* = C_2^*$. The same contradiction arises if w sees b, this time because of the P_4 $abwv$. Hence $abcw$ is another P_4 in C_1^*.

Obviously, the same argumentation holds for the third edge of the P_4 and, by induction, for every edge in C_2^*. Therefore, no edge in $C^*(vd)$ is incident to a or b, a contradiction to our assumption that $V(C_1^*) = V(C_2^*)$.

***F_7*:** Without loss of generality, let vb be the edge in C_2^*. Then vb cannot be the rib of a P_4, as otherwise a contradiction to Lemma 5.2.5 applied to vb and a would arise. Therefore vb is a wing, Lemma 5.2.4(ii) applies to vb and a; thus our P_4 can be labeled $vbxy$ and a sees x but misses y. If $y = d$, then $axdc$ is a P_4 which contradicts $C_1^* \neq C_2^*$. Hence $|\{a, b, c, d, v, x, y\}| = 7$.

Case 1: $cx \notin E$. As xb is a rib, we can apply Lemma 5.2.5 to xb and c; hence $C_1^* = C_2^*$, the usual contradiction.

Case 2: $cx \in E$. If d sees x, then $abcdvx$ is a pyramid which satisfies $V(C^*(vb)) = V(C^*(ab))$, Corollary 5.2.7 applies and again $C_1^* = C_2^*$. The same contradiction arises if c sees y, this time because of the pyramid $vbxyac$ and $V(C^*(vb)) = V(C^*(ab))$. Therefore $dx, cy \notin E$. So $yxcv$ and $axcd$ are P_4s; hence $C^*(cd) = C^*(yx)$, again a contradiction to $C_1^* \neq C_2^*$.

***F_2*:** Then $vc \in C_2^*$. Without loss of generality (symmetry), let vx be another edge in a P_4 which vc belongs to. In the following case analysis, we show that $abvd$ together with x again induces an F_2, i.e. the structure repeats itself. Therefore, by induction, all edges in C_2^* together with a, b and d induce an F_2; thus a, b and d are not covered by C_2^*, a contradiction to $V(C_1^*) = V(C_2^*)$.

Case 1: x sees b and d. If x sees a, the P_4s $axdc$ and $axvc$ imply $C_1^* = C_2^*$, a contradiction. Therefore x misses a and the P_4 $abvd$ together with x induces an F_2 as claimed.

Case 2: x misses b or d. If x misses b, Lemma 5.2.5 applies to bv and x, a contradiction to $C_1^* \neq C_2^*$. Hence x sees b but misses d. Then cv cannot be the wing of a P_4 that contains vx, as otherwise a contradiction to Lemma 5.2.4(i) applied to vc and d would arise. Therefore cv is a rib, Lemma 5.2.4(iii) applies cv and d; thus our P_4 can be labeled $ucvx$ and, together with d, induces an F_7. But we have already shown that such an F_7 leads to a contradiction.

***F_1*:** Let $a'b'c'd'$ be a P_4 weak-adjacent to $abcd$. Obviously, $v \notin \{a', b', c', d'\}$. Moreover, as all other possibilities have been ruled out, $a'b'c'd'$ and v induce another F_1. Therefore, by induction, v is $V(C_1^*)$-universal; thus v is not covered by C_1^*, a contradiction. □

It remains to show Theorem 5.2.1 for strong P_4-components. This proof is prepared by the following lemma, the analog of Lemma 3.3.7 for strong P_4-components.

Lemma 5.2.8 *Let $abcdrq$ be a net. If $D^*(abcd)$ is different from $D^*(abrq)$ and $D^*(dcrq)$, then r and q are not covered by $D^*(abcd)$.*

Proof. Let $D^* = D^*(abcd)$. We show that every P_4 $a'b'c'd' \in D^*$ together with r and q induces a net $a'b'c'd'rq$ and that neither $a'b'rq$ nor $d'c'rq$ is in D^*.

If $abcd$ is the only P_4 in D^*, then there is nothing to prove. For the inductive step, we show that our claim holds for some P_4 $a'b'c'd'$ on the assumption that it already holds for a strong-adjacent P_4 $abcd$. By the symmetry of the net, it suffices to consider the four cases of Table 5.2.

Case 1: $a'b'c'd' = abvd$. Then $abrq$ and $abvd$ are adjacent but not in the same strong P_4-component, thus Lemma 5.1.6 applies and $a'b'c'd'rq$ is a net. Moreover, $dcrq \sim dvrq$, hence $dvrq \notin D^*$.

Case 2: $a'b'c'd' = abcv$. Again $abcv$ and $abrq$ are adjacent but not in the same strong P_4-component, thus $a'b'c'd'rq$ is a net, and $vcrq \notin D^*$ follows from $dcrq \sim vcrq$.

Case 3: $a'b'c'd' = bcdv$. Since $bcdv$ is adjacent to $dcrq$, by Lemma 5.1.6, $bcdvrq$ is a net, a contradiction to $br \in E$.

Case 4: $a'b'c'd' = bavd$. The P_4 $bavd$ is adjacent to $abrq$, thus Lemma 5.1.6 implies that $bavdrq$ is a net, a contradiction to $br \in E$. □

Since two P_4s are strong adjacent if and only if the complement of those P_4s are strong adjacent, Lemma 5.2.8 also holds for the complement of a net, that is, for a pyramid.

Corollary 5.2.9 *Let abcdrp be a pyramid. If $D^*(abcd)$ is different from $D^*(apcr)$ and $D^*(bpdr)$, then r and p are not covered by $D^*(abcd)$.*

Now we are ready to show Theorem 5.2.1 for strong P_4-components. Its proof relies on the fact that we have already proved Theorem 5.2.1 for P_4-components.

Proof of Theorem 5.2.1 for strong P_4-components. Suppose the contrary, that is, two different strong P_4-components D_1^* and D_2^* satisfy $V(D_1^*) = V(D_2^*)$. Without loss of generality, we may assume that $V(D_1^*) = V$ (for otherwise we consider the graph $G_{V(D_1^*)}$).

Since D_1^* covers V, there is a P_4-component C^* that covers V. By Theorem 5.2.1, this P_4-component is unique, hence $D_1^* \subseteq C^*$ and $D_2^* \subseteq C^*$, thus a sequence $X_1^* = D_1^*, X_2^*, \ldots, X_{k-1}^*, X_k^* = D_2^*$ of strong P_4-components exists such that at least one P_4 in X_i^* is adjacent to a P_4 in X_{i+1}^*. Assume that this sequence is minimal with respect to k.

Since at least one P_4 in D_1^* is adjacent to a P_4 in X_2^* but $D_1^* \neq X_2^*$, by Lemma 5.1.6, a net $abcdrq$ exists with $abcd \in X_2^*$ and $abrq \in D_1^*$. If $dcrq \notin D_1^*$, then Lemma 5.2.8 implies that c and d are not covered by D_1^*, a contradiction. Hence $dcrq \in D_1^*$ and the same lemma implies that X_2^* does not cover r and q. Therefore X_2^* is separable, thus every P_4 $a'b'c'd'$ in X_2^* together with r and q induces the net $a'b'c'd'rq$. Furthermore two strong-adjacent P_4s must be of type (a) or (b) and, by induction, every P_4 $a'b'rq$ and $d'c'rq$ belongs to D_1^*.

Now consider a P_4 in X_2^* that is adjacent to a P_4 in X_3^*. By Lemma 5.1.6, those two P_4 are in a net $a'b'c'd'r'q'$ with $a'b'c'd' \in X_2^*$ and $a'b'r'q' \in X_3^*$. But $a'b'rq \in D_1^*$ is adjacent to $a'b'r'q' \in X_3^*$, a contradiction to our assumption that our sequence is minimal. Therefore $X_2^* = D_2^*$. But this is again a contradiction because X_2^* does not cover the whole graph. □

5.3 Prime split graphs

In this section, we analyze the structure of the (strong) P_4-components in prime split graphs. These results are then used to extend the split-modular decomposition of Section 4.3.

We start with investigating prime graphs that are covered by a (strong) P_4-component.

Theorem 5.3.1 *Let G be a prime graph. If G is covered by a strong P_4-component, then its maximal split-homogeneous sets are disjoint. Similarly, if G is covered by a P_4-component, then its maximal strict split-homogeneous sets are disjoint.*

Proof. Because of Theorem 5.2.2, Theorem 4.3.4 and Theorem 4.3.6, it suffices to show our theorem for prime split graphs G. Let D^* denote the strong P_4-component that covers G and suppose that two different maximal split-homogeneous sets $A = A^1 + A^2$ and $B = B^1 + B^2$ have nonempty intersection. Then $A \cup B$ is a split module, see Fact 4.3.3, hence $A^1 \cup B^1 + A^2 \cup B^2$ is a split-partition of G. Furthermore, Corollary 5.1.13 guarantees that no P_4 in D^* is in G_A or G_B.

Now suppose a P_4 $abcd$ in D^* satisfies $ab \in G_A$. Then it is impossible that $c \in A$ and $d \notin A$, for otherwise d would be A^1-partial. Similarly $c \notin A$ and $d \in A$ would imply that c is A^2-partial. Therefore both c and d are in $B - A$. As the symmetric argumentation applies to $cd \in G_B$, we also know that a and b are in $A - B$. Therefore no P_4 in D^* has a vertex in $A \cap B$, a contradiction because D^* covers G.

The above argumentation remains valid if we replace strong P_4-components, split modules and split-homogeneous sets with P_4-components, strict split modules and strict split-homogeneous sets. This proves the second part of the theorem. □

By the above theorem, it suffices to discuss the decomposition of prime split graphs that are not covered by any strong P_4-components. We first consider graphs containing vertices in no P_4.

Theorem 5.3.2 *Let $G = (V^1, V^2, E)$ be a prime split graph and v a vertex in no P_4 of G. Then $V^1 + V^2 - v$ is strict split-homogeneous in G and in $\overline{G}$.*

Proof. Since no P_4-component covers the whole graph, every P_4-component of G is separable and therefore implies a strict split-homogeneous set. Let $W = W^1 + W^2$ be a maximal strict split-homogeneous set. If a W-partial vertex r misses a W-universal vertex p, then a P_4 rw_1pw_2 exists with $w_1 \in W_1$ and $w_2 \in W_2$, a contradiction to the maximality of W because $V(C^*(rw_1)) \cup W$ is a larger strict split-homogeneous set. Therefore W is strict split-homogeneous in $\overline{G}$, thus our lemma follows from Lemma 4.3.5. □

The next lemma shows that, in the remaining cases, the graph is not covered by any P_4-component of G or $\overline{G}$.

Lemma 5.3.3 *Let $G = (V^1, V^2, E)$ be a prime split graph that is covered by a P_4-component of G and a P_4-component of $\overline{G}$. Then G is covered by a strong P_4-component.*

Proof. From Theorem 5.3.1 follows that the maximal strict split-homogeneous sets are disjoint, hence every P_4 not in a strict split-homogeneous set belongs to the P_4-component that covers G. By substituting nonadjacent marker vertices for maximal strict split-homogeneous sets, we therefore do not add P_4s to the P_4-component that covers G, and it is easy to see that we do not disconnect the P_4-component that covers $\overline{G}$. Furthermore, if a strong P_4-component covers the graph after the substitution, the same holds for the original graph. Therefore it suffices to show the theorem for prime split graphs without strict split-homogeneous sets in G and $\overline{G}$.

Suppose the theorem does not hold. Then the cover of every strong P_4-component is split-homogeneous but neither strict split-homogeneous in G nor strict split-homogeneous in $\overline{G}$. Let D^* denote a strong P_4-component that is maximal in the sense that no other strong P_4-component covers $V(D^*)$. Then vertices $p \in P$, $q \in Q$ and $r_1, r_2 \in R$ exist such that p misses r_1 and q sees r_2. Let $abcd$ be a P_4 in D^*.

Case 1: $r_1 = r_2$. If q misses p, then qr_1bpd is a P_5, thus $D^*(qr_1bp)$ is not split-homogeneous, a contradiction to our assumption. If q sees p, the same argument applies to the complement, so $r_1 = r_2$ is impossible.

Case 2: $r_1 \neq r_2$. Then p sees r_2 and q misses r_1, for otherwise we are back in Case 1. Note that we have the same situation in the complement, so we may assume that p misses q. Now $abr_2q \sim apr_2q \sim dpr_2q \sim dcr_2q$, and a simple inductive argument shows that $V(D^*) \subseteq V(D^*(abr_2q))$, a contradiction to the maximality of D^*. □

The following lemma provides the desired structural result.

Lemma 5.3.4 *Let $G = (V^1, V^2, E)$ be a prime split graph such that every vertex belongs to a P_4. If no P_4-component covers the whole graph, then $(V, \overline{E} - \overline{E}(V^2))$ consists of at least three connected components.*

Proof. In this proof, we call a P_4-component C^* maximal if no other P_4-component covers $V(C^*)$. Let C^* denote such a maximal P_4-component. As C^* does not cover the whole graph, $W = V(C^*)$ is strict split-homogeneous. If W is also strict split-homogeneous in $\overline{G}$, then Lemma 4.3.5 implies that not every vertex belongs to a P_4, a contradiction to our assumption. Therefore a vertex $p \in P$ misses a vertex $r \in R$.

Let $abcd$ be a P_4 in C^* and consider the bipartite graph $G' = (W, \overline{E}(W) - \overline{E}(V^2))$. Clearly G' has at most two connected components B and C with $b \in B$ and $c \in C$. Between b and every vertex $u \in B$, a (not necessarily simple) path $b, d, x_1, x_2, \ldots, x_k = u$ in G' exists. Clearly every pair of consecutive vertices x_i and x_{i+1} together with r and p induces a P_4 rx_ipx_{i+1} or $rx_{i+1}px_i$, and those P_4s belong to $C^*(rbpd)$. If G' has only one P_4-component, then $V(C^*)$ is covered by $C^*(rbpd)$, a contradiction to the maximality of C^*. Therefore the replacement of C^* with a maker P_4 $abcd$ neither increases the number of connected components of $(V, \overline{E} - \overline{E}(V^2))$ nor does it unify maximal P_4-components. We perform this substitution of marker P_4s for maximal P_4-components until every maximal P_4-component consists of a single P_4. It should be clear that the resulting graph is prime and split, so it suffices to show our theorem for graphs $G = (V^1, V^2, E)$ in which every P_4-component is a P_4 and vice versa.

Let $abcd$ denote a P_4 and let Q, R and P denote the vertex partition relative to $\{a, b, c, d\}$. Suppose that $Q \neq \emptyset$ and let $P_Q \subseteq P$ denote the vertices adjacent to some vertices in Q. Consider $H = \{a, b, c, d\} \cup R \cup (P - P_Q)$. Clearly every vertex in Q misses every vertex in H. Let p_q be a vertex in P_Q and let $q \in Q$ denote one of the vertices that sees p_q. Then p_q sees every vertex in R, for otherwise qp_qbr and qp_qcr would be adjacent P_4s, a contradiction for every P_4-component consists of a single P_4. Furthermore, if p_q misses a vertex in $p \in P - P_Q$, the same contradiction arises because of the adjacent P_4s qp_qap and qp_qdp. We conclude that every vertex in P_Q sees every vertex in H, hence H is homogeneous, a contradiction.

So we have shown that $Q = \emptyset$ for every P_4 $abcd$ in G. Since every vertex belongs to a P_4, the split partition is unique, hence $V^1 = \{b, c\} + P$ and $V^2 = \{a, d\} + R$. Note that b sees every vertex in V^2 except for d, and d sees every vertex in V^1 except for b. By symmetry, every vertex in V^1 misses precisely one vertex in V^2 and vice versa. Such a graph is called a thick spider in [5], and it is obvious that $(V, \overline{E} - \overline{E}(V^2))$ has $|V^1| = |V^2|$ components. But G contains a pyramid, hence $|V^1| \geq 3$. □

buildSplitModTree(G)

input: a graph $G = (V, E)$
output: the root of the split-modular decomposition tree of G

```
⋮
else (* G and Ḡ are connected and |V| > 1 *)
   let G' = (V', E') be the characteristic graph of G;
   if G' is split and a vertex v in V' is in no P_4 then
      let H be the vertex set of G that corresponds to v;
      let r_1 = buildSplitModTree( G_H );
      let r_2 = buildSplitModTree( G_{V-H} );
      return a 2-node with children r_1 and r_2
   elsif G' = (V^1, V^2, E') is split and (V', E - E(V^1)) has
     more than two connected components then
      let H_1, ... , H_t be the vertex sets of G that correspond
        to the connected components of (V', E - E(V^1));
      let r_i = buildSplitModTree( G_{H_i} ) for i = 1, ... , t;
      return a 3-node with children r_1, ... , r_t
   elsif G' = (V^1, V^2, E') is split and (V', Ē - Ē(V^2)) has
     more than two connected components then
      let H_1, ... , H_t be the vertex sets of G that correspond
        to the connected components of (V', Ē - Ē(V^2));
      let r_i = buildSplitModTree( G_{H_i} ) for i = 1, ... , t;
      return a 4-node with children r_1, ... , r_t
   else  (* G' is covered by a strong P_4-component *)
      let S_1, ··· , S_k be the vertex sets of G that correspond
        to maximal split-homogeneous sets of G';
      let s_i = buildSplitModTree( G_{S_i} ) for i = 1, ... , k;
      let H_1, ... , H_t be those maximal proper modules of G
        which are not contained in S_1, ... , S_k;
      let r_i = buildSplitModTree( G_{H_i} ) for i = 1, ... , t;
      return a 5-node with children s_1, ... , s_k, r_1, ... , r_t
   fi
fi
```

Algorithm 5.1

Lemma 5.3.4 holds for the complement as well, that is, if the graph $G = (V, E)$ is not covered by a P_4-component of $\overline{G}$, then $(V, E - E(V^1))$ has at least three connected components. It is also easy to see that $(V, E - E(V^1))$ is connected if the number of connected components of $(V, \overline{E} - \overline{E}(V^2))$ is greater than two. Theorem 5.3.5 summarizes our findings.

Theorem 5.3.5 *Let $G = (V^1, V^2, E)$ be a prime split graph in which every vertex belongs to a P_4. If no strong P_4-component covers the whole graph, then either $(V, \overline{E} - \overline{E}(V^2))$ or $(V, E - E(V^1))$ consists of at least three connected components.*

The split-modular decomposition extended with the results of Theorem 5.3.1, 5.3.2 and 5.3.5 is given in Algorithm 5.1.

5.4 Computing the split-modular decomposition

The purpose of this section is to propose an efficient implementation of Algorithm 5.1, that is, we prove Theorem 5.4.1.

Theorem 5.4.1 *The split-modular decomposition of an arbitrary graph $G = (V, E)$ can be found in $O(|V|^4)$.*

In our implementation of Algorithm 5.1, we use an associated graph $\tilde{G} = (\tilde{V}, \tilde{E})$ to compute the strong P_4-components. This associated graph is defined as follows.

- The vertices of $\tilde{V}$ are the ribs of the P_4s in G and $\overline{G}$.
- For every P_4 $abcd$ in G, there is an edge between bc and ad in $\tilde{E}$. We call these edges *p-edges* because they represent the P_4s of G.
- For every P_4 $abcd$ of G and $\overline{G}$ such that ab is a vertex in $\tilde{V}$, there is an edge between ab and bc in $\tilde{E}$.

Figure 5.5 illustrates this construction. The given graph G contains fourteen P_4s (the corresponding p-edges are indicated by thick lines), and its separable strong P_4-components $a_1b_1c_1d_1$ and $a_2b_2c_2d_2$ induce

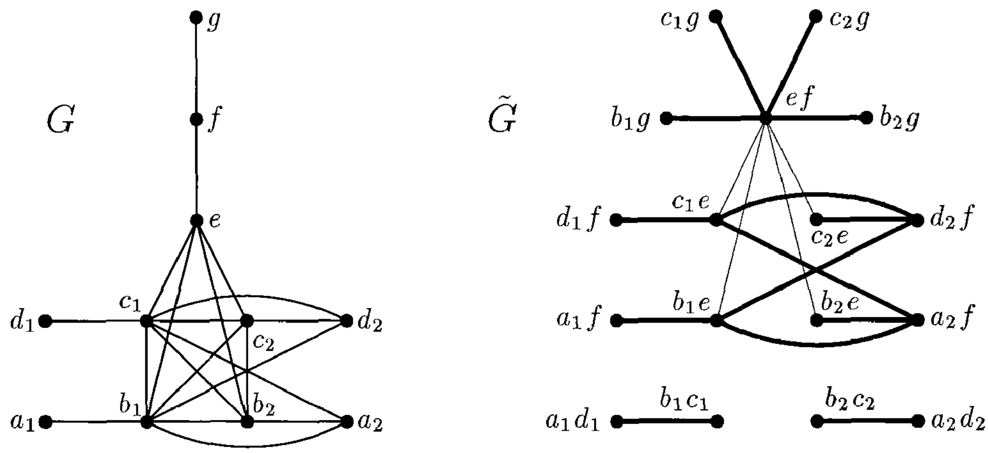


Figure 5.5: *A graph $G = (V, E)$ and its associated graph $\tilde{G} = (\tilde{V}, \tilde{E})$.*

the maximal split-homogeneous set $\{b_1, b_2, c_1, c_2\}+\{a_1, a_2, d_1, d_2\}$. Note that the strong P_4-components of G coincide with the connected components of $\tilde{G}$. This holds in general, as we prove in the next lemma.

Lemma 5.4.2 *The strong P_4-components of G are the connected components of $\tilde{G}$ and vice versa.*

Proof. First we prove that if two p-edges belong to the same connected component of $\tilde{G}$, then the corresponding P_4s are in the same strong P_4-component. Since every vertex in $\tilde{G}$ is incident to a p-edge, it suffices to give a proof for two p-edges that do not induce a $2K_2$.

If two p-edges have a common endpoint, then the corresponding P_4s or their complements have a common rib, thus, by Lemma 5.1.3, they belong to the same strong P_4-component. So suppose that two p-edges e_1 and e_2 have no common endpoint but an endpoint of e_1 is joined to an endpoint of e_2 by an edge, say e_3. If e_3 is a p-edge, then it follows from the above argumentation that the corresponding P_4s belongs to the same strong P_4-component. Otherwise, if e_3 is no p-edge, then the endpoints of e_3 are either the ribs of P_4s in G and one rib is the wing of the other P_4 or the same holds in the complement; thus Lemma 5.1.4 guarantees that those P_4s belong to the same strong P_4-component.

Second we show that if two P_4s belong to the same P_4-component, then the corresponding p-edges are in the same connected component of $\tilde{G}$. Clearly it suffices to show that two p-edges corresponding to strong-adjacent P_4s do not induce a $2K_2$. If $abcd$ denotes one P_4, then the other P_4 is of type (a) to (d) as given in Table 5.2.

Case 1: The P_4 is of type (a). Then the complement of the two P_4s have a common rib, hence the corresponding p-edges have a common endpoint.

Case 2: The P_4 is of type (b). Then the two P_4s have a common rib, and again the corresponding p-edges have a common endpoint.

Case 3: The P_4 is of type (c). Then, by construction, there is an edge between bc and cd.

Case 4: The P_4 is of type (d). Let $bavd$ denote our P_4. Then $adbv$ is a P_4 in the complement, and ad is a rib of the P_4 $bdac$ (in $\overline{G}$); thus there is an edge between bd and ad. □

We compute $\tilde{G}$ and its connected components in a preprocessing step. Since a P_4 is uniquely defined by its wings, this can be done in $|\tilde{E}| = O(|E|^2)$. By Lemma 5.1.3, all P_4s with the same rib belong to the same P_4-component; thus we can assign strong P_4-components to the ribs of P_4s in G. Furthermore, for every edge vw in G, we store the number $p(vw)$ of P_4s that contain vw.

To prove Theorem 5.4.1, we show that, except for the exploration of $\tilde{G}$, our algorithm runs in $O(|V^4|)$. Since there are at most $|V|$ recursive calls, it suffices to show that each recursive step can be done in $O(|V|^3)$.

Clearly, the computation of the connected component of G and $\overline{G}$ can be found in $O(|V|^2)$. If G is connected and coconnected, we calculate the characteristic graph G'. This step is in $O(|V|^3)$ as shown in Section 3.3 Page 33. Next, we check whether G' is split. This can also be done in linear time by testing whether G' and $\overline{G}'$ are triangulated, see Theorem 3.4.1(ii).

If G' is split, we compute a maximum clique of G'. By Lemma 4.3.5, Theorem 5.3.2 and the discussion on Page 48, the maximum clique of G' is unique, thus $G' = (V^1, V^2, E')$ where V^1 denotes the maximum clique. If $G' = (V^1, V^2, E')$ contains a vertex in V^1 that misses every vertex in V^2, we decompose G accordingly. Otherwise, we test whether $(V, \overline{E}' - \overline{E}'(V^2))$ or $(V, E - E(V^1))$ has more than two connected components. If so, we decompose G into the subgraphs induced by those connected components. Obviously all these steps can be done in $O(|V|^2)$.

In the remaining cases, by Theorem 5.2.2 and Lemma 5.3.3, G is covered by a strong P_4-component D^*. To find D^*, we scan the edges in G until we find an edge whose assigned strong P_4-component satisfies $|V(D^*)| = |V|$. Next, we scan the P_4s $abcd$ in D^*, decrease $p(ab)$, $p(bc)$ and $p(cd)$ and then compute the connected components $C_1, \ldots, C_k$ of the subgraph defined by the edges vw with $p(vw) > 0$.

Note that the C_is are homogeneous sets in G or strict split-homogeneous sets or split-homogeneous in G'. We can easily distinguish between these three possibilities in $O(|V|^2)$ by using an array of vertices for every connected component C_i that stores C_i-universal, C_i-null and C_i-partial vertices. However, maximal split-homogeneous sets need not be induced by a single connected component C_i: The graph of Figure 5.5 is such an example. As in case of the bipartite-modular decomposition, we have to take the disjoint union of the so far computed split-homogeneous sets if the union is again split-homogeneous. This can be done in $O(|V|^3)$ by first calculating the sets P, R and Q for every split-homogeneous set and then performing the tests whether the union of two sets is split-homogeneous.

In a last step, again as in case of the bipartite-modular decomposition, we have to find the maximal split-homogeneous sets that contain vertices which do not belong to a P_4 in the maximal split-homogeneous set. This last step can be implemented by examining all pairs of the so far computed split-homogeneous sets A and B together with the set of $A^1 \cup B^1$-partial or $A^2 \cup B^2$-partial vertices. By precalculating P, R and Q for the so far computed split-homogeneous sets, this step can be carried out in $O(|V|^3)$. The overall running time per recursive call is therefore $O(|V|^3)$, which proves our theorem.

5.5 Recognizing and orienting P_4-comparability graphs

In order to obtain an acyclic P_4-transitive orientation, it suffices to compute an acyclic orientation of the edges in the P_4s (all other edges can be oriented by topological sorting). In the following, we only discuss this part of the orientation.

If no P_4-component covers a proper subset of the vertices of G, then Theorem 5.2.1 guarantees that G has at most one nontrivial P_4-component. In this case, a P_4-transitive orientation is easy to compute because the orientation of one edge of a P_4 in a P_4-component forces the orientation of all P_4-edges in the same P_4-component.

So suppose a P_4-component C^* does not cover the whole graph. Then either $V(C^*)$ is homogeneous or C^* is separable. If G contains a homogeneous set H, we compute a P_4-transitive orientation of G as follows.

(*i*) Replace H with a marker vertex h

(*ii*) Compute a P_4-transitive orientation of the P_4s in G_H and in G_{V-H+h}.

(*iii*) Construct a P_4-transitive orientation of the P_4s in G by directing P_4-edges
vw with $v, w \in H$ as in G_H,
vw with $v, w \in V - H$ as in G_{V-H+h},
vw with $v \in H$ and $w \in V - H$ as hw in G_{V-H+h}.

Obviously, a P_4-transitive orientation of G induces a P_4-transitive orientation of G_H and G_{V-H+h}. The converse holds because of the following lemma.

Lemma 5.5.1 *If the orientation of the P_4s in G_H and G_{V-H+h} is P_4-transitive, then (iii) gives a P_4-transitive orientation of the P_4s in G.*

Proof. To begin with, we show that every P_4 in G is oriented properly. This is obvious for P_4s with all vertices in H and for P_4s with all vertices in $V-H$. The remaining P_4s have precisely one vertex in H, hence such a P_4 has a corresponding P_4 in G_{V-H+h}. Since both P_4s are oriented in the same way, those P_4s are oriented properly.

Now suppose the orientation of G is cyclic. As the orientation of G_H and G_{V-H+h} is acyclic, every cycle contains edges with both endpoints in H and edges with an endpoint in $V - H$. Choose a cycle with a minimal number of vertices in H and let $v \to \cdots \to w$ denote the longest part of this cycle in H. Furthermore, let u be the predecessor of v and x the successor of w in this cycle; thus $u, x \in V - H$. Since uv is directed, it must belong to a P_4 with precisely one vertex in H. By substituting w for v in this P_4, we obtain a P_4 that is oriented in the same way. Therefore $u \to w$, a contradiction because we have found a cycle with fewer vertices in H. □

If the cover of C^* is not homogeneous, then C^* is separable. In Section 5.1, we have seen that the coconnected components of G_{W_1} and the connected components of G_{W_2} are homogeneous sets, so we can substitute marker vertices for those components and compute a P_4-transitive orientation of the P_4s as described above. In the graph after the substitution, the vertex set corresponding to $V(C^*)$ is strict split-homogeneous. If a graph has a strict split-homogeneous set W, however, we can proceed as follows.

(*i*) Replace W^1 and W^2 with nonadjacent marker vertices w_1 and w_2.

(*ii*) Compute a P_4-transitive orientation of the P_4s in G_W and in $G_{V-W+w_1+w_2}$.

(*iii*) Construct a P_4-transitive orientation of the P_4s in G by directing P_4-edges
vw with $v, w \in W$ as in G_W,
vw with $v, w \in V - W$ as in $G_{V-W+w_1+w_2}$,
vw with $v \in V - W$ and $w \in W^1$ as vw_1 in $G_{V-W+w_1+w_2}$ and
vw with $v \in V - W$ and $w \in W^2$ as vw_2 in $G_{V-W+w_1+w_2}$.

A P_4-transitive orientation of G induces a P_4-transitive orientation of G_W and $G_{V-W+w_1+w_2}$. The converse is established by the next lemma.

Lemma 5.5.2 *If the orientation of the P_4s in G_W and $G_{V-W+w_1+w_2}$ is P_4-transitive, then (iii) gives a P_4-transitive orientation of the P_4s in G.*

Proof. The structure of this proof is identical to that of Lemma 5.5.1. So we first show that every P_4 in G is oriented properly. Again this is obvious for P_4s with all vertices in W and for P_4s with all vertices in $V - W$. The remaining P_4s are of types (1) to (6), for each of which a corresponding P_4 in $G_{V-W+w_1+w_2}$ exists that is oriented in the same way. Thus every P_4 is oriented properly.

Now suppose the orientation of G is cyclic. As the orientation of G_W and $G_{V-W+w_1+w_2}$ is acyclic, every cycle contains edges with both endpoints in W and edges with an endpoint in $V - W$. Choose a cycle with a minimal number of vertices in W and let $v \rightarrow \cdots \rightarrow w$ denote the longest part of this cycle in W. Furthermore, let u be the predecessor of v and x the successor of w in this cycle; thus $u, x \notin W$.

Since uv is directed, it must belong to a P_4 of types (1) to (6). Moreover v and w cannot belong to the same set of the split-partition $W^1 + W^2$ because this would imply $u \to w$, i.e. a cycle with fewer vertices in W exists. Without loss of generality, let $v \in W^2$ (otherwise we invert the orientation of the directed edges). Hence $u \in P$.

Then uv is in no P_4 of types (1) or (2), as otherwise $u \to w_2$ and $u \to w_1$ in $G_{V-W+w_1+w_2}$ and therefore $u \to w$, again a contradiction because this implies a cycle with fewer vertices in W. For the same reason, uv cannot belong to a P_4 of types (4) to (6), see Figure 4.1. Now assume that uv is in a P_4 of type (3), say p_1vur. Then $G_{V-W+w_1+w_2}$ contains the P_4s p_1w_2ur and $rw_1p_1w_2$; hence $r \to w_1$ in $G_{V-W+w_1+w_2}$ and therefore $r \to w$ in G. Thus $u \to v \to \cdots \to w$ can be replaced with $u \to r \to w$, a contradiction as this again implies a cycle with fewer vertices in W. □

Note that the above lemmas prove Theorem 5.1.1 because (a) if the P_4-classes of G can be P_4-transitively oriented, the same holds for the P_4-classes of G_H, G_{V-H+h}, G_W and $G_{V-W+w_1+w_2}$, and (b) this division into subproblems can be repeated until the graph has at most one P_4-class.

In the proof of Lemma 5.5.2, we have only used the fact that every P_4 with at least one but not all its vertices in W is of type (1) to (6). Therefore the described divide and conquer method is also applicable to the cover $W = V(C^*)$ of a separable P_4-component. We use this fact to prove Algorithm 5.2, the P_4-analog of GOLUMBIC's algorithm.

Note that, for a homogeneous set H, the removal of some edges in G_H does not create new P_4s with at least one edge in G_{V-H}. Similarly, it is easy to see that, for a strict split-homogeneous set $W = W^1 + W^2$, the only P_4s with some edges in G_{V-W} that are created by the removal of edges between W^1 and W^2 are of type (6). The latter accounts for Lines (6) to (9) in Algorithm 5.2, i.e. we remove only the wings of the P_4s in a separable P_4-component C^* from the graph $G = (V, E + E')$.

Now let $C^* = C^*(vw)$ be the P_4-component of $G = (V, E + E')$ as in Line (5). The orientation of the P_4-components of $G_{V(C^*)}$ is independent of the orientation of the other P_4-components if we guarantee that P_4-edges between vertices in $V(C^*)$ and a vertex in $V - V(C^*)$ are directed in the same way. We show that the latter constraint is satisfied because the corresponding P_4-edges belong to P_4s in the same P_4-component. This is obvious if $V(C^*)$ is homogeneous. Otherwise C^*

orient(G)

input: a graph $G = (V, E)$
output: a P_4-transitive orientation of the P_4s in G

```
let E' denote the set of edges in no P4 of G;
let E denote the set of P4-edges of G;
while E ≠ ∅ do
   choose an edge vw in E;
   orient the P4-component C*(vw) of G = (V, E + E');
   if C*(vw) is separable then
      let E_rib be the ribs of the P4s in C*(vw);
      E' ← E' + E_rib;
   fi;
   E ← E − C*(vw);
od
```

Algorithm 5.2

is separable, hence every P_4 with one but not all its vertices is of types (1) to (6) on Page 52. For P_4s of types (1) to (5), it follows from Figure 4.1 that they belong to the same P_4-component. For P_4s of type (6) , the removal of the wings in C^* ensures that the corresponding P_4s belong to the same P_4-component.

Now consider the orientation of $G_{V(C^*)}$. Without loss of generality, we may assume that $G_{V(C^*)}$ is prime as the substitution of marker vertices for homogeneous sets does not unify different P_4-components. Then Theorem 5.3.1 applies, hence all P_4s not in maximal strict split-homogeneous sets belong to C^* and are therefore oriented correctly relative to the maximal strict split-homogeneous sets. Similarly, the removal of the wings of the P_4s in C^* does not affect the remaining P_4-components of $G_{V(C^*)}$ as those P_4-components belong to disjoint maximal strict split-homogeneous sets.

So Algorithm 5.2 is correct and runs in $O(|V|^2 \cdot |E|)$, the time needed to find the P_4-components of G in BFS-manner.

Theorem 5.5.3 *P_4-comparability graphs can be oriented and recognized in $O(|V|^2 \cdot |E|)$ time and $O(|V| + |E|)$ space.*

To be more precise, the running time of our algorithm is bounded by $O(\delta^2 \cdot |E|)$ where δ is the maximal degree of a vertex. This can easily be seen because, given an edge vw, there are at most δ P_3s containing

vw, and it can be tested in $O(\delta)$ time whether this P_3 belongs to a P_4 (providing the adjacency lists of G are sorted). If we sacrifice the $O(|V| + |E|)$ space, we can improve the running time of our algorithm.

Theorem 5.5.4 *P_4-comparability graphs can be oriented and recognized in $O(|E|^2)$ time and $O(|V| \cdot |E|)$ space.*

Proof. First note that every P_4 is uniquely determined by its wings, thus all P_4s of $G = (V, E)$ can be found in $O(|E|^2)$ time. To compute and orient the P_4-components of G, we use the graph $\tilde{G} = (\tilde{V}, \tilde{E})$ where $\tilde{V} = E$ and two vertices e_1, e_2 are adjacent in $\tilde{G}$ if e_1 and e_2 are adjacent edges in a P_4 of G, i.e. e_1 and e_2 form a P_3 that is part of a P_4. Obviously the connected components of $\tilde{G}$ correspond to the P_4-components of G.

The initial construction of $\tilde{G}$ requires scanning every P_4 of G. As mentioned before, this can be carried out in $O(|E|^2)$. Furthermore $O(|\tilde{E}|) = O(|V| \cdot |E|)$ because an edge can belong to at most 2n P_3s.

When replacing $V(C^*)$ with marker vertices, $\tilde{G}$ can be updated by relabeling and deleting vertices of $\tilde{G}$; hence all these updates can be done in $O(|V| \cdot |\tilde{V}|) + O(|\tilde{E}|) = O(|V| \cdot |E|)$. But a connected component of $\tilde{G}$ is explored at most twice (to find a P_4-component that does not cover G and to orient the P_4-component). Therefore, after the initialization of $\tilde{G}$, our algorithm runs in $O(|V| \cdot |E|) + O(|\tilde{V}| + |\tilde{E}|) = O(|V| \cdot |E|)$. □

5.6 A general recognition algorithm

In this section, we show how to apply the split-modular decomposition to recognize perfectly orderable graphs. If a graph is disconnected or codisconnected, it is straight-forward to obtain a perfect order. If a graph $G = (V, E)$ contains a homogeneous set H, then we substitute a marker vertex h for H, find a perfect order of G_H and G_{V-H+h}. A perfect order of G can then be constructed from the perfect order of G_{V-H+h} by replacing h with the vertices in H where the vertices in H retain their order in G_H. As in the previous section, no obstruction can arise as every P_4 in G has a corresponding P_4 in G_H or G_{V-H+h} oriented in the same way.

It remains to find a perfect order of prime graphs. Note that to avoid an obstruction, it suffices to orient one wing in every P_4 from the midpoint to the endpoint. We call this a partial obstruction-free orientation. If a partial obstruction-free orientation is acyclic, then a perfect orientation is easily obtained by topological sorting.

If G is a prime split graph $G = (V^1, V^2, E)$, then any order that satisfies $v_1 < v_2$ for $v_1 \in V^1$ and $v_2 \in V^2$ is perfect. On the other hand, every partial obstruction-free orientation of a split graph $G = (V^1, V^2, E)$ has to orient some edges from V^1 to V^2. Since split-homogeneous sets induce split graphs, we substitute nonadjacent marker vertices w^1 and w^2 for split-homogeneous sets $W = W^1 + W^2$ and compute a perfect order in $G_{V-W+w^1+w^2}$ with $w^1 < w^2$. A perfect order of G is then obtained by replacing w^1 and w^2 with the vertices in W^1 and W^2. The following lemma shows that this method is correct.

Lemma 5.6.1 *A graph G with split-homogeneous set $W = W^1 + W^2$ has a perfect order if and only if $G_{V-W+w^1+w^2}$ has a perfect order that satisfies $w^1 < w^2$.*

Proof. If $G_{V-W+w^1+w^2}$ has a perfect order that satisfies $w^1 < w^2$, then the order arising from replacing w^1 and w^2 with the vertices in W^1 and W^2 is obstruction-free as every P_4 in G except for P_4s with a wing in G_W has a corresponding P_4 in $G_{V-W+w^1+w^2}$ oriented in the same way. P_4s with a wing in G_W are either in G_W or of type (12) as defined on Page 53. In both cases, the wing in G_W is oriented from the midpoints to the endpoints.

Conversely, let $\vec{G}$ be a perfect orientation of G and let $abcd$ be a P_4 in G_W. Then $a \leftarrow b$ or $c \rightarrow d$. Without loss of generality, let $c \rightarrow d$. From $\vec{G}$, we construct an orientation $\vec{G'}$ of $G_{V-W+a+c}$ by orienting every edge as in $\vec{G}$ except for edges av with $v \leftarrow a$ and $v \rightarrow d$ in $\vec{G}$. The latter edges are oriented $v \rightarrow a$ in $\vec{G'}$. We claim that $\vec{G'}$ is a perfect orientation and that inserting $c \rightarrow a$ in $\vec{G'}$ leaves the graph acyclic.

Since $\vec{G}$ is a perfect orientation, an obstruction in $\vec{G'}$ corresponds to a P_4 with a wing whose orientation in $\vec{G'}$ is different from that in $\vec{G}$. Therefore an obstruction is a P_4 $vaxy$ with $v \rightarrow a$ in $\vec{G'}$ and $v \leftarrow a$ in $\vec{G}$. It is easy to see that this P_4 must be of type (2) or (3). But this is a contradiction because $vdxy$ would be an obstruction in $\vec{G}$.

Now assume that $\vec{G'}$ has a cycle. Since $\vec{G}$ is acyclic, this cycle contains an edge whose orientation in $\vec{G}$ is different from that in $\vec{G'}$. Let $u \rightarrow a$ in $\vec{G'}$ be this edge in the cycle and let x denote the successor of a in the cycle. Then $u \rightarrow d$ and $d \rightarrow x$ in $\vec{G}$ by construction. We can therefore replace every occurrence of $u \to a \to x$ in our cycle with $u \to d \to x$, thereby obtaining a cycle in $\vec{G}$, a contradiction to our assumption that $\vec{G}$ is acyclic.

Finally, suppose that inserting $c \to a$ in $\vec{G'}$ causes a cycle $c \to a \to v \to \cdots \to c$. By replacing $c \to a \to v$ with $c \to d \to v$ and by replacing $u \to a \to x$ with $u \to d \to x$ as before, we obtain a cycle in $\vec{G}$, again a contradiction to our assumption. □

The graph after replacing maximal split-homogeneous sets with marker vertices need not be prime relative to the split-modular decomposition. The above method can therefore be applied repeatedly until we end up with a prime graph without maximal split-homogeneous sets. To find a perfect order that satisfies the additional constraints $w^1 < w^2$, however, might be difficult.

On the other hand, if the graph admits at most two partial obstruction-free orientations according to some predefined rules, then we just have to test whether topological sorting together with the additional constraints yields an acyclic orientation. This is clearly the case for rules which enforce that $a \leftarrow b$ or $c \to d$ implies $a' \leftarrow b'$ or $c' \to d'$ for any pair of strong-adjacent P_4s $abcd$ and $a'b'c'd'$. Orienting every P_4 transitively is an example of such a rule.

Furthermore, every set of rules that orients the wings of strong-adjacent P_4s is almost sufficient: The only exception of strong-adjacent P_4s without a common wing is the P_5. So if we give a set of rules that orients the wings of strong-adjacent P_4s and the wings in the P_5s, then there are at most two orientations possible, thus the corresponding class of perfectly orderable graphs can be recognized in polynomial time.

In [35], Hertz proposed a simple rule to obtain a partial obstruction-free orientation from a 2-coloring of the edges in the complement given the edges ab and cd of a P_4 $abcd$ and the edges ab and de of a P_5 $abcde$ have different colors. He also showed that the arising partial obstruction-free orientation is acyclic. Even without this result, we can easily recognize this class of graphs since, by the above remarks, a graph without split-homogeneous sets has precisely two such 2-colorings. So we just have to orient the edges according to Hertz' rule and test whether the arising orientation is acyclic.

Chapter 6

Graphs with Threshold Dimension Two

Graph dimension theory deals with the (edge-)intersections of graphs with special properties. COZZENS AND ROBERTS [20] gave the following definition.

Definition 6.0.1 *The $\mathcal{P}$ dimension of a graph $G = (V, E)$ is the least integer k such that $E = E_1 \cap E_2 \cap \cdots \cap E_k$ and each of the graphs $G_i = (V, E_i), i = 1, \ldots k$, has property $\mathcal{P}$.*

The term "dimension" in Definition 6.0.1 comes from the interval dimension problem, one of the first graph dimension problems that has been investigated. The interval dimension or *boxicity* $b(G)$ of a graph is the least number k such that G is the intersection of k interval graphs. Since an interval graph is the intersection graph of intervals on the real line $\mathbb{R}$, a graph with boxicity k is the intersection graph of iso-oriented boxes in the Euclidean space $\mathbb{R}^k$. A graph with boxicity 2 is therefore the intersection graph of axially parallel rectangles in the plane, which is why such graphs is also called *rectangle graphs*, see Figure 6.1.

The representation of a graph by geometrical objects is interesting because it is possible to use geometrical algorithms to solve certain graph problems [47]. For instance, if a geometrical model of a boxicity k graph is given, a maximum clique can be found in $O(|V|^{(k-1)} \log(|E|))$, see [42]. However, the maximum stable set problem, the minimum

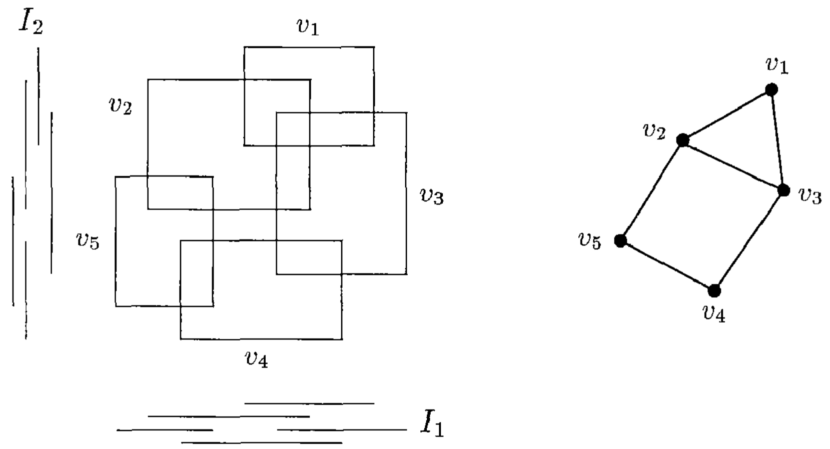


Figure 6.1: *A rectangle graph, its rectangle model and the two interval models I_1 and I_2.*

coloring problem and the minimum clique cover problem remain NP-complete even for rectangle graphs [42]. Furthermore, YANNAKAKIS [76] showed that the recognition of graphs with boxicity k is NP-complete for all $k \geq 3$, and KRATOCHVÍL [46] obtained the same complexity result for the case $k = 2$.

Closely related to the boxicity of a graph is its *threshold dimension*, defined as the least integer k such that the graph is the intersection of k threshold graphs. Threshold graphs are a proper subclass of interval graphs, hence the boxicity of a graph is less or equal to its threshold dimension. The motivation for studying the threshold dimension of graphs also comes from the many applications in integer programming [13, 14, 33], in psychology [18, 19] and in the synchronization of parallel processes [25, 34, 61, 62, 64, 74].

Unfortunately, it is NP-complete to test whether a graph has threshold dimension k for all $k \geq 3$, see YANNAKAKIS [76]. So research focused on the recognition of graphs with threshold dimension two. These graphs are also interesting because of their nice optimization behavior: Both graphs with threshold dimension two and their complements (called *2-threshold graphs*) are perfectly orderable [31, 15, 35].

For over a decade, the complexity status of the recognition of threshold dimension two remained open. In fact, it was widely believed that the problem is NP-complete [18, 32, 52, 64]. Recently, however, MA [51] and, independently, RASCHLE AND SIMON [66] succeeded in finding polynomial time algorithms for the recognition of graphs with threshold dimension two.

MA's idea to recognize graphs with threshold dimension two is to construct a geometrical representation for such a graph. The running time of his algorithm is $O(|V|^5)$. Recently, STERBINI AND RASCHLE [75] proposed an improved version of MA's algorithm that runs in $O(|V|^3)$. Although this is currently the fastest algorithm, we discuss neither MA's approach nor the improvements made by STERBINI AND RASCHLE for the following reasons. First, MA's algorithm is rather complicated and relies on other quite complicated algorithms like the $O(|V|^2)$ recognition of 2-chain graphs and the $O(|V|^2)$ recognition of partial order dimension two. Second, the geometrical arguments used in MA's and STERBINI AND RASCHLE's work do not fit in with the graph theoretical outline of this thesis.

This chapter is thus devoted to a in-depth discussion of RASCHLE AND SIMON's approach. Their main result is a constructive proof of a conjecture by IBARAKI AND PELED which states that a graph is 2-threshold if and only if its edges can be colored with two colors such that the nonincident edges in a P_4, C_4 or $2K_2$ receive different colors. This immediately leads to a $O(|E|^2)$ algorithm for the recognition of 2-threshold graphs.

In the next section, we give CHVÀTAL AND HAMMER's definition of threshold graphs and threshold numbers and discuss their motivation for studying threshold graphs. In Section 6.2, we review previous results in connection with 2-threshold graphs and state the main theorems. Finally, Section 6.3 gives a constructive proof of the IBARAKI-PELED conjecture. Part 1 and 2 of this proof follow RASCHLE AND SIMON's original work [66] whereas Part 3 contains new and hopefully simpler proofs based on a new structure theorem on what we call the AC_4-structure of graphs.

6.1 Threshold graphs

Threshold graphs were introduced by CHVÁTAL AND HAMMER [13] in 1973. Their motivation for studying these graphs comes from the aggregation of linear inequalities in integer programming. It is frequent in integer and zero-one programming that the problem is given in the form "maximize cx such that $Ax \leq b$," and it is well-known that the work involved in solving the problem often increases sharply with the number of linear inequalities. Therefore, given a set of constraints

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i, \quad i = 1, 2, \ldots, m \tag{6.1}$$

one is interested in finding a system with the smallest number k of linear inequalities

$$\sum_{j=1}^{n} a'_{ij}x_j \leq b'_i \quad i = 1, 2, \ldots, k \tag{6.2}$$

such that (6.1) and (6.2) have precisely the same set of zero-one solutions. In particular, one wishes to know whether $k = 1$, namely whether the constraints (6.1) can be aggregated to a knapsack constraint in the same binary variables.

If (6.1) are set-packing constraints, that is, if A is a zero-one matrix and b is the vector of ones, system (6.1) can be represented as a graph $G = (V, E)$ whose vertices correspond to the columns of $A = (a_{ij})$ and two vertices are adjacent if the corresponding columns have a 1 in a common row. A solution of (6.1) corresponds to a stable set of G and vice versa. This observation motivates the following definition of threshold graphs and threshold numbers.

Definition 6.1.1 *Let $G = (V, E)$ be a graph with $V = \{v_1, v_2, \ldots, v_n\}$. For any subset $S \subseteq V$, its* characteristic vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ *is defined by $x_i = 1$ if $v_i \in S$ and $x_i = 0$ otherwise (for $i = 1, 2, \ldots, n$). The* threshold number $t(G)$ *of $G = (V, E)$ is the least integer k for which linear inequalities*

$$\begin{array}{rcl} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n & \leq & t_1, \\ & \vdots & \\ a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kn}x_n & \leq & t_k, \end{array} \tag{6.3}$$

exist such that a subset S of V is a stable set of G if and only if its characteristic vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ satisfies the above inequalities. A graph with $t(G) \leq 1$ is called threshold graph.

Since each subset S of V corresponds to a corner of the unit hypercube in $\mathbb{R}^n$, a threshold graph is a graph for which a hyperplane exists that cuts the n-space in half such that the corners of the hypercube corresponding to the stable sets of G lie on one side of the hyperplane and the corners of the hypercube corresponding to nonstable sets lie on the other side. In this interpretation, the threshold number of a graph is the minimal number of half spaces needed such that their intersection contains precisely the stable sets of G.

CHVÁTAL AND HAMMER [14] showed that the threshold number can be defined in an equivalent way.

Theorem 6.1.2 *The threshold number $t(G)$ is the least integer k such that G is the union of k threshold graphs.*

For a given graph $G = (V, E)$, a set of threshold graphs $G_i = (V, E_i)$ $i = 1, \dots, k$, with $E = E_1 \cup \dots \cup E_k$ is a *threshold cover* of *size* k. The threshold number of a graph is therefore the least integer k for which a threshold cover exists.

We conclude this section by showing that the threshold number of a graph is identical to the threshold dimension of its complement. Recall that the threshold dimension is defined to be the least integer k such that the graph is the intersection of k threshold graphs. Thus the threshold dimension of the complement $\overline{G}$ is the least integer such that G can be written has the union of the complements of k threshold graphs. Our claim now follows from the fact that the complement of a threshold graph is again a threshold graph. The latter is a consequence of Theorem 6.1.3(ii).

Theorem 6.1.3 *For a graph $G = (V, E)$, the following statements are equivalent:*

(i) *G is a threshold graph, i.e. there is a linear inequality $a_1x_1 + a_2x_2 + \dots + a_nx_n \leq t$ whose zero-one solutions are precisely the characteristic vectors of the stable sets of G.*

(ii) *G does not contained a P_4, C_4 or $2K_2$.*

(iii) *Every induced subgraph G_W contains a dominating or an isolated vertex.*

(iv) *G can be constructed from one vertex by repeatedly adding an isolated or a dominating vertex.*

Proof. (i) $\Rightarrow$ (ii) Suppose a P_4, C_4 or $2K_2$ exists and let it be labeled as in Figure 6.2. Then $\{v_1, v_3\}$ and $\{v_2, v_4\}$ are stable sets

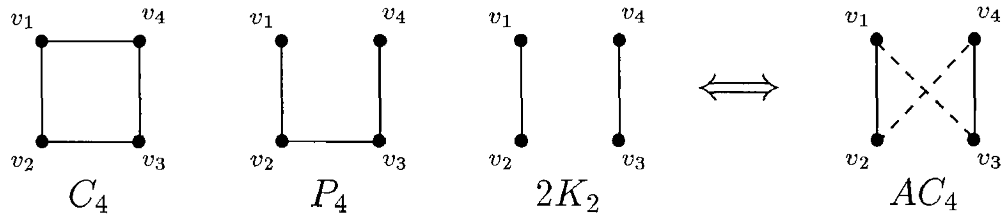


Figure 6.2: *The forbidden subgraphs of a threshold graph.*

whereas $\{v_1, v_2\}$ and $\{v_3, v_4\}$ are cliques. Therefore every inequality $a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq t$ has to satisfy $a_1 + a_3 \leq t$, $a_2 + a_4 \leq t$, $a_1 + a_2 > t$ and $a_3 + a_4 > t$, which is impossible.

$(ii) \Rightarrow (iii)$ Since G_W is P_4-free, by Lemma 3.1.1, every subgraph G_W is either disconnected or codisconnected. If G_W is disconnected and has no $2K_2$, then at most one connected component contains edges and all other connected components are isolated vertices. If $\overline{G_W}$ is disconnected and has no $2K_2$, the above applies to the complement, thus G_W contains at least one dominating vertex.

$(iii) \Rightarrow (iv)$ follows by induction.

$(iv) \Rightarrow (i)$ We show by induction on the number of vertices that a linear inequality $a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq t$ that satisfies (i) exists such that $a_1, \ldots, a_n$ and t are positive integers. If the graph consists of a single vertex v_1, we assign $a_1 = 1$ and $t = 1$. For the induction step, we assume that the linear inequality $a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq t$ has the desired properties. If we add a dominating vertex v_{n+1}, then we assign $a_{n+1} = t$. If we add an isolated vertex v_{n+1}, then we assign $a_{n+1} = 1$ and $t = 2t + 1$ and $a_i = 2a_i$ for $i = 1, \ldots, n$. □

6.2 Previous results

In this section, we review previous results on the threshold dimension. It is convenient to work on the complement rather than on the graph, so we consider the problem of finding the threshold number instead of the threshold dimension.

In a first step, we transform the threshold number problem into a coloring problem with additional constraints. For this purpose, we need the notion of alternating cycles and threshold completions.

An *alternating cycle* AC_{2k} in a graph $G = (V, E)$ is a sequence of vertices $v_0, v_1, \ldots, v_{2k-1}$ such that $v_i v_{i+1} \in E$ for i odd and $v_i v_{i+1} \in \overline{E}$ for i even (indices modulo $2k$). Note that, by convention, we always take $v_0 v_1$ to be a nonedge. The edges $v_i v_{i+1} \in E$ for i odd are called the *edges of* the alternating cycle. If all the edges of an alternating cycle belong to $S \subseteq E$, it is an *alternating cycle in* S. Figure 6.3 illustrates the only possible AC_4 and the only two possible AC_6s. There are no alternating cycles smaller than an AC_4 and, by Theorem 6.1.3(*ii*) and Figure 6.2, threshold graphs are precisely those graph without induced AC_4.

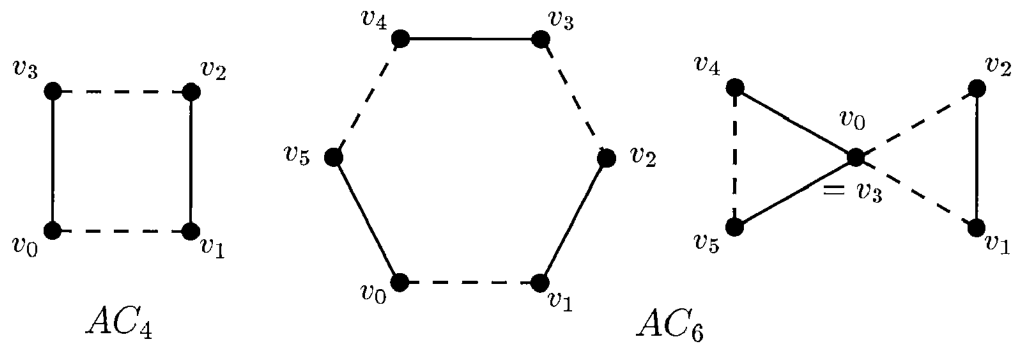


Figure 6.3: *The alternating cycles AC_4 and AC_6 (dashed lines indicate nonedges).*

For a given graph $G = (V, E)$ and a subset $S \subseteq E$, an edge set E' is called a *completion* of S if $S \subseteq E' \subseteq E$. Furthermore E' is a $\mathcal{P}$ completion if $G' = (V, E')$ satisfies the graph property $\mathcal{P}$. The problem of finding a $\mathcal{P}$ completion is also known as *graph sandwich problem*, see GOLUMBIC, KAPLAN AND SHAMIR [28].

We claim that Algorithm 6.1 solves the threshold sandwich problem. The algorithm is correct because every vertex in G_W is incident to an edge in $S \cap E(W)$, thus, by Theorem 6.1.3(*iii*), G_W must have a dominating vertex. It is also easy to implement Algorithm 6.1 to run in linear time by storing and updating the degree of the vertices in G_W similar to the implementation of topological sorting in [27]. Thus we have

Fact 6.2.1 *If an edge set $S \subseteq E$ has a threshold completion, then a threshold completion of S can be found in $O(|V| + |E|)$.*

threshold completion

input: a graph $G = (V, E)$ and edge sets $S \subseteq E$
output: a threshold completion T of E

```
T ← ∅;
W ← V(S);
while W ≠ ∅ do
   if G_W contains an dominating vertex v then
      T ← T + {vw ∈ E(W)};
      W ← W − {v}
   else
      stop    (* S has no threshold completion *)
   fi
od
```

Algorithm 6.1

Obviously the threshold number problem is equivalent to finding a minimal partition of E into edge sets for which a threshold completion exists. A necessary and sufficient condition for the existence of a threshold completion is the following.

Fact 6.2.2 *An edge set S has a threshold completion if and only if G does not contain an alternating cycle in S.*

Proof. Let E' be a threshold completion of S and let W denote the set of vertices in an alternating cycle in S. Since every vertex in W is incident to edges in $S(W)$ and $\overline{E}(W)$, the graph $G'_W = (W, E'(W))$ contains neither dominating nor isolated vertices. It follows from Theorem 6.1.3(ii) that $G' = (V, E')$ cannot be a threshold graph, a contradiction to our assumption, which proves the "only if" part.

To prove the "if" part, suppose that S has no threshold completion and G does not contain an alternating cycle in S. Without loss of generality, let G be minimal in the sense that $S(W)$ has a threshold completion in G_W for every $W \subset V$. Then every vertex is incident to a nonedge of G and to an edge in S. So we can grow a path alternating between nonedges and S-edges until an alternating cycle is obtained. □

In the light of Fact 6.2.1 and Fact 6.2.2, the threshold cover problem looks like an edge coloring problem.

> Find the smallest integer k for which a partition $E_1 + \ldots + E_k$ of E exists such that no alternating cycle has all its edges in E_i for $i = 1, \ldots, k$.

However, the constraint that every E_i must not contain an alternating cycle is much more complex than ordinary graph coloring. It is therefore natural to look for constraints which are easier to deal with. CHVÁTAL AND HAMMER [14] considered minimal AC_4–free edge partitions instead of the more restrictive AC_{2l}–freeness. For this purpose, they defined the graph G^* as follows.

Definition 6.2.3 *Two edges xy and vw of a graph $G = (V, E)$ are in* conflict *if they are the edges of an AC_4, and the* conflict graph *$G^* = (V^*, E^*)$ is defined by taking $V^* = E$ and by joining two vertices of G^* if the corresponding edges in G are in conflict.*

Notation: As in Chapter 4, we write $xy \parallel vw$ if x, v, w, y is an AC_4 and $xy \parallel wv$ if x, w, v, y is an AC_4. Thus $xy \parallel \cdots \parallel vw$ or $xy \parallel \cdots \parallel wv$ holds for any pair of edges xy and vw in the same connected component of G^*.

Trivially $\chi(G^*) \leq t(G)$ because a threshold cover of size $t(G)$ induces a coloring of G^* of the same size. CHVÁTAL AND HAMMER [14] asked whether there are graphs G with $t(G) > \chi(G^*)$. COZZENS and LEIBOWITZ [18] found examples of such graphs for the case $\chi(G^*) \geq 4$. On the other hand, IBARAKI AND PELED [41] showed that $t(G) = \chi(G^*)$ if G^* is bipartite and G is a split graph. They also conjectured that $t(G) = \chi(G^*)$ holds if G^* is bipartite. In this chapter, we give a constructive proof of IBARAKI AND PELED's conjecture. In other words, our main result is

Theorem 6.2.4 *If G^* is bipartite, then an AC_{2l}-free bipartition of G^* can be found in $O(|E|^2)$.*

The above theorem implies that if the bipartition of G^* is unique, then the color classes of G^* are a solution to our problem. In general, however, the number of connected components k of G^* is rather large, and the number of 2–colorings of G^* is 2^k.

We prove Theorem 6.2.4 by presenting an $O(|E|^2)$ algorithm that gradually transforms a given 2-coloring of G^* into a 2-coloring of G^* without monochromatic alternating cycles. In a first step, we show that it suffices to avoid the AC_6 rather than all alternating cycles.

Theorem 6.2.5 (Corollary 6 in [30]) *Let $G = (V, E)$ be a graph with $\chi(G^*) = 2$ and let E_1 and E_2 denote the color classes of G. If G has an alternating cycle in E_1, then G has an AC_6 in E_1 or E_2.*

Proof. Since G^* is bipartite, there is no AC_4 in E_1 or E_2. Let $C = v_0, v_1, \ldots, v_{2l-1}$ be a minimal alternating cycle in E_1.

First, we show that $v_iv_{i+3} \in E_2$ if v_i, v_{i+1}, v_{i+2}, and v_{i+3} are distinct. If $v_iv_{i+1} \in \overline{E}$, then $v_iv_{i+3} \in E$ because C is minimal, and $v_iv_{i+3} \in E_2$ because $v_iv_{i+3} \parallel v_{i+1}v_{i+2}$. Otherwise, if $v_iv_{i+1} \in E_1$, then the minimality of C prohibits $v_iv_{i+3} \in E_1$. But $v_iv_{i+3} \in \overline{E}$ would imply $v_iv_{i+1} \parallel v_{i+3}v_{i+2}$, thus indeed $v_iv_{i+3} \in E_2$ as claimed.

To prove the theorem, let $v_0, v_1, \ldots, v_{2l-1}$ be a labeling of our minimal alternating cycle C such that $v_0, v_1, \ldots, v_k$ is a longest sequence of distinct vertices. By the above remarks, $v_iv_{i+3} \in E_2$ for $i = 0, \ldots, k-3$. We distinguish the following cases.

Case 1: $k \geq 5$. Since v_0v_3, v_1v_4 and v_2v_5 belong to E_2, the sequence $v_0, v_1, v_4, v_5, v_2, v_3$ is an AC_6 in E_2.

Case 2: $k = 4$. Since $v_0v_3, v_1v_4 \in E_2$, either $v_5 = v_0$ or $v_5 = v_2$. If $v_5 = v_2$, then $v_0v_2 \in E$, for otherwise C could be shortened by replacing $v_0, \ldots, v_5$ with v_0, v_5. Furthermore $v_0v_2 \in E_1$ because $v_0v_2 \parallel v_1v_4$, hence $v_0, v_1, v_2, v_3, v_4, v_2$ is an AC_6 in E_1.

If $v_5 = v_0$, then $v_0v_2 \in E$ for otherwise we could obtain a shorter alternating cycle than C by replacing $v_0, \ldots, v_5$ with v_0, v_1, v_2, v_5. But $v_0v_2 \in E_2$ because $v_0v_2 \parallel v_4v_3$, thus $v_0, v_1, v_4, v_0, v_2, v_3$ is an AC_6 in E_2.

Case 3: $k = 3$. Then $v_4 = v_1$ because $v_0v_3 \in E_2$. Furthermore, it is easy to see that the vertices v_2, v_3, v_4 and v_5 are distinct, hence v_2v_5 is in E_2. Since $k = 3$, the vertex v_6 cannot be different from $v_2, \ldots, v_5$, thus $v_6 = v_3$. But this is a contradiction as $v_1v_2 \parallel v_5v_6$.

Case 4: $k = 2$. Then $v_3 = v_0$. Since $k = 2$, the vertex v_5 cannot be different from v_2, v_3, v_4, hence $v_5 = v_2$. Again this is a contradiction because $v_1v_2 \parallel v_3v_4$. □

As we saw in Figure 6.3, there are only two possibilities of an AC_6. We define these two possibilities as follows.

Definition 6.2.6 *An AC_6 $v_0, \ldots, v_5$ in one of the color classes of G^* is called an* alternating polygon *of length 5 or 6 (i.e* ***AP_5*** *and* ***AP_6*** *for short) according to the number of distinct vertices in $v_0, \ldots, v_5$.*

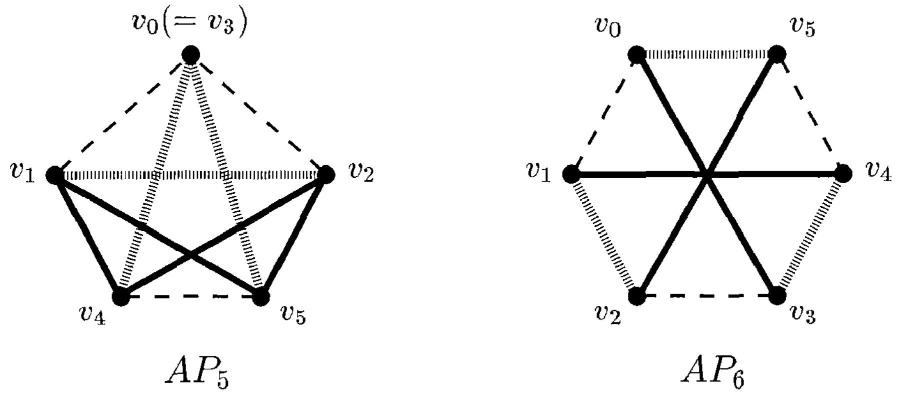


Figure 6.4: *An alternating polygon of length* 5 *and* 6.

In Figure 6.4 and subsequent figures, edges in one color class (usually E_1) are indicated by dotted lines and edges in the other color class (usually E_2) by thick lines. As illustrated in the figure, an AP_5 and an AP_6 force edges in the other color class by the bipartiteness of G^*. Thus an AP_6 $v_0, \ldots, v_5$ implies the *complementary* AP_6 $v_0, v_1, v_4, v_5, v_2, v_3$ in the *complementary* color class. Similarly, an AP_5 $v_0, \ldots, v_5$ implies the edges v_1v_4, v_1v_5, v_2v_4 and v_2v_5 in the complementary color class. Note also that all the edges of an AP_5 except possibly v_1v_2 and all the edges of an AP_6 belong to connected components of G^* with size greater than one.

6.3 Recognizing 2-threshold graphs

By Theorem 6.2.5, to prove the main result, it suffices to transform a given 2-coloring of G^* into an AC_6-free 2-coloring of G^*. We do this in three parts.

6.3.1 Part 1

In this part, we show how to transform an AP_6-free 2-coloring of G^* into an AC_6-free 2-coloring. Parts 2 and 3 show how to obtain an AP_6-free coloring.

Theorem 6.3.1 *From a given AP_6-free 2-coloring of G^*, an AC_6-free 2-coloring of G^* can be computed in $O(|E|^2)$.*

Proof. In this proof, we call an edge $v_1v_2 \in E$ *critical* if an AP_5 $v_0, \dots, v_5$ exists. Since a critical edge v_1v_2 in an AC_4 $v_1v_2 \parallel xy$ results in an AP_6 v_4, v_5, v_2, y, x, v_1, we conclude that every critical edge v_1v_2 is an isolated vertex in G^*. We claim that the following statement holds.

> The 2–coloring of G^* obtained by inverting the color of all critical edges is AC_6–free.

Certainly no new AP_6 can arise because every edge e of an AP_6 belongs to a connected component of G^* with size greater than one. On the other hand, all the original AP_5s are destroyed. So it remains to show that no new AP_5 is created.

Suppose the contrary, i.e. that a new AP_5 $w_0, \dots, w_5$ is created. Let $v_0, \dots, v_5$ be the AP_5 in the old coloring that caused $v_1v_2 = w_1w_2$ to change its color. Without loss of generality, we may assume that $v_1 = w_1$ and $v_2 = w_2$.

Note that all considered edges other than v_1v_2 retain their color (their connected component in G^* has size greater than one). Therefore $\{v_4, v_5\} \cap \{w_0, \dots, w_5\} = \emptyset$, and the situation is as illustrated in Figure 6.5.

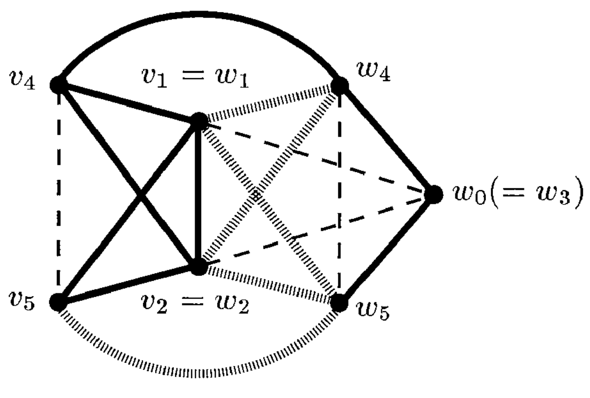


Figure 6.5: *A configuration in the proof of Theorem 6.3.1*

Since v_1v_4 and w_0w_4 have the same color and $w_0v_1 \notin E$, we find that $v_4w_4 \in E$ and similarly $v_5w_5 \in E$. But $v_4w_4 \parallel v_5w_5$, hence these edges must have different colors. If v_4w_4 has the same color as v_2v_5, then $v_5, v_4, w_4, w_5, w_0, v_2$ is an AP_6 in the old coloring of G^*, and otherwise $v_4, v_5, w_5, w_4, w_0, v_1$ is an AP_6 in the old coloring. Both cases contradict our assumption that the original coloring of G^* is AP_6–free.

To achieve the desired running time, we observe that the above proof does not make use of the vertex $v_0 = v_3$ of the AP_5 $v_0, \ldots, v_5$. Therefore the argument still holds if we relax the definition of a critical edge and say that an edge $v_1v_2 \in E$ is *critical* when there are vertices v_4, v_5 such that $v_4v_5 \notin E$, $\{v_1v_4, v_1v_5, v_2v_4, v_2v_5\} \subseteq E$ and all these four edges belong to the complementary color class of v_1v_2 and their connected components of G^* have size greater than one.

The decision whether an edge v_1v_2 is critical or not can now be made in linear time as follows. Mark all vertices x for which the edges xv_1 and xv_2 are in the complementary color class of v_1v_2 and their connected components in G^* have size greater than one. Then scan through the adjacency lists of the marked vertices to discover a pair v_4, v_5 of nonadjacent vertices. Clearly each of these operations can be done in $O(|V| + |E|)$, which completes our proof. □

6.3.2 Part 2

It remains to construct an AP_6-free 2-coloring of G^*. In order to study an AP_6 more closely, we extend our notation.

Definition 6.3.2 *The vertices v_0, v_1 of an AP_6 $v_0, \ldots, v_5$ are called the* base *of the AP_6 and the edge v_2v_5 its* front. *If in addition $v_0v_2, v_1v_5 \in E$ and v_0v_2, v_1v_5 belong to the color class of v_0v_5, then $v_0, \ldots, v_5$ is called a* double *AP_6.*

Figure 6.6 illustrates a double AP_6. Note that the complementary AP_6 $v_0, v_1, v_4, v_5, v_2, v_3$ is also a double AP_6.

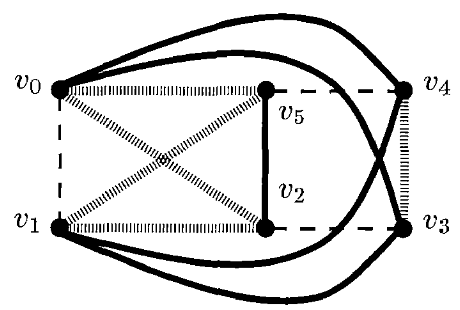


Figure 6.6: *A double AP_6.*

In this part, we transform a given double AP_6-free 2-coloring of G^* into an AP_6-free 2-coloring. The next three fundamental facts on AP_6s are also proved in [30].

Fact 6.3.3 *Let $v_0, \dots, v_5$ be an AP_6 and $v_2v_5 = x_0y_0 \parallel \cdots \parallel x_hy_h$ a path in G^* satisfying $\{x_0, \dots, x_h, y_0, \dots, y_h\} \cap \{v_0, v_1\} = \emptyset$. Then an AP_6 with base v_0, v_1 and front x_hy_h exists.*

Proof. We use induction on h. The case $h = 0$ is just our assumption. If $h \geq 1$, then the induction hypothesis implies the existence of an AP_6 $v_0, v_1, y_{h-1}, \dots, x_{h-1}$ or an AP_6 $v_0, v_1, x_{h-1}, \dots, y_{h-1}$. In the former case, from $x_{h-1}y_{h-1} \parallel x_hy_h$ and $\{x_h, y_h\} \cap \{v_0, v_1\} = \emptyset$ we infer the existence of the AP_6 $v_0, v_1, y_{h-1}, y_h, x_h, x_{h-1}$. Then the complementary AP_6 $v_0, v_1, x_h, x_{h-1}, y_{h-1}, y_h$ satisfies our claim. The latter case is similarly treated. □

Fact 6.3.4 *Let $v_0, \dots, v_5$ be a double AP_6 and $v_2v_5 = x_0y_0 \parallel \cdots \parallel x_hy_h$ a path in G^*. Then $\{x_0, \dots, x_h, y_0, \dots, y_h\} \cap \{v_0, v_1\} = \emptyset$ and a double AP_6 with base v_0, v_1 and front x_hy_h exists.*

Proof. Again we use induction on h. The case $h = 0$ is our assumption. For $h \geq 1$, the induction hypothesis implies $\{x_0, \dots, x_{h-1}, y_0, \dots, y_{h-1}\} \cap \{v_0, v_1\} = \emptyset$ and the double AP_6 $v_0, v_1, y_{h-1}, \dots, x_{h-1}$. Therefore $\{x_h, y_h\} \cap \{v_0, v_1\} = \emptyset$. From this and the fact that $x_{h-1}y_{h-1} \parallel x_hy_h$, we obtain the double AP_6 $v_0, v_1, y_{h-1}, y_h, x_h, x_{h-1}$, whose complementary double AP_6 $v_0, v_1, x_h, x_{h-1}, y_{h-1}, y_h$ satisfies our claim. □

Fact 6.3.5 *Let $v_0, \dots, v_5$ be an AP_6 and $v_2v_5 = x_0y_0 \parallel \cdots \parallel x_hy_h$ a path in G^* satisfying $\{x_h, y_h\} \cap \{v_0, v_1\} \neq \emptyset$. Then a double AP_6 exists.*

Proof. Without loss of generality, we may assume that $\{x_0, \dots, x_{h-1}, y_0, \dots, y_{h-1}\} \cap \{v_0, v_1\} = \emptyset$. Thus Fact 6.3.3 guarantees either an AP_6 $v_0, v_1, y_{h-1}, \dots, x_{h-1}$ or an AP_6 $v_0, v_1, x_{h-1}, \dots, y_{h-1}$. Because of symmetry, it suffices to discuss the first possibility.

Again without loss of generality, assume that $y_h = v_0$ as illustrated in Figure 6.7 (the case $x_h = v_1$ is similar). From $x_{h-1}y_{h-1} \parallel x_hy_h =$

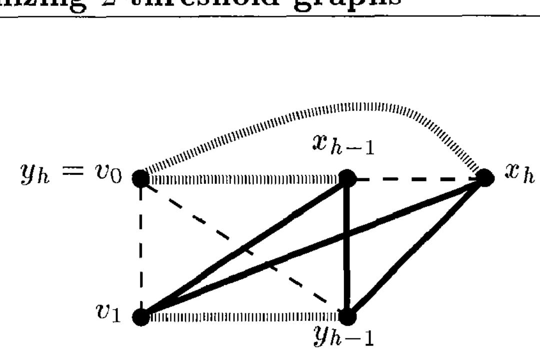


Figure 6.7: *A configuration in the proof of Fact 6.3.5*

$x_h v_0$, we obtain the AP_5 $v_0, v_1, y_{h-1}, v_0, x_{h-1}, x_h$, which also induces the complementary edges $v_1 x_{h-1}, v_1 x_h, y_{h-1} x_{h-1}$ and $y_{h-1} x_h$.

On the other hand, $v_1 y_{h-1}$ is an edge of the AP_6 $v_0, v_1, y_{h-1}, \dots, x_{h-1}$ and therefore another edge $xy \in E$ with $xy \parallel v_1 y_{h-1}$ exists. Thus $x_h, x_{h-1}, v_1, x, y, y_{h-1}$ is a double AP_6. □

Now we are ready to prove the main result of Part 2.

Theorem 6.3.6 *From a given double AP_6-free 2-coloring of G^*, an AP_6-free 2-coloring of G^* can be computed in $O(|E|^2)$.*

Proof. Let $E_1 + E_2$ be the bipartition of G^*. We assume there is an AP_6 $v_0, \dots, v_5$, for otherwise we are done. With respect to these fixed vertices v_0 and v_1, let

$$H = \{xy \in E \mid xy \text{ is the front of an } AP_6 \text{ with base } v_0, v_1\}$$

Since $E_1 + E_2$ is double AP_6-free, Fact 6.3.5 and Fact 6.3.3 imply that if an edge xy belongs to H, then all edges in the same connected component $C^*(xy)$ belong to H. Therefore, if we swap the color of all edges in H, we obtain another 2-coloring of G^*. For this new 2-coloring, we assert the following.

$$\textit{No edge in } H \textit{ is an edge of an } AP_6. \qquad (6.4)$$

In order to prove this assertion, we assume that the new coloring has an AP_6 $w_0, \dots, w_5$ with one of its edges in H, and obtain a contradiction. Without loss of generality, we assume that $w_0 w_5 \in H$ and that $w_0 w_5 \in E_2$. (We always refer to the "old" coloring if not mentioned otherwise, thus $w_0, \dots, w_5$ is an AP_6 in the new coloring.) Since $w_0 w_5 \parallel w_1 w_4$, either $v_0, v_1, w_0, w_1, w_4, w_5$ or $v_0, v_1, w_5, w_4, w_1, w_0$ is an AP_6 in E_1. The symmetry allows us to assume the first case.

Figure 6.8 illustrates this situation, including the edges $v_0w_1, v_1w_4 \in E_2$ of the complementary AP_6 $v_0, v_1, w_4, w_5, w_0, w_1$ in E_2. The vertices w_2 and w_3 remain to be specified.

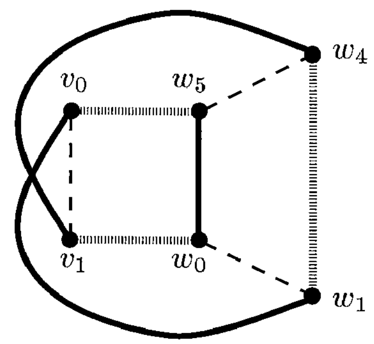


Figure 6.8: *The base configuration in the proof of Theorem 6.3.6*

Since all edges incident to v_0 or v_1 retain their color and $v_0w_1, v_1w_4 \in E_2$, a choice of $w_2 = v_0$ or $w_3 = v_1$ would contradict our assumed AP_6 $w_0, \ldots, w_5$ in the new coloring. The following possibilities remain.

Case 1: $|\{v_0, v_1, w_0, \ldots, w_5\}| = 8$.

Case 1.1: $w_1w_2 \in H$. Then $w_1w_2 \in E_2$ and $w_0w_3 \in E_1$. From $w_1w_2 \in H$, Fact 6.3.3 implies the existence of an AP_6 in E_1 with base v_0, v_1 and front w_1w_2. But $v_0w_1 \in E_2$, and therefore $v_0w_2, v_1w_1 \in E_1$ must be edges of this AP_6. A closer look reveals that $v_0, v_1, w_1, w_0, w_3, w_2$ is an AP_6 in E_1, thus its complementary AP_6 guarantees $v_0w_0, v_1w_3 \in E_2$, as illustrated in Figure 6.9.

Case 1.1.1: $w_3w_4 \in H$. The above argument applied to w_3w_4 instead of w_1w_2 results in $v_0w_4, v_1w_3 \in E_1$, which is impossible.

Case 1.1.2: $w_3w_4 \notin H$. Then $w_3w_4 \in E_1$ and therefore $w_2w_5 \in E_2$. Further, since $v_1w_3, w_2w_5 \in E_2$ and $w_3w_2 \notin E$, we must have $v_1w_5 \in E$. Moreover, $v_1w_5 \in E_2$, for otherwise $v_0, v_1, w_5, w_4, w_3, w_2$ would be an AP_6 in E_1, which would result in $w_2w_5, w_3w_4 \in H$, contrary to our case. Further, as depicted in Figure 6.9, since $w_0v_0, v_1w_4 \in E_2$ and $v_0v_1 \notin E$, we must have $w_0w_4 \in E$. But $w_0w_4 \in E_2$ implies the double AP_6 $w_4, w_5, w_0, w_1, v_0, v_1$ whereas $w_0w_4 \in E_1$ implies the double AP_6 $w_0, w_1, w_4, w_5, v_0, v_1$, a contradiction to our assumption.

Case 1.2: $w_3w_4 \in H$. This case is symmetric to Case 1.1.

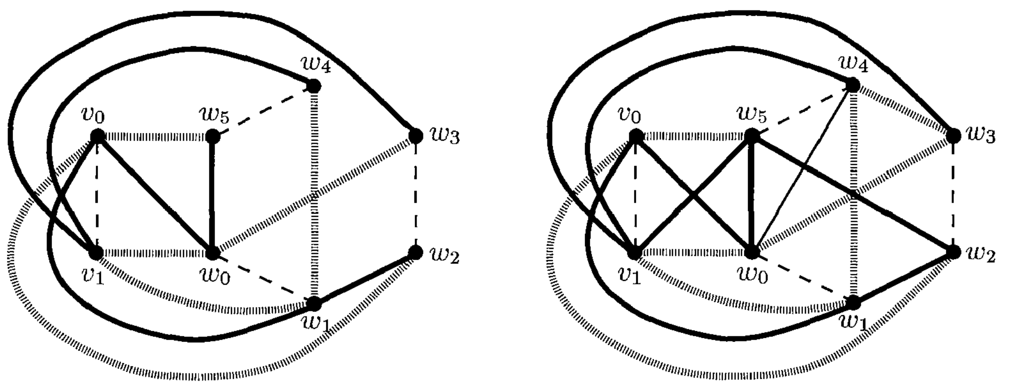


Figure 6.9: *Cases in the proof of Theorem 6.3.6*

Case 1.3: $w_1w_2 \notin H$ and $w_3w_4 \notin H$. In this case $w_1w_2, w_3w_4 \in E_1$ and $w_0w_3, w_2w_5 \in E_2$. Since $v_0w_5, w_4w_3 \in E_1$ and $w_5w_4 \notin E$, we must have $v_0w_3 \in E$. Further, $v_0w_3 \in E_1$, for otherwise $w_2w_5, w_3w_4 \in H$ because of the AP_6 $v_0, v_1, w_4, w_5, w_2, w_3$. The symmetric argument leads to $v_1w_2 \in E_1$, which contradicts $v_0w_3 \parallel v_1w_2$, see Figure 6.10.

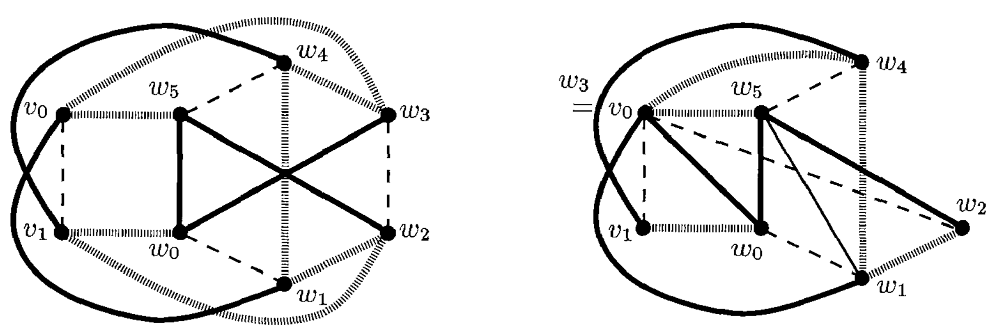


Figure 6.10: *Cases in the proof of Theorem 6.3.6*

Case 2: $v_0 = w_3$ and $v_1 \neq w_2$. If $w_1w_2 \in H$, then $w_0w_3 = w_0v_0 \in H$, a contradiction. Therefore $w_1w_2 \notin H$, hence $w_1w_2 \in E_1$ and $w_0w_3 = w_0v_0 \in E_2$. Further, since $w_1v_0, w_2w_5 \in E_2$ and $v_0w_2 \notin E$, we have $w_1w_5 \in E$. But w_1w_5 does not have an admissible coloring because $w_1w_5 \in E_2$ implies the double AP_6 $w_0, w_1, v_0, v_1, w_4, w_5$ and $w_1w_5 \in E_1$ implies the double AP_6 $w_4, w_5, v_0, v_1, w_0, w_1$.

Case 3: $v_0 \neq w_3$ and $v_1 = w_2$. This case is symmetric to Case 2.

Case 4: $v_0 = w_3$ and $v_1 = w_2$. Then $w_1v_1, v_0w_4 \notin H$, hence $w_1v_1, v_0w_4 \in E_1$ and $w_0v_0, v_1w_5 \in E_2$. Since $w_0v_0, v_1w_4 \in E_1$ and $v_0v_1 \notin E$, we have $w_0w_4 \in E$. Further, $w_0w_4 \in E_1$, for otherwise the double AP_6 $w_4, w_5, v_1, v_0, w_1, w_0$ exists. The symmetric argument leads to $w_1w_5 \in E_1$, which contradicts $w_0w_4 \parallel w_1w_5$, see Figure 6.11.

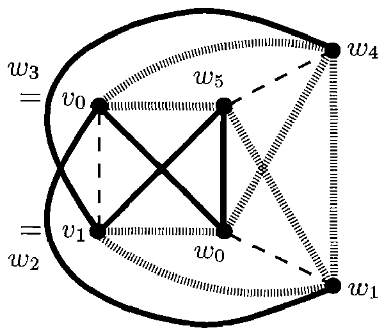


Figure 6.11: *Case 4 in the proof of Theorem 6.3.6*

Since all the cases above lead to contradictions, we conclude that the new coloring does not have an AP_6 with an edge in H, and therefore our assertion (6.4)) holds.

But then the new coloring has fewer AP_6s than the old one. Continuing in this way, we achieve an AP_6-free coloring in $|E|$ steps. To show the $O(|E|^2)$ running time, it therefore suffices to prove that the determination whether an AP_6 $v_0, \ldots, v_5$ exists for a given edge v_2v_5 and the computation of H can be done in $O(|V| + |E|)$.

First we consider the former problem. Since the coloring of G^* is double AP_6-free, Fact 6.3.5 implies $\{x, y\} \cap \{v_0, v_1\} = \emptyset$ for each AP_6 $v_0, \ldots, v_5$ with $xy \parallel v_2v_5$; hence the AP_6 v_0, v_1, v_2, x, y, v_5 also exists. Therefore a fixed edge v_3v_4 conflicting with v_2v_5 can be chosen in advance. The remaining search for the base v_0, v_1 is in $O(|V| + |E|)$.

As to the computation of H, an edge xy is in H if and only if it is the front of an AP_6 with base v_0, v_1. Again, it is easy to see that if such an AP_6 exists, then an AP_6 with base v_0, v_1, front xy and an edge vw must also exist whenever $vw \parallel xy$. Therefore, the computation of H is also in $O(|V| + |E|)$. □

6.3.3 Part 3

In this part, we present an efficient method to transform a 2-coloring of G^* into a double AP_6-free 2-coloring, which is the remaining task according to Theorem 6.2.4. In order to do this, we need a deeper analysis of the AC_4-structure in the presence of an AP_6.

We start with studying the connected components of G^* for arbitrary graphs. In analogy to the previous chapters, we call the edges in a connected component of G^* an *AC_4-class*. In the rest of this section, C^* stands for an AC_4-class and $C^*(vw)$ for the AC_4-class that contains the edge vw. Let P, Q and R denote the sets of $V(C^*)$-universal, $V(C^*)$-null and $V(C^*)$-partial vertices, respectively. Note that every edge with one endpoint in Q must have the other endpoint in P.

The next theorem analyzes the neighborhood relation between the vertices in $V(C^*)$ and those in $V - V(C^*)$.

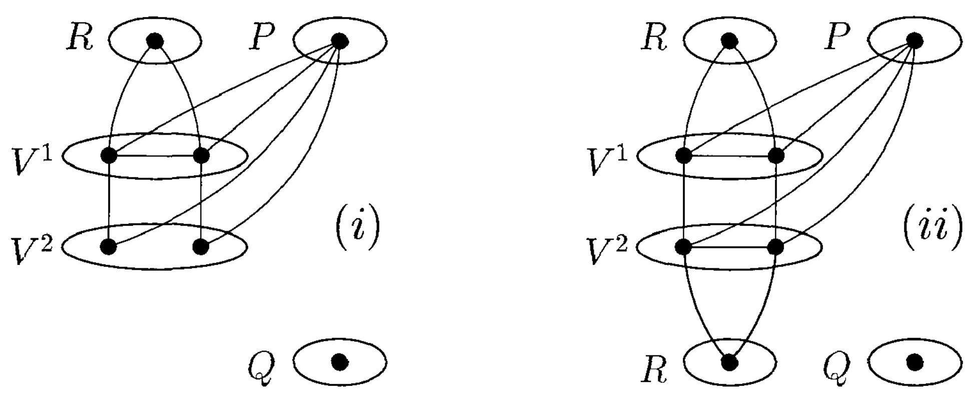


Figure 6.12: *Case (i) and (ii) of Theorem 6.3.7.*

Theorem 6.3.7 *Let C^* be a nontrivial AC_4-class of an arbitrary graph $G = (V, E)$. If $R \neq \emptyset$, then a unique partition $V(C^*) = V^1 + V^2$ exists such that every edge in C^* has one endpoint in V^1 and the other in V^2, and either*

(i) V^1 is a clique, V^2 is a stable set and every vertex in R is V^1-universal and V^2-null or

(ii) V^1 and V^2 are cliques and no vertex in R is V^1- or V^2-partial.

Proof. Let v be an arbitrary vertex in R. Clearly an AC_4 $ab \parallel cd$ in C^* exists such that v is $\{a, b, c, d\}$-partial. If v is $\{a, b\}$-universal and

$\{c,d\}$-null, then $av \parallel cd$, a contradiction to $v \notin V(C^*)$. Without loss of generality, we may assume that v sees b but misses a. Then v sees c, for otherwise $bv \parallel dc$, a contradiction to $v \notin V(C^*)$. Similarly, v misses d, because otherwise $dv \parallel ba$. By induction, every edge in C^* has one endpoint in $V^1 = N(v) \cap V(C^*)$ and the other in $V^2 = \overline{N}(v) \cap V(C^*)$.

Next, we show that V^1 is a clique. Suppose the contrary, i.e. there are nonadjacent vertices x and z in V^1. Since x is covered by C^*, an edge $xy \in C^*$ exists. Then v misses y and therefore $xy \parallel zv$, a contradiction to $v \notin V(C^*)$.

Since V^1 is a clique, every pair of conflicting edges in C^* induces a P_4 or a C_4. We show that either every AC_4 in C^* is a P_4 or every AC_4 in C^* is a C_4. Suppose that this does not hold. Then AC_4s $ab \parallel cd$ and $cd \parallel ef$ in C^* exist such that $abcd$ is a P_4 and c,d,e,f is a C_4. Clearly $a,d,e \in V^2$ and $b,c,f \in V^1$, thus e is different from a,b,c and d. Furthermore $C^* \neq C^*(de)$ because $cv \parallel ed$ and because v is not covered by C^*. So $ab \parallel ed$ is impossible, hence a sees e. Now $cv \parallel ae \parallel dc$, a contradiction to $v \notin V(C^*)$.

It remains to prove that (i) holds if every AC_4 in C^* is a P_4 and that (ii) holds if every AC_4 in C^* is a C_4. We first consider the case that every AC_4 in C^* is a P_4. Assume that V^2 contains adjacent vertices a and x. Then there is a P_4 $abcd$ with $ab, cd \in C^*$. Now x sees d, for otherwise $cv \parallel ax \parallel cd$, a contradiction to $v \notin V(C^*)$. Thus we have shown that given x sees one endpoint of a P_4 with its wings in C^*, then x sees the other endpoint as well. It follows by induction that x cannot be covered by C^*, a contradiction to our assumption. So V^2 is a stable set. Since every vertex in $V^1 + V^2$ belongs to a P_4 in $G_{V^1+V^2}$, the partition $V^1 + V^2$ is unique and, as we have chosen v arbitrarily, every vertex in R sees V^1 but misses V^2.

Finally, we prove that if every pair of conflicting edges in C^* induces a C_4, then (ii) holds. To show that V^2 is a clique, we assume the contrary, i.e. there are nonadjacent vertices a and x in V^2. Let $ab \parallel cd$ denote a C_4 in C^*. Then $xd \notin E$, for otherwise $bv \parallel dx \parallel ba$, a contradiction to $v \notin V(C^*)$. By induction, x misses every point V^2, which implies that x cannot be covered by C^*, a contradiction. So we have shown that V^2 is a clique. Since $\overline{G}_{V^1+V^2}$ is connected, the partition $V(C^*) = V^1 + V^2$ is unique, hence every vertex in R induces the same partition. □

Remark: The above theorem shows that the cover of an AC_4-class is a module, a special split module or a special cobipartite module. From the theorems of Chapter 4, it follows that Theorem 6.3.7 can be used to obtain a unique graph decomposition. A GALLAI-type theorem, however, does not hold, even if G^* is bipartite: The complement of a P_6 is a prime graph with bipartite G^* and it has two AC_4-classes which both cover the whole graph.

If G^* is bipartite and there is a double AP_6 $v_0, \ldots, v_5$, then Fact 6.3.4 asserts that $C^* = C^*(v_2v_5)$ does not cover the whole graph. Now this is just the interesting case with respect to the recognition of 2-threshold graphs. So we study this situation in more detail.

Without loss of generality, assume that $v_0, \ldots, v_5$ is a double AP_6 in E_1. From the definition of a double AP_6 and the existence of its complementary double AP_6, we derive

$$\begin{aligned} \{v_0v_2, v_0v_5, v_1v_2, v_1v_5, v_3v_4\} &\subseteq E_1, \\ \{v_0v_3, v_0v_4, v_1v_3, v_1v_4, v_2v_5\} &\subseteq E_2. \end{aligned} \tag{6.5}$$

Let W stand for the cover of $C^* = C^*(v_2v_5)$, that is, $W = V(C^*)$. Note that v_0 and v_2 see both endpoints of v_2v_5, so Theorem 6.3.7 implies that v_0 and v_1 belong to P. Therefore $W^1 = \{k \in W \mid v_0k \in E_1\}$ and $W^2 = \{k \in W \mid v_0k \in E_2\}$ is a partition of W.

Lemma 6.3.8 *$W^1 + W^2$ is a partition of W into cliques.*

Proof. Let x and y be nonadjacent vertices in W^1. Then $v_0x, v_0y \in E_1$. By Fact 6.3.4, there is a double AP_6 $v_0, v_1, x, \ldots$, hence $v_1x \in E_1$. But this contradicts $v_1x \parallel v_0y$. The same contradiction arises if we assume that x and y are nonadjacent vertices in W^2. □

Next assume that case (ii) of Theorem 6.3.7 holds. Then v_2, v_4, v_3, v_5 is a C_4. Without loss of generality, we may assume that a vertex r in R sees v_2, v_4 and misses v_3, v_5. If r misses v_0, then $v_0v_5 \parallel rv_4 \parallel v_3v_5 \parallel v_2r \parallel v_3v_0$, a contradiction because v_0v_5 and v_0v_3 have different colors. But if r sees v_1, then $v_1v_5 \parallel v_0r \parallel v_1v_3$, again a contradiction because v_1v_5 and v_1v_3 have different colors.

Therefore either W is a module or Theorem 6.3.7(i) holds. In the latter case, $v_2v_5 \parallel v_2v_4$ induces a P_4. Without loss of generality, we assume that this P_4 is $v_2v_5v_3v_4$.

Lemma 6.3.9 *$Q \cup R \cup \{v_0, v_1\}$ is a stable set and*

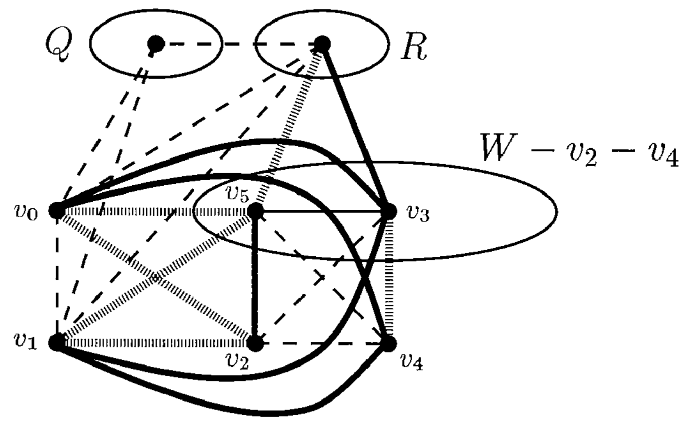


Figure 6.13: *Case (ii) of Lemma 6.3.9.*

(i) W is a module or

(ii) $W - v_2 - v_4$ *is a clique,* $\{v_2, v_4\}$ *is a stable set and every vertex in R sees every vertex in W except for* v_2 *and* v_4.

Proof. Since $W = V(C^*)$, vertices in Q can only be adjacent to vertices in P. If a vertex $q \in Q$ sees v_0, then $v_1v_2 \parallel v_0q \parallel v_1v_3$, a contradiction because v_1v_2 and v_1v_3 have different colors. The case that a vertex in Q sees v_1 is similar, thus $Q \cup \{v_0, v_1\}$ is stable, which proves the lemma if $R = \emptyset$.

If $R \neq \emptyset$, then Theorem 6.3.7(*i*) holds, thus $v_2v_5v_3v_4$ is a P_4. Since G_W induces a split graph and every vertex in W is in a P_4 in G_W, the split partition of W is unique and v_2 and v_4 belong to the stable set in the split partition. On the other hand, the stable set consists of at most two vertices because of Lemma 6.3.8. So it remains to show that $R \cup \{v_0, v_1\}$ is stable.

If a vertex $r \in R$ sees v_0, then $v_1v_2 \parallel v_0r \parallel v_1v_4$, a contradiction because v_1v_2 and v_1v_4 have different colors. The case that a vertex in R sees v_1 is similar, thus every vertex in R misses v_0 and v_1. If adjacent vertices r_1 and r_2 in R exist, then $v_0v_2 \parallel r_1r_2 \parallel v_0v_4$, again a contradiction to (6.5). □

The above lemma implies that every AC_4 $vw \parallel xy$ with $v, w \in W$ satisfies $x, y \in W$. The next corollary follows by induction.

Corollary 6.3.10 *If* $vw \in E(W)$, *then* $C^*(vw) \subseteq E(W)$.

Next, we investigate AC_4-classes that contain edges between $V - W$ and W and edges with both endpoints in $V - W$.

Lemma 6.3.11 *If an edge vw with $v \in V - W$ and $w \in W$ satisfies $C^*(vw) \cap E(V - W) \neq \emptyset$, then v is W-universal and every edge between v and a vertex in W belongs to $C^*(vw)$ and has the same color as vw.*

Proof. We first show the lemma for the case that no vw belongs to an AC_4 $vw \parallel xy$ with $x, y \in V - K$.

If $v \in R$, then $vv_5 \in E_1$ and $vv_3 \in E_2$ because $vv_5 \parallel v_0v_4$ and $vv_3 \parallel v_0v_2$, see Lemma 6.3.9(*ii*) and Figure 6.13. Furthermore $y \in Q$ because v sees $w \in W$ and y misses w. But this is contradiction to the coloring of vv_3 and vv_5 because $vv_3 \parallel xy \parallel vv_5$.

So v is W-universal. If $y \in Q$, then $xy \parallel vz$ for every vertex $z \in W$, thus every edge between v and a vertex in W belongs to $C^*(vw)$ and has the same color as vw.

If $y \in R$, then $x \in P$ and $vv_2 \parallel xy \parallel vv_4$. By Lemma 6.3.9(*ii*), v_2 misses v_4, hence $\overline{G}_W$ is connected. Thus every vertex $z_0 \in W$ is connected to either v_2 or v_4 by a path of length $2k+1$, say $z_0, \dots, z_{2k}$. Therefore $vz_0 \parallel xz_1 \parallel vz_2 \parallel xz_3 \parallel \cdots \parallel vz_{2k}$ with $z_{2k} = v_2$ or $z_{2k} = v_4$. So again every edge between v and a vertex in W belongs to $C^*(vw)$ and has the same color as vw.

It remains to show our lemma in the general case. Let $x_0y_0 \parallel x_1y_1 \parallel \cdots \parallel x_{k+1}y_{k+1}$ be a path in G^* that connects $vw = x_0y_0$ with an edge $x_{k+1}y_{k+1}$ in $E(V - K)$. Furthermore, let $x_{k+1}v_{k+1}$ be the first edge in this path with both endpoints in $V - W$ and let $x_k \in V - W$ and $y_k \in W$. We have already shown that our lemma holds for x_ky_k. By induction, it suffice to prove that it holds for $x_{k-1}y_{k-1}$.

Clearly $x_{k-1} \in V - W$ and $y_{k-1} \in W$. Since no vertex is G_W-dominating, every edge between x_{k-1} and a vertex in W belongs to an AC_4 whose other edge connects x_k with a vertex in W. Therefore every edge between x_{k-1} and W belongs to $C^*(x_ky_k) = C^*(vw)$ and has the same color as x_ky_k. Finally, if x_{k-1} were not W-universal, then $x_{k-1} \in R$, hence $x_{k-1}v_5 \parallel v_0v_3$ and $x_{k-1}v_3 \parallel v_0v_2$ and therefore $x_{k-1}v_5 \in E_1$ and $x_{k-1}v_3 \in E_2$, a contradiction because every edge between x_{k-1} and W has the same color. □

Let vw be an edge as described in the above lemma and suppose that vw is the front of an double AP_6 $k_0, k_1, v, \dots, w$. Then vv_2 and

vv_3 belong to $C^*(vw)$ and have the same color. By Fact 6.3.4, both $k_0, k_2, v, \ldots, v_2$ and $k_0, k_2, v, \ldots, v_3$ are double AP_6s in the same color, a contradiction to $k_0v_2 \parallel k_1v_3$. So the following corollary holds.

Corollary 6.3.12 *If an edge vw with $v \in V - W$ and $w \in W$ satisfies $C^*(vw) \cap E(V-W) \neq \emptyset$, then vw cannot be the front of an AP_6 (in any 2-coloring of G^*).*

Based on the structural results obtained so far, we propose the following recursive procedure to compute a double AP_6-free 2-coloring of the edges of G.

(*i*) Replace W^1 and W^2 with nonadjacent marker vertices w_1 and w_2.

(*ii*) Compute a double AP_6-free 2-coloring in G_W and in $G_{V-W+w_1+w_2}$.

(*iii*) Construct a 2-coloring of the edges of G by coloring
vw with $v, w \in W$ as in G_W,
vw with $v, w \in V - W$ as in $G_{V-W+w_1+w_2}$,
vw with $v \in V - W$ and $w \in W^1$ as vw_1 in $G_{V-W+w_1+w_2}$ and
vw with $v \in V - W$ and $w \in W^2$ as vw_2 in $G_{V-W+w_1+w_2}$.

(*iv*) Assign to vw the color E_1
if $w \in W^1$ and $v \in W^1$ or
if $w \in W^1$ and $v \in V - W$ and $C^*(vw) \cap E(V-W) = \emptyset$.

(*v*) Assign to vw the color E_2
if $w \in W^2$ and $v \in W^2$ or
if $w \in W^2$ and $v \in V - W$ and $C^*(vw) \cap E(V-W) = \emptyset$.

The next theorem proves that the computed 2-coloring is indeed a double AP_6-free 2-coloring of G^*.

Theorem 6.3.13 *If the 2-coloring of G_W and $G_{V-W+w_1+w_2}$ is double AP_6-free, then (iii), (iv) and (v) construct a double AP_6-free 2-coloring of G.*

Proof. In a fist step, we show that the coloring computed by (*iii*) to (*v*) does not contain an AC_4 in E_1 or E_2. This holds for the coloring computed by (*iii*) because every AC_4 in G has a corresponding AC_4 either in G_W or in $G_{V-W+w_1+w_2}$ with the same colors, so it suffices to consider edges vw in AC_4s that might change their color in (iv) and (v).

If an edge vw has its endpoints in W^1, then $x, y \in W^2$ for every AC_4 $vw \parallel xy$ because W^1 is a clique and $x, y \in W$ by Corollary 6.3.10. So AC_4s with one edge in $E(W)$ are colored properly by (iv) and (v), and it remains to discuss AC_4s with an edge between V and W.

If an edge vw satisfies $v \in V - W$, $w \in W^1$ and $C^*(vw) \cap E(V-W) = \emptyset$, then no AC_4 $vw \parallel xy$ has $x \in W$ and $y \in V - W$, for otherwise v and y would have to be R-vertices, which is impossible because v sees $w \in W$ whereas y misses w. Thus every edge xy in an AC_4 $vw \parallel xy$ satisfies $x \in V - W$ and $y \in W$. Moreover $y \in W^2$ because W^1 is a clique. Therefore vw and xy received their colors in (iv) and (v), respectively, and the AC_4 $vw \parallel xy$ is therefore properly colored. As the case $v \in V - W$, $w \in W^2$ is similar, the coloring constructed in (iii), (iv) and (v) is indeed a 2-coloring of G^*.

To show that the constructed 2-coloring contains no double AP_6, we assume that a double AP_6 $u_0, \ldots, u_5$ exists and show that this assumption leads to a contradiction. Without loss of generality, let $u_0, \ldots, u_5$ be an AP_6 in E_1, thus

$$\begin{array}{l}\{u_0u_2, u_0u_5, u_1u_2, u_1u_5, u_3u_4\} \subseteq E_1, \\ \{u_0u_3, u_0u_4, u_1u_3, u_1u_4, u_2u_5\} \subseteq E_2.\end{array} \tag{6.6}$$

Because of symmetry, it suffices to distinguish the following three cases.

Case 1: $u_2, u_5 \in W$. Then Corollary 6.3.10 implies $u_3, u_4 \in W$. Since $u_2u_5 \in E_2$, not both u_2 and u_5 belong to W^1 because of (iv). Without loss of generality, let $u_2 \in W^2$.

If $u_0 \in V - W$, then $C^*(u_0u_2) \cap E(V - W) \neq \emptyset$, for otherwise $u_0u_2 \in E_1$ by (v). So Lemma 6.3.11 applies to u_0u_2 and u_0u_3 has the same color as u_0u_2, a contradiction to (6.6).

If $u_0 \in W$, then $u_1 \in W$ because of $u_0u_2 \parallel u_1u_3$ and Corollary 6.3.10. Since $u_0u_2, u_1u_2 \in E_1$ and $u_2 \in W^2$, by (v), both u_0 and u_1 must belong to W^2. But this is a contradiction because W^2 is a clique.

Case 2: $u_2 \in W$ *and* $u_5 \in V - W$. From Corollary 6.3.12 follows that $C^*(u_2u_5) \cap E(V - W) = \emptyset$. Therefore $u_3 \in W$ or $u_4 \in W$. Furthermore $u_2 \in W^2$ because of (v).

If $u_0 \in V - W$, then $C^*(u_0u_2) \cap E(V - W) \neq \emptyset$, for otherwise $u_0u_1 \in E_2$ by (v). From Lemma 6.3.11 follows that every edge between

u_0 and a vertex in W has the same color as u_0u_2, thus u_3 and u_4 cannot belong to W. But this contradicts $u_3 \in W$ or $u_4 \in W$.

The same contradiction arises if $u_1 \in V - W$, thus $u_0, u_1 \in W$. Since $u_2 \in W^2$ and $u_0u_2, u_1u_2 \in E_1$, by (v), both u_0 and u_1 must belong to W^1. But this is impossible because W^1 is a clique.

Case 3: $u_2, u_5 \in V - W$. Then $u_3, u_4 \in V - W$, for otherwise we consider the complementary double AP_6 $u_0, u_1, u_4, u_5, u_2, u_3$ and are back in Case 1 or 2. Furthermore $u_0, u_1 \in V - W$ cannot hold, as otherwise the same AP_6 would be contained in $G_{V-W+w_1+w_2}$. So $u_0 \in W$ or $u_1 \in W$. Since $u_0u_2 \parallel u_1u3$ and by Corollary 6.3.10, we may assume that $u_0, u_1 \in W$.

Let $u_0 \in W^1$ and $u_1 \in W^2$ without loss of generality. Then both $C^*(u_1u_2) \cap E(V - W) \neq \emptyset$ and $C^*(u_1u_5) \cap E(V - W) \neq \emptyset$ because otherwise $u_1u_2 \in E_2$ and $u_1u_5 \in E_2$ by (v). Now Lemma 6.3.11 applies to u_1u_2 and u_1u_5, hence $u_0u_2 \in C^*(u_1u_2)$ and $u_0u_5 \in C^*(u_1u_5)$, thus none of the edges of our double AP_6 received its color in (iv) or (v). Therefore $w_1, w_2, u_2, u_3, u_4, u_5$ is a double AP_6 in $G_{V-W+w_1+w_2}$, a contradiction to our assumption. □

The following theorem together with the foregoing theorems establishes the claimed running time to cover G with two threshold graphs.

Theorem 6.3.14 *A double AP_6-free 2-coloring of G^* can be computed in $O(|E|^2)$.*

Proof. The initial computation of G^* and its 2-coloring can be carried out in $O(|E|^2)$. When replacing $W = V(C^*)$ with marker vertices, G^* can be updated by relabeling and deleting vertices of G^*, hence all these updates can also be made in $O(|E|^2)$.

For the search for the double AP_6s, we exploit the fact that a given edge $xy \in E_i$, $i \in \{1, 2\}$, is the front of a double AP_6 if and only if $|C^*(xy)| > 1$ and the set $L = \{v \in V \mid xv \in E - E_i \text{ and } yv \in E - E_i\}$ is not a clique. Observe that the vertices in L can be marked in $O(|V|)$-time. To obtain a nonedge in L, if any, simply build and use the adjacency lists of G_L. The running time for all those searches is therefore $O(|E|^2)$. □

Chapter 7

Cobithreshold graphs

In this chapter, we study the recognition of cobithreshold graphs. HAMMER AND MAHADEV [31] called a graph *cobithreshold* if it is the complement of a bithreshold graph, and they defined a graph to be *bithreshold* if it is the intersection of two threshold graphs and every stable set of the graph is stable in one of the two threshold graphs. Since the complement of a threshold graph is again threshold, we can define cobithreshold graphs as follows.

Definition 7.0.1 *A graph $G = (V, E)$ is* cobithreshold *if it is the union of two threshold graphs T_1 and T_2 such that every clique of G is also a clique of T_1 or T_2.*

The two threshold graphs T_1, T_2 in the above definition are also called a *cobithreshold cover*. Clearly, a cobithreshold cover is a threshold cover of size 2, thus cobithreshold graphs are a subclass of 2-threshold graphs. Besides being 2-threshold, cobithreshold graphs are interesting because of their connection with Boolean functions [53].

In [31], HAMMER AND MAHADEV proposed an $O(|V|^4)$ algorithm for recognizing cobithreshold graphs. In search of faster recognition algorithms, subclasses of cobithreshold graphs were considered. Indeed, HAMMER ET AL. [33] and PETRESCHI AND STERBINI [64, 65] found linear time algorithms for the recognition of bipartite cobithreshold graphs and strict 2-threshold graphs, respectively. In [1], DE AGOSTINO ET AL suggested reducing the recognition problem for cobithreshold graphs to the recognition of bipartite cobithreshold graphs to achieve an $O(|V|^3)$

recognition algorithm. The first substantial improvement for the general case, however, is due to RASCHLE AND STERBINI [68], who found a linear time algorithm for recognizing cobithreshold graphs.

This chapter describes RASCHLE AND STERBINI's approach. The next section contains results on threshold and 2-threshold graphs as far as they go beyond those of Section 6.1 and Section 6.2. In Section 7.2, we describe a new threshold completion algorithm and a new linear algorithm for testing whether a threshold cover is a cobithreshold cover. Those algorithms are needed in Section 7.3 to solve the recognition problem for some special classes of graphs. The general case is then treated in Section 7.4.

7.1 Background and terminology

The following result on threshold graphs is needed for testing in linear time whether a threshold cover is a cobithreshold cover. Let $G = (V, E)$ be a graph and let $\delta_1 < \ldots < \delta_k$ denote the distinct, positive degrees of the vertices with $\delta_0 = 0$ (even if no vertex of degree 0 exists). The *degree partition* of V is then given by $V = D_0 + D_1 + \ldots + D_m$ where D_i is the set of all vertices of degree δ_i.

Theorem 7.1.1 *Let $G = (V, E)$ be a threshold graph with degree partition $V = D_0 + D_1 + \ldots + D_m$. Then a vertex $v \in D_i$ sees a vertex $w \in D_j$ if and only if $i + j > m$.*

Proof. This proof is by induction on the number of vertices in G. If G consists of a single vertex, we are done. For the inductive step, let $D_0 + \cdots + D_m$ denote the degree partition of the graph G before we added the isolated or dominating vertex v_n according to Theorem 6.1.3(iv).

If v_n is isolated, then the new vertex partition is $(D_0 \cup \{v_n\}) + D_1 + \cdots + D_m$. Similarly, if v_n is dominating and G contains a dominating vertex, the new vertex partition is $D_0 + \cdots + D_{m-1} + (D_m + \{v_n\})$. Finally, if v_n is dominating and G contains no dominating vertex, then $D_0 \neq \emptyset$ and the new vertex partition is $D'_0 + D'_1 + \cdots + D'_m + D'_{m+1} + D'_{m+2}$ with $D'_0 = \emptyset$ and $D_{m+2} = \{v_n\}$ and $D'_i = D_{i-1}$ for $i = 1, \ldots, m+1$. In every case, it is easy to see that the theorem holds. □

If a graph is 2-threshold, its conflict graph G^* must be bipartite. It is therefore easy to see that a C_5 and the graphs F_2 and F_3 in Figure 3.1

are not 2-threshold, thus Theorem 3.4.4 implies the following.

Fact 7.1.2 *If a cobithreshold graph G is not split, the complement of an F_2 or a P_4 abcd such that bc belongs to an AC_4 can be found in linear time.*

Let T_1 and T_2 be the two threshold graphs in a 2-threshold cover or in a cobithreshold cover. Since both have the same vertex set V, we usually identify T_1 and T_2 with the corresponding edge sets. Furthermore, we refer to the edges in T_1 and T_2 as the black and red edges, respectively, and call the resulting 2-coloring of the edges of G a *2-threshold coloring* or a *cobithreshold coloring.* Note that, unlike the 2-colorings in the previous chapter, an edge in G can be red and black at the same time, that is, it can be *bicolored.*

On the other hand, in a 2-threshold coloring, no bicolored edge belongs to an AC_4, and the two edges of an AC_4 must receive different colors. We call a 2-coloring of the edges of G that satisfies the above conditions a *proper 2-coloring* of G. Furthermore, we say that a clique is *uniformly colored* if every edge in this clique has the same color. Clearly, every clique in a cobithreshold coloring is uniformly colored.

As in Definition 6.2.6, an AC_6 with all its edges in the same color is called *alternating polygon of length* 5 *or* 6 depending on the number of vertices involved, see Figure 6.4. In the figures of this chapter, red edges are indicated by dotted lines and black edges by bold lines, thus Figure 6.4 shows a red AP_5 and a red AP_6.

Lemma 7.1.3 *A proper 2-coloring can be extended to a cobithreshold coloring in linear time if every clique is colored uniformly and every edge in an AP_5 belongs to an AC_4.*

Proof. Since every clique is uniformly colored, it suffices to show that we can color additional edges such that both the red edges and the black edges are edge sets of threshold graphs. By Theorem 6.2.5, this is equivalent to proving that no AP_5 or AP_6 exists.

In an AP_5 $v_0, \dots, v_5$, the edges v_0v_5 and v_3v_4 are in AC_4s. But our coloring is proper and every clique is uniformly colored, so v_1v_2 must be bicolored because of $v_1v_4 \parallel v_0v_5$ and because of the triangle $\{v_1, v_2, v_4\}$. Hence v_1v_2 is in no AC_4, a contradiction.

In an AP_6 $v_0, \dots, v_5$, the edges v_0v_5, v_1v_2 and v_3v_4 belong to AC_4s. Furthermore, v_0 misses v_2 and v_1 misses v_5, as otherwise the triangles

$\{v_0, v_2, v_5\}$ or $\{v_1, v_2, v_5\}$ would not be uniformly colored. So $v_0v_5 \parallel v_2v_1$, a contradiction as v_0v_5 and v_1v_2 have the same color. □

In the rest of this chapter, we develop an algorithm that computes a cobithreshold coloring whenever the given graph is cobithreshold. To begin with, we give two rules which allow us to infer the color of further edges in a cobithreshold coloring provided that we already know the color of some other edges in that cobithreshold coloring. Those rules follow easily from the fact that no edge in an AC_4 can be bicolored and that every clique is uniformly colored.

Rule 1: *If $vw \parallel xy$, then xy receives the color different from the color of vw.*

Rule 2: *If a clique C contains v and w and vw belongs to an AC_4, then every edge between vertices in C receives the same color as vw.*

In the next section, we show that coloring the edges in the AC_4s suffices to compute a cobithreshold coloring of G.

7.2 Threshold completions

Let $G = (V, E)$ be a graph and $E_1, E_2 \subseteq E$ edge sets that satisfy

$$ab, cd \in E_1 \cup E_2 \text{ for every } AC_4\ ab \parallel cd \text{ in } G. \tag{7.1}$$

We claim that Algorithm 7.1 computes a threshold completion T_1 of E_1 if threshold completions of E_1 and E_2 exist.

If line 13 is never executed, it follows from Theorem 6.1.3(iv) that T_1 is a threshold completion of E_1. So suppose that Algorithm 7.1 stops at line 13. Then G_U contains neither an isolated nor a dominating vertex and either $W = \emptyset$ or $W \neq \emptyset$ and $W_{ud} = \emptyset$. If $W = \emptyset$, then, since G_U is not a threshold graph, an AC_4 $ab \parallel cd$ exists such that both edges belong to E_2, thus E_2 has no threshold completion. Otherwise, if $W \neq \emptyset$ and $W_{ud} = \emptyset$, then G_W has no dominating vertex. But every vertex in G_W is incident to an edge in $E(W) \cap E_1$, hence E_1 has no threshold completion.

Algorithm 7.1 runs in linear time as the number of vertices in W_{ud} is proportional to the number of edges incident to the vertex u chosen in line 15, and u is removed during the next execution of the while loop.

threshold completion

input: a graph $G = (V, E)$ and edge sets $E_1, E_2 \subseteq E$ as in (7.1)
output: a threshold completion T_1 of E_1

$T_1 \leftarrow \emptyset$;
$U \leftarrow V$;
while $U \neq \emptyset$ **do**
  if G_U contains an isolated vertex u **then**
    $U \leftarrow U - \{u\}$
  elsif G_U contains a dominating vertex u **then**
    $T_1 \leftarrow T_1 + \{uv \in E(U)\}$;
    $U \leftarrow U - \{u\}$
  else (* no vertex in G_U is isolated or dominating *)
    $W \leftarrow V(E_1 \cap E(U))$;
    $W_{ud} \leftarrow \{w \in U \mid w \text{ is } W\text{-universal or dominating in } G_W\}$
    if $W = \emptyset$ or $W_{ud} = \emptyset$ **then**
      stop (* E_1 or E_2 has no threshold completion *)
    fi;
    choose $u \in W_{ud}$ with maximal $\deg_{G_U}(u)$;
    $U \leftarrow \{u\} + N_{G_U}(u)$
  fi
od

Algorithm 7.1

The following lemma states that a maximum threshold completion exists and that it is computed by our algorithm.

Lemma 7.2.1 *Let T_1 denote the threshold completion of E_1 computed by Algorithm 7.1. Then every threshold completion T_1' of E_1 satisfies $T_1' \subseteq T_1$.*

Proof. Let T_1' denote an arbitrary threshold completion of E_1. Clearly, the removal of an isolated vertex in G_U does not affect any threshold completion. Similarly, as all edges incident to a dominating vertex in G_U are added to T_1, the removal of a dominating vertex in G_U does not lose any edge relative to T_1'. So assume that G_U contains neither isolated nor dominating vertices and therefore $W \neq \emptyset \neq W_{ud}$.

We claim that u is dominating in $G_{W_{ud}}$. Suppose a vertex $w \in W_{ud}$ exists that does not see u. Then from our definition of W_{ud} follows $u, w \in U - W$. In this case, W is a clique because two nonadjacent vertices $x, y \in W$ would induce $ux \parallel wy$ which contradicts $ux, wy \notin E_1$. Similarly, a vertex x in $N_U(u) - W$ cannot miss a vertex y in W, as otherwise $ux \parallel wy$, again a contradiction to $ux, wy \notin E_1$. So every vertex $v \in W$ is dominating and satisfies $\deg_U(v) > \deg_U(u)$, which is impossible as we have chosen u as the vertex with maximal degree in G_U, thus indeed u is dominating in $G_{W_{ud}}$.

Next we claim that $N[w] \subseteq N[u]$ for every $w \in W_{ud}$. Otherwise, since u sees w, a vertex $y \in N(w) - N[u]$ exists. Similarly, since $\deg_U(u) \geq \deg_U(w)$, a vertex $x \in N(u) - N[w]$ also exists. But $ux, wy \notin E_1$ because u and w belong to W_{ud}, a contradiction to $ux \parallel yw$, which proves our claim. Therefore, no edge vw incident to a vertex v in $U - N[u]$ can belong to T_1', so the removal of $U - N[u]$ in line 16 does not lose any edges relative to T_1'. □

Corollary 7.2.2 *Given a graph $G = (V, E)$ and edge sets E_1, E_2 such that every AC_4 $ab \parallel cd$ satisfies $ab, cd \in E_1 \cup E_2$. Then Algorithm 7.1 applied to E_1 and E_2 computes a cobithreshold cover T_1, T_2 with $E_1 \subseteq T_1$ and $E_2 \subseteq T_2$ if such a cobithreshold cover exists.*

Therefore a cobithreshold cover T_1', T_2' with $E_1 \subseteq T_1'$ and $E_2 \subseteq T_2'$ exists if and only if the two threshold graphs T_1 and T_2 computed by Algorithm 7.1 constitute a cobithreshold cover. To test whether T_1, T_2 is a cobithreshold cover, we have to verify that $E = T_1 \cup T_2$ and that every clique of G is also a clique of T_1 or T_2. The first task is trivial, so it remains to discuss how to perform the second.

Let $D_0, D_1, \ldots, D_m$ denote the degree partition of T_2. By Theorem 7.1.1, a vertex in D_i sees a vertex in D_j if and only if $i + j > m$. We claim that Algorithm 7.2 stops at Line 15 if and only if a clique C of G exists such that precisely one edge of C belongs to $T_1 - T_2$ but C is not a clique of T_1.

Let vw be an edge in $T_1 - T_2$ with $v \in D_i$ and $w \in D_j$ such that $i \leq j$, and let $K = D_k + D_{k+1} + \cdots + D_m$ where $k = m - i + 1$. Since vw does not belong to T_2, we have $i + j \leq m$ and therefore $2i \leq m$, hence $j + k \geq i + k > m$ and $2k > m$. In other words $K \cup \{v, w\}$ is a clique of G and vw is the only edge not in T_2.

Let C denote an arbitrary clique C of G such that vw is the only edge in C that belongs to $T_1 - T_2$. Then every vertex different from v and w in C must belong to K, hence $C \subseteq K + \{v, w\}$. To make sure that

cobithreshold cover test

input: a graph $G = (V, E)$ with 2-threshold cover T_1, T_2 and the degree partition $D_0, D_1, \ldots, D_m$ of T_2

```
(1)  forall v ∈ V do
(2)     a[v] ← m + 1
(3)  od;
(4)  c ← m + 1;
(5)  forall vw ∈ T1 − T2 do
(6)     let v ∈ Di and w ∈ Dj with i ≤ j;
(7)     k ← m − i + 1;
(8)     a[v] ← min{a[v], k};
(9)     a[w] ← min{a[w], k};
(10)    c ← min{c, k};
(11) od;
(12) forall xy ∈ T2 − T1 do
(13)    let x ∈ Dh and y ∈ Dl with h ≥ l;
(14)    if l ≥ c or h ≥ a[y] then
(15)       stop    (* T1, T2 is no cobithreshold cover *)
(16)    fi
(17) od
```

Algorithm 7.2

C has no edge in $T_2 - T_1$, it suffices to verify that every edge between vertices in K and every edge uv and uw with $u \in K$ belongs to T_1.

The edges between vertices in K are precisely those edges xy with $x \in D_h$ and $y \in D_l$ where $h \geq l \geq k$. Similarly, as $k = m - i + 1$ and $i \leq j$, edges between K and v or w are precisely those edges xy with $x \in D_h$ and $y \in D_l$, $h \geq l$, for which $h \geq k$ and $y = v$ or $y = w$.

In the algorithm, the variable c holds the smallest value k for any $vw \in T_1 - T_2$, and $a[u]$ stores the smallest value k for an edge $vw \in T_1 - T_2$ with $u = v$ or $u = w$. Therefore all edges $xy \in T_2 - T_1$ with $x \in D_h$ and $y \in D_l$, $h \geq l$, satisfying $l \geq c$ or $h \geq a[y]$ belong to a clique with precisely one edge in $T_1 - T_2$, thus the algorithm stops at Line 15 if and only if a clique exists with precisely one edge in $T_1 - T_2$ and at least one edge in $T_2 - T_1$.

If we exchange T_1 and T_2 in Algorithm 7.2, the resulting algorithm stops at Line 15 if a clique C of G exists with precisely one edge in $T_2 - T_1$ but C is not a clique of T_2.

If a clique of G is neither in T_1 nor in T_2, then it has at least one edge $vw \in T_1 - T_2$ and another edge xy in $T_2 - T_1$, thus $\{v, w, x, y\}$ is a clique of size 3 or 4 that is not in T_1 or T_2. But a clique of size 3 that is not a clique of T_1 or T_2 has precisely one edge in $T_1 - T_2$ or precisely one edge in $T_2 - T_1$. Therefore either the above algorithm or the algorithm with T_1 and T_2 exchanged stops whenever such a clique exists.

Now assume that every clique of size 3 is a clique of T_1 or T_2. Then a clique of size 4 that is not a clique of T_1 or T_2 has precisely one edge that belongs to $T_1 - T_2$. Therefore Algorithm 7.2 also detects those cliques, and the following lemma holds.

Lemma 7.2.3 *Given a graph G and two edge sets $E_1 \subseteq E$ and $E_2 \subseteq E$ such that $ab, cd \in E_1 \cup E_2$ whenever $ab \parallel cd$ in G. Then there is a linear time algorithm that either computes a cobithreshold cover T_1, T_2 of G with $E_1 \subseteq T_1$ and $E_2 \subseteq T_2$ or decides that no such cobithreshold cover exists.*

7.3 Special classes of cobithreshold graphs

In this section, we show how to recognize some special classes of cobithreshold graphs in linear time. We start with cobithreshold split graphs.

Since a split graph has no C_4, it cannot contain an AP_5; thus, by Lemma 7.1.3, it suffices to find a proper 2-coloring of G that colors every clique uniformly. But every AC_4 in a split graph is a P_4 $abcd$ with $a, d \in S$ and $b, c \in K$. So we may bicolor every edge between vertices in K. Every maximal clique not contained in K can be written as $\{v\} \cup N(v)$ with $v \in S$, hence the color of such a clique can be assigned to the vertex v.

Let $\tilde{S}$ denote the graph with vertex set S such that two vertices a and d are adjacent in $\tilde{S}$ if there is a P_4 $abcd$ in G. Then G is cobithreshold split if and only if $\tilde{S}$ is bipartite. We claim the following.

Lemma 7.3.1 *For a split graph $G = (V, E)$ with $V = S + K$, a spanning forest of $\tilde{S}$ can be computed in linear time.*

We restrict ourselves to graphs in which no two vertices in S have the same neighborhood. If a graph fails to satisfy this property, we generate a new graph G_{subst} by removing copies of such a vertex. A spanning forest of $\tilde{S}$ is readily obtained from a spanning forest of $\tilde{S}_{subst}$ by connecting the copies of a vertex v to one vertex adjacent to v in the spanning forest of $\tilde{S}_{subst}$.

Lemma 7.3.1 follows from the two subsequent Lemmas.

Lemma 7.3.2 *Let $G = (V, E)$ denote a cobithreshold split graph with stable set S and clique K such that no two vertices in S have the same neighborhood. Then at most two vertices have the same degree and $|V|^2 = O(|E|)$.*

Proof. To begin with, we show that at most two vertices in S have the same degree. Every pair of vertices a and d in S with $\deg(a) = \deg(d)$ belongs to a P_4 $abcd$, i.e. a and d are adjacent in $\tilde{S}$. Thus more than two vertices in S with the same degree would induce a triangle in $\tilde{S}$, a contradiction because $\tilde{S}$ must be bipartite.

Let Δ denote the maximal degree of a vertex in S. Then $|K| \geq \Delta$ and, since at most two vertices in S have the same degree, $|S| \leq 2\Delta$; thus $|S| = O(|K|)$ and therefore $|V|^2 = (|S| + |K|)^2 = O(|K|^2)$. But K is a clique, hence $|K|^2 = O(|E|)$ and $|V|^2 = O(|E|)$ as claimed. □

Lemma 7.3.3 *Given a cobithreshold split graph with stable set S and clique K such that no two vertices in S have the same neighborhood. Then a spanning forest of $\tilde{S}$ can be computed in linear time.*

Proof. For every vertex w in K, let $w_{\min}$ denote a vertex with minimal degree in $N(w) \cap S$ and let $w_{\max}$ denote a vertex with maximal degree in $\overline{N}(w) \cap S$. By Lemma 7.3.2, those vertices can be found in linear time. Let F be empty. We scan the vertices v in S and, for every vertex $u \in \overline{N}(v)$ with $\deg(u_{\min}) \leq \deg(v)$, we add an edge $vu_{\min}$ to F. Similarly, for every vertex $w \in N(v)$ with $\deg(w_{\max}) \geq deg(v)$, we add an edge $vw_{\max}$ to F. Again by Lemma 7.3.2, this can be done in linear time. We claim that (S, F) is a spanning forest of $\tilde{S}$.

Note that for each pair of vertices a and d in S with $deg(a) \geq deg(d)$, a P_4 $abcd$ exists if and only there is a vertex w in K such that w sees d and misses a. Therefore all edges $vu_{\min}$ and $vw_{\max}$ belong to $\tilde{S}$. To show that (S, F) is indeed a spanning forest, we suppose the contrary. Then a P_4 $abcd$ exists such that a and d belong to different connected components of (S, F). Without loss of generality, let $\deg(a) \geq \deg(d)$. Now $\deg(c_{\max}) \geq \deg(a) \geq \deg(d) \geq \deg(c_{\min})$ and, by construction, $ac_{\min}$ and $dc_{\max}$ belong to F. But this is a contradiction as $c_{\min}c_{\max}$ also belongs to F. □

Next, we consider cobithreshold graphs that contain a P_4 $abcd$ such that bc belongs to an AC_4.

Lemma 7.3.4 *Let G be a cobithreshold graph and let $abcd$ denote a P_4 in G such that bc belongs to an AC_4. Then a cobithreshold coloring of G can be computed in linear time.*

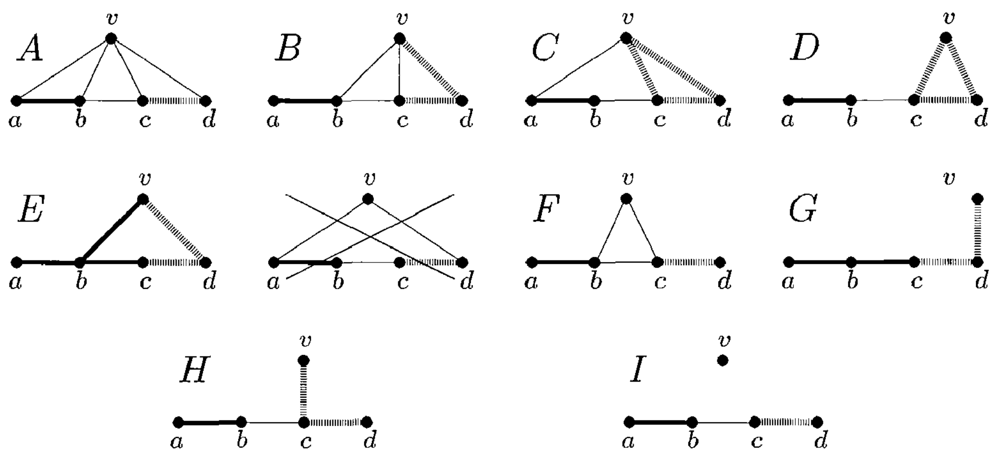


Figure 7.1: *All possibilities of a P_4 together with a fifth vertex v.*

Proof. Up to symmetry, the P_4 $abcd$ together with a fifth vertex v induces one of the graphs depicted in Figure 7.1. Except for the C_5, all graphs $A, B, \ldots, I$ are cobithreshold. We say a vertex v is type $A, B, \ldots, I$ if v and the P_4 $abcd$ induce either the corresponding graph in Figure 7.1 or its symmetric counterpart, e.g. a B-vertex either sees b, c and d and misses a or sees a, b and c and misses d.

Without loss of generality, let ab be black. In the rest of this proof, we give an algorithm for coloring the edges of G based on the color of ab and bc. Since we can execute this algorithm twice, once with bc colored black and another time with bc colored red, by Lemma 7.2.3, we may assume that we know the color of bc.

In Step 1 and 2, we show that the color of ab and bc implies the color of every edge in G that has no endpoint of type I and that this coloring can be found in linear time.

Step 1: *Edges with at least one endpoint in* $\{a, b, c, d\}$. The color of ab and the repeated application of Rule 1 determines the color of a number of edges in Figure 7.1. These edges are indicated by bold lines if they are black and by dotted lines if they are red (e.g. in E, the edges cd and vd are red whereas bc and bv are black because of $ab \parallel cd \parallel vb$ and $ab \parallel vd \parallel cb$).

Furthermore, by Rule 2, the edges in the cliques $\{a, b, v\}$, $\{b, c, v\}$ and $\{c, d, v\}$ receive the same color as ab, bc and cd respectively, and the color of av in C is determined by Rule 1 because of $bc \parallel va$. Thus all edges with at least one endpoint in $\{a, b, c, d\}$ are colored.

Step 2: *Edges between vertices in* $V - \{a, b, c, d\}$ *except for edges incident to type I vertices.* Let vw denote such an edge, i.e. $v, w \in V - \{a, b, c, d\}$ and neither v nor w is type I. Depending on the neighborhood of v and w relative to b and c, we distinguish the following cases.

Case 1: v or w is $\{b, c\}$-partial. Without loss of generality (symmetry), suppose that v sees b but misses c. Then v is type C, D, E or H, hence vb belongs to an AC_4. If w misses b, then $vw \parallel cb$ and vw can be colored according to Rule 1. Otherwise, if w sees b, then $\{b, v, w\}$ is a clique and, by Rule 2, vw receives the same color as vb.

Case 2: v and w are $\{b, c\}$-universal. Then $\{b, c, v, w\}$ is a clique, Rule 2 applies and vw receives the same color as bc.

Case 3: v and w are $\{b, c\}$-null. Then $vw \parallel bc$, hence vw can be colored by Rule 1.

Case 4: v is $\{b, c\}$-universal and w is $\{b, c\}$-null or vice versa. Because of symmetry, it suffices to discuss the former case. So v is type A, B or F and w is type G. Furthermore, suppose that w sees d (the symmetric case is similar). If v misses d, then $vw \parallel dc$ and vw is black. Otherwise, if v sees d, then $\{d, v, w\}$ is a clique and, by Rule 2, vw may be colored in the same way as wd.

Step 2 again takes linear time if we precompute which of the above cases applies to which graph in Figure 7.1 (or its symmetric counterpart).

After the completion of Step 2, only edges incident to type I vertices are not colored. Furthermore, in the rest or this proof, we only use the fact that bc is colored, not the assumption that bc belongs to an AC_4.

Observation 1: The set of all type I vertices is stable. An edge v_1v_2 between two type I vertices satisfies $ab \parallel v_1v_2$ and $cd \parallel v_1v_2$, thus a third color would be required.

Observation 2: The set of all type A vertices is a clique and every edge between type A vertices is bicolored. Let w_1 and w_2 be two type A vertices. Because of the clique $\{a, b, w_1\}$ and $\{c, d, w_1\}$, the edge bw_1 is black and cw_1 is red. If w_1 misses w_2, then $bw_1 \parallel dw_2$ and $cw_1 \parallel aw_2$, so bw_1 cannot be red and cw_1 cannot be black, a contradiction because of the clique $\{b, c, w_1\}$. Thus w_1 and w_2 are adjacent and, by Rule 2 applied to the cliques $\{a, b, w_1, w_2\}$ and $\{c, d, w_1, w_2\}$, the edge w_1w_2 must be bicolored.

Step 3: *Edges vw between type I vertices v and vertices w of type $B, \ldots, H$.* It is easy to verify, see Figure 7.1, that a P_3 wxy exists with $x, y \in \{a, b, c, d\}$. Clearly, this P_3 can be extended to a P_4 $vwxy$, thus the color of vw follows from applying Rule 1. Since the color of vw solely depends on the color of ab, bc, cd and on the type of w (or its symmetric counterpart), Step 3 can be carried out in linear time.

For the remaining steps, we need some further precomputation.

- For each type I vertex v, let $n(v)$ denote a vertex of type $B, \ldots, H$ that sees v, or let $n(v) = 0$ if no such vertex exists.
- For each type A vertex w, let $m(w)$ denote a vertex of type $A, \ldots, H$ that misses w, or let $m(w) = 0$ if no such vertex exists.

Obviously, the values $n(v)$ and $n(w)$ can be computed in linear time.

Step 4: *Edges incident to type I vertices v which see some type A vertices w with $m(w) \neq 0$.* Since $m(w)$ is of type $A, \ldots, H$, it sees a vertex $x \in \{a, b, c, d\}$. But $xm(w)$ is already colored and $vw \parallel xm(w)$, hence the color of vw can be determined by Rule 1. Furthermore, every type A vertex z that sees v but satisfies $m(z) = 0$ sees every vertex w, hence $\{v, w, z\}$ is a clique and vz receives the same color as vw. Since edges between v and a vertex of type $B, \ldots, H$ were colored in Step 3, the color of every edge incident to v is determined, and it should be clear that Step 4 can be carried out in linear time.

Step 5: *Edges incident to type I vertices v with $n(v) \neq 0$.* Because of Step 3, it again suffices to color edges between v and type A vertices w. Moreover, we may assume that $m(w) = 0$, as otherwise the edges incident to v were colored in Step 4. Therefore w sees $n(v)$ and $\{v, w, n(v)\}$ is a clique. From the argumentation in Step 3 follows that $vn(v)$ belongs to an AC_4, so Rule 2 applies and vw must be colored as $vn(v)$. So Step 5 is linear.

Now let S denote the set of all type I vertices v that

(i) are not adjacent to a type A vertex w with $m(w) \neq 0$ and

(ii) satisfy $n(v) = 0$.

Note that the uncolored edges are precisely those edges incident to a vertex in S. Furthermore, let $K = N(S)$. Because of (ii), every vertex in K is of type A and, because of (i), every vertex w in K misses only vertices of type I.

The next Step again requires some precomputation. Let X denote the set of all type I vertices x that see a vertex $r(x) \notin K$. Furthermore, for every vertex w in K, let $s(w)$ denote a vertex in X that misses w, or $s(w) = 0$ if no such vertex exists. Clearly $r(x)$ and $s(w)$ can be computed in linear time.

Step 6: *Edges in AC_4s that are not entirely in G_{S+K}.* For every vertex v in S that sees a vertex $w \in K$ with $s(w) \neq 0$, a P_4 $vwr(s(w))s(w)$ exists. Since $r(s(w)) \notin K$, the edge $r(s(w))s(w)$ was colored in Step 1 to 5, so Rule 1 determines the color of vw. Furthermore, as $\{v\} \cup N(v)$ is a clique, every edge incident to v receives the same color as vw. Clearly, this can be achieved in linear time.

Now let $vw \parallel xy$ be an AC_4 with $v \in S$ but not entirely in G_{S+K}. Then y cannot belong to S. On the other hand, as w is type A and $m(w) = 0$, the vertex y must be of a type I. Therefore either $n(y) \neq 0$ or a type A vertex u adjacent to y satisfies $m(u) \neq 0$. In both cases, y sees a vertex $z \notin K$ and a P_4 $vwzy$ exists, so $y \in X$ and $s(w) \neq 0$, i.e. every edge incident to v and therefore every edge in an AC_4 is colored by the above procedure.

Step 7: Regarding the AC_4s in G_{S+K}, we compute a spanning forest of $\tilde{S}$ as described in Lemma 7.3.1. Then we color the connected components of $\tilde{S}$ in accordance with the colored edges in Step 1 to 6, and we end up with a proper 2-coloring of the edges in G.

We claim that this 2-coloring can be completed to a cobithreshold coloring of G. Let $R \subseteq S$ denote the set of vertices in S that belong to components whose vertices are incident to no edge colored in Step 1 to 6. Since the color of every edge in G_{V-R} is implied from the color of *ab* and *bc* by Rule 1 and 2, the coloring of G_{V-R} can be completed to a cobithreshold coloring, c.f. Lemma 7.2.3. To apply Lemma 7.1.3, it suffices to show that every clique is uniformly colored and that no AP_5 exists.

The latter is easy as the neighborhood of no vertex in an AP_5 is a clique, so no vertex in R belongs to an AP_5, thus every AP_5 is in G_{V-R}, which is impossible because of the cobithreshold coloring of G_{V-R}. For the same reason, every clique in G_{V-R} is uniformly colored. But every maximal clique of G that is not entirely in G_{V-R} can be written as $\{v\} \cup N(v)$ with $v \in R$, hence it is uniformly colored because of Step 7 and because edges between vertices in $N(v)$ are bicolored. □

A similar result holds if a P_4 *abcd* is known together with two non-adjacent type B vertices, one adjacent to a and the other adjacent to d, as depicted in Figure 7.2. We call this graph the *bridge abcdef*.

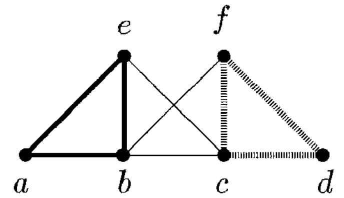


Figure 7.2: *The bridge abcdef.*

Lemma 7.3.5 *Let G be a cobithreshold graph that contains a bridge abcdef. Then a cobithreshold coloring of G can be computed in linear time.*

Proof. Again assume that *ab* is black. In Figure 7.2, the edges colored by repeatedly applying Rule 1 are indicated by bold and dotted lines, respectively. Moreover, as $\{b, c, e\}$ and $\{b, c, f\}$ are cliques, the edges bc, ce are black and bc, bf are red. Note that bc receives both colors, hence it cannot belong to an AC_4.

In order to color the remaining edges, we proceed as in the proof of Lemma 7.3.4, i.e. we consider the type $A \dots I$ vertices relative to the P_4 *abcd*. This time, however, no type C, E and G vertices exist as otherwise bc would belong to an AC_4.

As in the proof of the previous lemma, the color of the edge *ab* implies the color of every edge in G that has no endpoint of type I and this coloring can be computed in linear time.

Step 1: *Edges with at least one endpoint in* $\{a, b, c, d\}$. We have already seen how the edges between $\{a, b, c, d, e, f\}$ must be colored. Furthermore, the application of Rule 1 colors some edges as shown in Figure 7.1. So it remains to discuss the edges incident to a type A, B and F vertex v. If v is type A, then, by Rule 2, the edges in the cliques $\{a, b, v\}$ and $\{c, d, v\}$ receive the same color as ab and cd, respectively.

Now suppose v is type B or F. Because of symmetry, we may assume that v misses a. If v sees e or f, then $\{b, c, e, v\}$ or $\{b, c, f, v\}$ is a clique, hence Rule 2 implies the color of the edges bv and cv. If v misses both e and f, then $bv \parallel df$ and $cv \parallel ae$, so the color of bv and cv is determined by Rule 1.

Step 2: *Edges between vertices in* $V - \{a, b, c, d\}$ *except for edges incident to type* I *vertices.* Let vw denote such an edge, so $v, w \in V - \{a, b, c, d\}$ and neither v nor w is type I. Depending on the neighborhood of v and w relative to b and c, we distinguish the following cases.

Case 1: v *or* w *is* $\{b, c\}$*-partial.* Without loss of generality (symmetry), suppose that v sees b but misses c. Then v is type D or H, hence vb belongs to an AC_4. If w misses b, then $vw \parallel cb$, a contradiction to our assumption that bc is both black and red. Otherwise, if w sees b, then $\{b, v, w\}$ is a clique and, by Rule 2, vw receives the same color as vb.

Case 2: v *and* w *are* $\{b, c\}$*-universal.* If e sees both v and w, then $\{b, e, v, w\}$ is a clique and, by Rule 2, vw receives the same color as be. If e misses v or w, then $ae \parallel vc$ or $ae \parallel cw$. But $\{c, v, w\}$ is a clique, thus vw must be red by Rule 2.

Case 3: v *or* w *is* $\{b, c\}$*-null.* This case is impossible because v and w are neither type G nor type I.

The remaining Steps are identical to those in the proof of Lemma 7.3.4 and therefore omitted. □

7.4 Recognizing cobithreshold graphs

In this section, we give a recursive algorithm for coloring the edges in a cobithreshold graph. For every graph $G = (V, E)$, let $G' = (V', E')$ denote the prime graph that arises from substituting marker vertices for maximal homogeneous sets of G and, for every vertex $v \in V'$, let $H(v)$ stand for the corresponding module in G.

cobithreshold recognition

(1) **if** G has an isolated vertex v **then**
(2)   recurse on $G - \{v\}$
(3) **elsif** G has a dominating vertex v **then**
(4)   bicolor the edges incident to v;
(5)   recurse on $G - \{v\}$
(6) **elsif** G is disconnected **then**
(7)   color G as described in Lemma 7.4.1;
(8) **elsif** $\overline{G}$ is disconnected **then**
(9)   color G as described in Lemma 7.4.2;
(10) **elsif** G' is not a split graph **then**
(11)   find a bridge or a P_4 $abcd$ with bc in an AC_4;
(12)   color G according to Lemma 7.3.4 or Lemma 7.3.5
(13) **else** (* G' is a split graph *)
(14)   let S and K be the vertex sets as defined in Lemma 7.4.3
(15)   **if** $V' = S + K + v$ and $H(v)$ is not a threshold graph **then**
(16)   recurse on $G_{H(v)}$;
(17)   color the edges not in $G_{H(v)}$ as described in Lemma 7.4.4
(18)   **else** (* every homogeneous set induces a threshold graph *)
(19)   **if** $H(K)$ is not a clique **then**
(20)   color G as described in Lemma 7.4.5
(21)   **else** (* $H(K)$ is a clique *)
(22)   color G as described in Lemma 7.4.6

Algorithm 7.3

Lines (1) to (5) of Algorithm 7.3 are correct as an isolated or dominating vertex can always be added to a threshold graph, see Theorem 6.1.3(*iv*), hence those vertices may be added to both threshold graphs T_1 and T_2 that constitute a cobithreshold cover of $G - \{v\}$.

The following Lemma discusses *Lines (6) and (7)*.

Lemma 7.4.1 *A cobithreshold graph G without isolated vertices is disconnected if and only if G is the disjoint union of two nontrivial connected threshold graphs G_1 and G_2.*

Proof. Obviously the disjoint union of two threshold graphs is a cobithreshold graph. Conversely, since G has no isolated vertices, G

consists of nontrivial connected components $G_1, G_2, \ldots, G_k$ with $k \geq 2$. But every pair of edges in different connected components induces a $2K_2$, thus we can color at most two nontrivial connected components G_1 and G_2. Moreover, every edge in such a component receives the same color, hence G_1 and G_2 are threshold graphs. □

Now suppose that $\overline{G}$ is disconnected but has no dominating vertex. Recall that the join $G_1 \oplus G_2$ of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the graph $(V_1 + V_2, E_1 + E_2 + E_{12})$, where E_{12} is the set of edges between vertices in V_1 and vertices in V_2.

Lemma 7.4.2 *A cobithreshold graph $G = (V, E)$ without dominating vertices is codisconnected if and only if G is the join of two nontrivial coconnected graphs G_1 and G_2 such that*

(*i*) *G_1 is the complement of a complete bipartite graph with bipartition (V_1^1, V_1^2) and*

(*ii*) *G_2 is a threshold graph.*

Moreover, $G_{V_1^1+V_2}$ and $G_{V_1^2+V_2}$ constitute a cobithreshold cover of G.

Proof. If is easy to verify that $G_{V_1^1+V_2}$ and $G_{V_1^2+V_2}$ constitute a cobithreshold cover of the join of G_1 and G_2. Conversely, as G has no dominating vertices, G must be the join of nontrivial coconnected graphs $G_1, G_2, \ldots, G_k$. Furthermore, every edge between different coconnected graphs G_i and G_j belongs to a C_4, hence such an edge receives precisely one color.

To begin with, we show that G is the join of two coconnected graphs G_1 and G_2. If $k \geq 3$, we can choose pairs of nonadjacent vertices a_1, b_1 and a_2, b_2 in G_1 and G_2, respectively, and a single vertex v in G_3. Without loss of generality, let a_1a_2 be black; hence, by Rule 1, b_1b_2 is red. Since $\{a_1, a_2, v\}$ and $\{b_1, b_2, v\}$ are cliques, by Rule 2, a_1v is black and b_2v is red. But this is impossible because of the clique $\{a_1, b_2, v\}$.

Next, we prove that at least one of the two graphs G_1 and G_2 is a threshold graph. Otherwise AC_4s $a_1b_1 \parallel c_1d_1$ in G_1 and $a_2b_2 \parallel c_2d_2$ in G_2 would exist. But a_1b_1 and c_1d_1 have different colors; hence Rule 2 applies to the cliques $\{a_1, b_1, a_2, b_2\}$ and $\{c_1, d_1, a_2, b_2\}$, and therefore a_2b_2 is bicolored, a contradiction because $a_2b_2 \parallel c_2d_2$.

Note that not both $\overline{G_1}$ and $\overline{G_2}$ can contain a triangle, as triangles a_1, b_1, c_1 in $\overline{G_1}$ and a_2, b_2, c_2 in $\overline{G_2}$ imply that a_1a_2 is bicolored because of $a_1a_2 \parallel b_1b_2 \parallel c_1c_2 \parallel a_1a_2$. If both G_1 and G_2 are threshold graphs, let G_1 denote that graph whose complement has no triangle. Since G_1 is threshold and coconnected, it contains an isolated vertex. But $\overline{G_1}$ has no triangle, so all other vertices in G_1 induce a clique and G_1 is the complement of a complete bipartite graph as claimed.

Now suppose that G_1 is no threshold graph. Then G_1 contains no P_4 because a P_4 $a_1b_1c_1d_1$ in G_1 implies that, for any vertex v in G_2, the edges b_1v and c_1v have different colors because of the cliques $\{a_1, b_1, v\}$ and $\{c_1, d_1, v\}$, which is impossible because of the clique $\{b_1, c_1, v\}$. But a P_4-free graph is either disconnected or codisconnected; hence G_1 is disconnected.

Furthermore, no vertex v in G_1 can miss a and c in an AC_4 $ab \parallel cd$ in G_1, as otherwise $ax \parallel vy \parallel cx$ holds for every pair of nonadjacent vertices x, y in G_2, a contradiction to the fact that ax and cx have different colors because of the cliques $\{a, b, x\}$ and $\{c, d, x\}$. Therefore G_1 has no isolated vertices and, by Lemma 7.4.1, G_1 consists of two nontrivial connected threshold graphs T_1 and T_2.

If T_1 were no clique, it would contain a vertex v and an edge ab such that v sees b and misses a, a contradiction because $ab \parallel cd$ for every edge cd in T_2 but no vertex in T_1 can miss a and c in an AC_4 $ab \parallel cd$. Since the same reasoning holds for T_2, both T_1 and T_2 are cliques, hence $\overline{G_1}$ is complete bipartite. □

If G' is no split graph, by Fact 7.1.2, a P_4 $abcd$ with bc in an AC_4 or the complement of an F_2 can be found in linear time. Since the corresponding graphs also exist in G and the complement of an F_2 is a bridge, we can indeed color the edges as described in Lemma 7.3.4 and Lemma 7.3.5.

It remains to show how to color connected cobithreshold graphs G that are also coconnected and whose associated prime graph G' is split. As G' is also connected and coconnected, Lemma 3.1.1 implies that G' contains a P_4, hence its conflict graph contains a nontrivial connected component C^*. The next lemma exhibits the structure of G'.

Lemma 7.4.3 *Let $G = (V, E)$ be a prime cobithreshold split graph and let $C^* \subseteq E$ be a nontrivial connected component of its conflict graph. Furthermore let S denote the stable set and K the clique in the induced split graph $G_{V(C^*)}$. Then either*

(i) $V = S + K$ or

(ii) $V = S + K + v$ *and* $N(v) = K$.

In part, the above lemma can be derived from Theorem 6.3.7. Nevertheless, we give a full proof in order to make this chapter independent of Section 6.3.

Proof. Let $H = V - K - S$. Since $G_{V(C^*)}$ is a split graph, every AC_4 $ab \parallel cd$ with edges in C^* is a P_4 $abcd$. If a vertex $v \in H$ sees a but misses b, then either $ab \parallel cv$ or $av \parallel dc$, in both cases a contradiction to $v \in H$. Thus every $\{a,b\}$-partial vertex in H sees b and misses a. Moreover, an $\{a,b\}$-partial vertex v in H is also $\{c,d\}$-partial, for otherwise either $abvd$ or $vbcd$ is a P_4, again a contradiction to $v \in H$.

Therefore an $\{a,b,c,d\}$-partial vertex in H sees b and c and misses a and d. By induction, this hold for every P_4 with a wing in C^*. But every $V(C^*)$-partial vertex is $\{a,b,c,d\}$-partial for at least one P_4 $abcd$, therefore a $V(C^*)$-partial vertex sees every vertex in K and misses every vertex in S.

Furthermore, an edge between a $V(C^*)$-partial vertex v and a $V(C^*)$-null vertex q implies a P_4 $qvba$, a contradiction to $v \in H$. Similarly, a $V(C^*)$-partial vertex v sees every a $V(C^*)$-universal vertex p, as otherwise, if v misses p, the graph $G_{\{a,b,c,d,v,p\}}$ is split with clique $\{b,c,p\}$ and stable set $\{a,d,v\}$ such that no two vertices in $\{a,d,v\}$ have the same neighborhood but three vertices have the same degree, so G would not be cobithreshold as shown in the proof of Lemma 7.3.2.

But now the union of $V(C^*)$ and all $V(C^*)$-partial vertices is a module and G is prime, so every vertex in $V - V(C^*)$ must be $V(C^*)$-partial. In this case, however, the set of all $V(C^*)$-partial vertices is a module; thus at most one vertex v can be $V(C^*)$-partial, i.e. $H = \{v\}$ and therefore $N(v) = K$ as claimed. □

In the rest of this section, let $V' = S + K$ or $V' = S + K + v$ as described in Lemma 7.4.3. Let $ab \parallel cd$ be an AC_4 in G'. Obviously, every edge in a maximal homogeneous set that corresponds to a or b receives the same color as ab, i.e. the corresponding graph is threshold.

But every vertex in $S + K$ belongs to a P_4, hence every maximal homogeneous set that corresponds to a vertex in $S + K$ is a threshold graph. Thus, if a maximal homogeneous set is not threshold, then it must be $H(v)$.

Lemma 7.4.4 *If $H(v)$ contains an AC_4, then every cobithreshold coloring of $G_{H(v)}$ together with the coloring arising from*

(a) bicoloring edges with one endpoint in $H(K)$ and the other in $H(K+v)$ and

(b) coloring edges incident to $H(S)$ as the corresponding vertex in an $\tilde{S}$-coloring

can be extended to a cobithreshold coloring of G.

Proof. From the previous discussion follows that an AC_4 $uv \parallel xy$ in $H(v)$ exists. Suppose $H(b), b \in K$, is no clique and let $abcd$ a P_4 in G'. Then $a_1b_1 \parallel b_2c_1$ for any choice $a_1 \in H(a)$, $c_1 \in H(c)$ and $b_1, b_2 \in H(b)$ such that b_1 misses b_2. But this is a contradiction because both $\{a_1, b_1, u, v\}$ and $\{a_1, b_1, x, y\}$ are cliques. Similarly, suppose $H(a), a \in S$, is not stable. Then any pair of adjacent vertices $a_1, a_2 \in H_a$ implies $a_1a_2 \parallel uv$ and $a_1a_2 \parallel xy$, again a contradiction as uv and xy have different colors.

Hence $H(K)$ is a clique and $H(S)$ is a stable set, thus every AC_4 in G is either in $G_{H(v)}$ or in $G_{H(S+K)}$ and therefore our coloring is a proper 2-coloring of G. Furthermore, it is easy to verify that every maximal clique of G is uniformly colored. Finally, an AP_5 has no vertex in $H(S)$ as the neighborhood of vertex in an AP_5 is not a clique. Similarly, an AP_5 has no vertex in $H(K)$, as no vertex in an AP_5 is dominating. Thus every AP_5 is in $G_{H(v)}$ but $G_{H(v)}$ has no AP_5 because of its cobithreshold coloring. The claim of our Lemma now follows from Lemma 7.1.3. □

It remains to discuss *Lines (18) to (22)* of Algorithm 7.3. Therefore we may assume that every maximal homogeneous set of G induces a threshold graph.

Lemma 7.4.5 *If $H(v)$ is a threshold graph and $H(b)$ is not a clique for a vertex $b \in K$, then a cobithreshold coloring of G can be found in linear time.*

Proof. Since every vertex in $S \cup K$ belongs to a P_4, we may assume that a P_4 $abcd$ exists. For any pair of nonadjacent vertices $b_1, b_2 \in H(b)$ and any choice of $a_1 \in H(a), c_1 \in H(c), d_1 \in H(d)$, the P_4 $a_1b_1c_1d_1$ is a P_4 with b_1c_1 in a C_4. By Lemma 7.3.4, this P_4 can be used to compute a cobithreshold coloring of G in linear time. Furthermore, it is straightforward to find such a P_4 in linear time as $O(|V'|^2) = O(|E'|)$ by Lemma 7.3.2. □

In the next lemma, we color cobithreshold graphs corresponding to *Line (22)* in Algorithm 7.3.

Lemma 7.4.6 *If $H(v)$ is a threshold graph and $H(K)$ a clique, then a cobithreshold coloring of G can be found in linear time.*

Proof. Let $S' = \{s \in S \mid H(s) \text{ is not stable}\}$. Then edges in G incident to $H(s), s \in S$, receive the color of $s \in \tilde{S}$. Compute a 2-coloring of $\tilde{S}$ and let S'_{red} and S'_{black} denote the vertices in S' colored red and black in $\tilde{S}$, respectively. Then every edge between vertices in $\bigcup_{x\in\overline{N}(S'_{red})} H(x)$ must be colored black, and every edge between vertices in $\bigcup_{x\in\overline{N}(S'_{black})} H(x)$ must be colored red. This coloring of the edges in G can be found in linear time as the corresponding coloring of the vertices in G' can be found in linear time because of $O(|V'|^2 = O(|E'|)$ by Lemma 7.3.2.

It is easy to verify that every AC_4 in G is colored. Moreover, as the coloring of $\tilde{S}$ is unique and the remaining edges are colored by Rule 1, this coloring admits a cobithreshold cover that can be computed in linear time, c.f. Lemma 7.2.3. □

As to the complexity of Algorithm 7.3, we rely on the modular decomposition of the graph. The modular decomposition of an arbitrary graph is computed in linear time, see [17, 54, 21]. It also provides the so-called modular decomposition tree and, at each level, the corresponding prime graph.

Given the modular decomposition, *Lines (1), (3), (6)* and *(8)* can be executed in constant time by inspecting the corresponding node in the modular decomposition tree. The test whether G' is a split graph can be carried out in $O(|V'| + |E'|)$, see [27]. With a split partition of V', the computation of S and K in *Line (14)* is in $O(|V'| + |E'|)$.

Finally, the computation of *Line (15)* and *(19)* can be done in constant time per node in the decomposition tree given we have precomputed the type of modules contained in the subtree of a node, i.e. whether the corresponding graph is threshold. This precompilation can be done in linear time bottom up from the leaves of the modular decomposition tree. Thus, the overall running time of Algorithm 7.3 is linear.

Chapter 8

Conclusions

In the previous chapters, we presented several new algorithms to recognize classes of perfectly orderable graphs. Most of these algorithms are based on results obtained from the generalization of GALLAI's modular decomposition. In fact, we believe that our extension of GALLAI's theory is the main contribution of this thesis to algorithmic graph theory.

Many problems, however, had to remain unsolved. In the following, we give a brief overview of further directions of research related to our work.

- In Chapter 4, we introduced k-modules as generalizations of modules and then focused on 2-modules. As it turned out, special 2-modules can be used to obtain new unique decompositions for arbitrary graphs. We wonder whether other k-modules can also be specialized such that they imply unique graph decompositions and, if so, whether those decompositions can be applied to recognize further classes of perfectly orderable graphs.

- A key theorem in GALLAIs work on comparability graphs states that different P_3-classes cover different vertex sets. Since the cover of a P_3-class is a module, comparability graphs can be oriented by substituting marker vertices for modules, either explicitly or implicitly.

  In Chapter 5, we showed that a similar theorem holds for P_4-components. Since the cover of a P_4-component is a strict split module, P_4-comparability graphs can also be oriented by substituting marker vertices for strict split modules, again explicitly or implicitly.

It is an open problem, however, whether $2K_2$-components and bipartite modules, perhaps in conjunction of P_4-components, can be applied to recognize other more general classes of perfectly orderable graphs. Theorem 6.3.7 indicates that this might well be.

- Yet another open problem related to our work is the question whether there is a polynomial algorithm for recognizing graphs with quasi threshold dimension two[1], that is, graphs which are the intersection of two graphs without induced P_4 and C_4. Since the complement of a graph with quasi threshold dimension two is the union of two graphs without induced P_4 and $2K_2$, it is necessary that its edges can be 2-colored such that the two edges in a $2K_2$ and the two wings of a P_4 have different colors. So the question naturally arises whether this condition is also sufficient.

  With the results presented in this thesis, it is not difficult to see that such a graph can be decomposed into graphs with unique 2-colorings with respect to the edges in $2K_2$s and the wings in P_4s. By applying substitution, it can also be shown that the problem were solved if a polynomial algorithm for recognizing graphs with unique 2-colorings exists. To date, however, such an algorithm is not known.

[1] Quasi threshold graphs are also called trivially perfect in [26] or arborescence-comparability graphs in [22].

Appendix A

List of Symbols

Set Theory

$\forall x$	For *all* x.
$\exists y$	There *exists* a y.
$x \in X$	x is a member of X.
$A \subseteq X$	A is a *subset* of X.
$B \subset X$	B is *proper subset* of X.
$\|X\|$	The cardinality of a set X.
$A \cap B$	The *intersection* of A and B.
$A \cup B$	The *union* of A and B.
$A + B$	The *union* of *disjoint* sets A and B.
$A - B$	The *difference* set A minus B.
$\emptyset$	The *empty* set.

Graph Theory

$G = (V, E)$	The *undirected graph* G with vertex set V and edge set E.
$G = (V_1, V_2, E)$	The *split graph* G with vertex set $V_1 + V_2$ and edge set E where V_1 is a clique and V_2 a stable set.

$\vec{G} = (V, \vec{E})$	An *orientation* of the graph $G = (V, E)$.
vw	The undirected edge between v and w.
$v \rightarrow w$	The directed edge from v to w.
$\overline{G} = (V, \overline{E})$	The complement of $G = (V, E)$.
$G_W = (W, E(W))$	The subgraph of G *induced* by the vertex set W.
$(V(F), F)$	The subgraph of G *spanned* by the edge set F.
K_n	The *complete* graph with n vertices.
mK_n	The disjoint union of m copies of the K_n.
C_k	The chordless cycle on k vertices.
P_k	The chordless path on k vertices.
AC_{2k}	The alternating cycle on $2k$ vertices.
$N(v)$	The *neighborhood* of a vertex v in G.
$\overline{N}(v)$	The *non-neighborhood* of a vertex v in G.
$N(W)$	The *neighborhood* of a vertex set W.
$N[v]$	The *closed neighborhood* of a vertex v.
$\omega(G)$	The *clique number* of G.
$k(G)$	The *clique cover number* of G.
$\alpha(G)$	The *stability number* of G.
$\chi(G)$	The *chromatic number* of G.
$t(G)$	The *threshold dimension* of G.
$G_1 \cup G_2$	The *union* of G_1 and G_2.
$G_1 + G_2$	The *disjoint union* of G_1 and G_2.
$G_1 \oplus G_2$	The *join* of G_1 and G_2.
$abcd \sim a'b'c'd'$	$abcd$ and $a'b'c'd'$ are strong-adjacent P_4s.
$ab \parallel cd$	The sequence c, a, b, d is an AC_4.

List of Figures

Bibliography

[1] S. De Agostino, R. Petreschi, and A. Sterbini. An $O(n^3)$ recognition algorithm for bithreshold graphs. *Algorithmica*, 17:416–425, 1997.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms.* Addison-Wesley, Reading MA, 1974.

[3] S. R. Arikati and U. N. Peled. A polynomial algorithm for the parity path problem on perfectly orientable graphs. *Discrete Applied Mathematics*, 65:5–20, 1996.

[4] S. Arora and S. Safra. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996.

[5] L. Babel and S. Olariu. On the structure of P_4-connected graphs and the separable-homogeneous decomposition. In *WG'97*, volume 1335 of *Lecture Notes in Computer Science*, pages 25–36, 1997.

[6] C. Berge. Färbung von Graphen deren sämtliche bzw. deren ungerade Kreise starr sind. *Wissenschafliche Zeitschrift der Martin-Luther-Universität, Halle-Wittenberg, Mathematisch-naturwissenschaftliche Reihe*, 114:114–115, 1961.

[7] C. Berge. Sur une conjecture relative au problème des codes optimaux. In *Communications de 13ième assemblée générale d'URSI*, Tokio, 1962.

[8] D. Bienstock. On the complexity of testing for odd holes and induced odd paths. *Discrete Mathematics*, 90:85–92, 1991.

[9] J. A. Bondy and U. S. R. Murty. *Graph theory with applications.* North-Holland, Amsterdam, 1976.

[10] V. Chvátal. Perfectly ordered graphs. *Annals of Discrete Mathematics*, 21:63–65, 1984.

[11] V. Chvátal. A semi-strong perfect graph conjecture. *Annals of Discrete Mathematics*, 21:279–280, 1984.

[12] V. Chvátal. On the P_4-structure of perfect graphs III: Partner decompositions. *Journal of Combinatorial Theory, Series B*, 43:349–353, 1987.

[13] V. Chvátal and P. L. Hammer. Set-packing problems and threshold graphs. In *CORR*, volume 73-21, University of Waterloo, Canada, 1973.

[14] V. Chvátal and P. L. Hammer. Aggregation of inequalities in integer programming. *Annals of Discrete Mathematics*, 1:145–162, 1977.

[15] V. Chvátal, C. T. Hoàng, N. V. R. Mahadev, and D. DeWerra. Four classes of perfectly orderable graphs. *Journal of Graph Theory*, 11:481–495, 1987.

[16] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progression. In *19th Annual ACM Symposium on the Theory of Computing*, pages 1–6, 1987.

[17] A. Cournier and M. Habib. A new linear time algorithm for modular decomposition. *LNCS*, 787:68–84, 1994.

[18] M. B. Cozzens and R. Leibowitz. Threshold dimension of graphs. *SIAM Journal on Algebraic and Discrete Methods*, 5:579–595, 1984.

[19] M. B. Cozzens and R. Leibowitz. Multidimensional scaling and threshold graphs. *Journal of Mathematical Psychology*, 31:179–191, 1987.

[20] M. B. Cozzens and F. S. Roberts. On dimensional properties of graphs. *Graphs and Combinatorics*, 5:29–46, 1989.

[21] E. Dahlhaus, J. Gustedt, and R. M. McConnell. Efficient and practical modular decomposition. In *Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 26–35, 1997.

[22] P. Duchet. Classical perfect graphs. *Annals of Discrete Mathematics*, 21:67–96, 1984.

[23] T. Gallai. Transitiv orientierbare Graphen. *Acta Mathematica Hungarica*, 18:25–66, 1967.

[24] A. Ghouila-Houri. Caractérisation des graphes non orientés dont on peut orienter les arêtes de manière à obtenir le graphe d'une relation d'ordre. *C. R. Acad. Sci. Paris*, 254:1370–1371, 1962.

[25] M. C. Golumbic. Threshold graphs and synchronizing parallel processes. *Coll. Mathematica Societatis János Bolyai, Combinatorics, Keszthely*, pages 331–352, 1976.

[26] M. C. Golumbic. Trivially perfect graphs. *Discrete Mathematics*, 24:105–107, 1978.

[27] M. C. Golumbic. *Algorithmic graph theory and perfect graphs*. Academic Press, London, 1980.

[28] M. C. Golumbic, H. Kaplan, and R. Shamir. Graph sandwich problems. *Journal of Algorithms*, 19:449–473, 1995.

[29] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[30] P. L. Hammer, T. Ibaraki, and U. N. Peled. Threshold numbers and threshold completions. *Annals of Discrete Mathematics*, 11:125–145, 1981.

[31] P. L. Hammer and N. V. R. Mahadev. Bithreshold graphs. *SIAM Journal on Algebraic and Discrete Methods*, 6:497–506, 1985.

[32] P. L. Hammer, N. V. R. Mahadev, and U. N. Peled. Some properties of 2-threshold graphs. *Networks*, 19:17–23, 1989.

[33] P. L. Hammer, N. V. R. Mahadev, and U. N. Peled. Bipartite bithreshold graphs. *Discrete Applied Mathematics*, 119:79–96, 1993.

[34] P. H. Henderson and Y. Zalcstein. A graph-theoretic characterization of the PV_{chunk} class of synchronizing primitives. *SIAM Journal on Computing*, 6:88–108, 1977.

[35] A. Hertz. Bipartable graphs. *Journal of Combinatorial Theory, Series B*, 45:1–12, 1988.

[36] C. T. Hoàng. Efficient algorithms for minimum weighted colouring of some classes of perfect graphs. *Discrete Applied Mathematics*, 55:133–143, 1994.

[37] C. T. Hoàng. On the complexity of recognizing a class of perfectly orderable graphs. *Discrete Applied Mathematics*, 66:218–226, 1996.

[38] C. T. Hoàng, S. Hougardy, and F. Maffray. On the P_4-structure of perfect graphs V: Overlap graphs. *Journal of Combinatorial Theory, Series B*, 67:212–237, 1996.

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

[40] C. T. Hoàng and B. A. Reed. Some classes of perfectly orderable graphs. *Journal of Graph Theory*, 13:445–463, 1989.

[41] T. Ibaraki and U. Peled. Sufficient conditions for graphs to have threshold number 2. *Annals of Discrete Mathematics*, 11:241–268, 1981.

[42] H. Imai and T. Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *Journal of Algorithms*, 4:310–323, 1983.

[43] B. Jamison and S. Olariu. P-components and the homogeneous decomposition of graphs. *SIAM Journal on Discrete Mathematics*, 8:448–463, 1995.

[44] D. S. Johnson. A catalog of compexity classes. *Handbook of Theoretical Computer Science, Elsevier*, pages 67–161, 1990.

[45] D. Kelly. Comparability graphs. In I. Rival, editor, *Graphs and Order*, NATO ASI Series, pages 3–40, Banff, Canada, 1984.

[46] J. Kratochvíl. A special planar satisfiability problem and a consequence of its NP-completeness. *Discrete Applied Mathematics*, 52:233–352, 1994.

[47] J. Kratochvíl and J. Nešetřil. Independent set and clique problems in intersection defined classes of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 31:85–93, 1990.

[48] L. Lovász. Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics*, 2:253–267, 1972.

[49] L. Lovász. *Selected topics in graph theory, Volume 2*. Academic Press, London, 1983.

[50] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. In *Proceedings of the 25th ACM Symposium on the Theory of Computing*, pages 286–293, San Diego, California, 1993.

[51] T. H. Ma. On the threshold dimension 2 graphs. Technical report, Institute of Information Science, Academia Sinica, Nankang, Taipei, Republic of China, 1993.

[52] N. V. R. Mahadev and U. N. Peled. Strict 2-threshold graphs. *Discrete Applied Mathematics*, 21:113–13, 1988.

[53] N. V. R. Mahadev and U. N. Peled. *Threshold graphs and related topics*, volume 56. Annals of Discrete Mathematics, 1995.

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

[55] R. M. McConnell and J. P. Spinrad. Linear-time transitive orientation. In *Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 19–25, 1997.

[56] M. Middendorf and F. Pfeiffer. On the complexity of recognizing perfectly orderable graphs. *Discrete Mathematics*, 80:327–333, 1990.

[57] R. H. Möhring. Algorithmic aspects of comparability graphs and interval graphs. In I. Rival, editor, *Graphs and Order*, NATO ASI Series, pages 41–101, Banff, Canada, 1984.

[58] R. H. Möhring. Algorithmics aspects of substitution decomposition in optimization over relations, set systems and boolean functions. *Annals of Operations Research*, 4:195–224, 1985.

[59] R. H. Möhring and F. J. Rademacher. Substitution decomposition and connections with combinatorial optimization. *Annals of Discrete Mathematics*, 19:257–356, 1984.

[60] S. Olariu. A decomposition for strongly perfect graphs. *Journal of Graph Theory*, 13:301–311, 1989.

[61] E. T. Ordman. Threshold coverings and resource allocation. In *Sixteenth Southeastern Conference on Combinatorics, Graph Theory and Computing*, pages 99–113, 1985.

[62] E. T. Ordman. Minimal threshold separators and memory requirements for synchronization. *SIAM Journal on Computing*, 18:152–165, 1989.

[63] C. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[64] R. Petreschi and A. Sterbini. Strict 2-threshold exclusion graphs. In *Theoretical Computer Science - Proceedings of the Fourth Italian Conference*, pages 293–304, L'Aquila, 1992.

[65] R. Petreschi and A. Sterbini. Recognizing strict 2–threshold graphs in $O(m)$-time. *Information Processing Letters*, 54:193–198, 1995.

[66] T. Raschle and K. Simon. Recognition of graphs with threshold dimension two. In *27th Annual ACM Symposium on the Theory of Computing*, pages 650–661, Las Vegas NE, 1995.

[67] T. Raschle and K. Simon. On the P_4-components of graphs. *Discrete Applied Mathematics*, to appear.

[68] T. Raschle and A. Sterbini. Recognizing cobithreshold graphs in linear time. In *16th International Symposium of Mathematical Programming*, Lausanne, 1997.

[69] B. Reed. A semi-strong perfect graph theorem. *Journal of Combinatorial Theory, Serie B*, 43:223–240, 1987.

[70] R. L. Rivest. Cryptography. *Handbook of Theoretical Computer Science, Elsevier*, pages 717–755, 1990.

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

[72] A. Schäffer. Recognizing brittle graphs: remarks on a paper of Hoàng and Khouzam. *Discrete Applied Mathematics*, 31:29–35, 1991.

[73] D. Seinsche. On a property of the class of n-colorable graphs. *Journal of Combinatorial Theory, Series B*, 16:191–193, 1974.

[74] A. Sterbini. *2-thresholdness and its implications: from the synchronization with PV_{chunk} to the Ibaraki-Peled conjecture.* PhD thesis, University "La Sapienza", Rome, 1994.

[75] A. Sterbini and T. Raschle. An $O(n^3)$ recognition algorithm for threshold dimension 2 graphs. *Information Processing Letters*, 67(5):255–259, 1998.

[76] M. Yannakakis. The complexity of the partial order dimension problem. *SIAM Journal on Algebraic and Discrete Methods*, 3:351–358, 1982.

Index

Curriculum Vitae

Name:	Thomas RASCHLE
Date of Birth:	1^{st} of May, 1960
Place of Birth:	Bütschwil
Nationality:	Swiss

1967 – 1973	Primarschule Bütschwil
1973 – 1976	Sekundarschule Bütschwil
1976 – 1980	Heberlein Maschinenfabrik AG (Mechanikerlehre)
1977 – 1980	Berufsmittelschule St.Gallen
1980 – 1983	Zweitweg-Matura St.Gallen
1981 – 1982	Maschinenfabrik Egli (Maschinenbau)
1982 – 1983	Fremasoft (Programmierung)
1983 – 1984	ETH Zürich (1.Vordiplom Physik)
1984 – 1987	Universität Zürich (1.Vordiplom Mathematik)
1985 – 1989	Ericsson Information Systems AG bzw. Nokia Data AG (Programmierung)
1988 – 1993	ETH Zürich (Dipl. Informatik-Ing. ETH)
1991	Swissair AG (Praktikum)
1992	Alcatel AG (Praktikum)
1993 – 1998	ETH Zürich (Assistenz)
1998 – 1999	Roland Messerli AG (Programmierung)