2019

Separability and Vertex Ordering of Graphs

Elizabeth Gorbonos
gorb6620@mylaurier.ca

Separability and Vertex Ordering of Graphs

By:

Elizabeth Gorbonos

A thesis

Submitted to the Department of Physics and Computer Science

in partial fulfilment of the requirements for

Master of Applied Computing

in

Computer Science

Wilfrid Laurier University

Waterloo, Ontario, Canada, 2019

Abstract

Many graph optimization problems, such as finding an optimal coloring, or a largest clique, can be solved by a divide-and-conquer approach. One such well-known technique is decomposition by clique separators where a graph is decomposed into special induced subgraphs along their clique separators. While the most common practice of this method employs minimal clique separators, in this work we study other variations as well. We strive to characterize their structure and in particular the bound on the number of atoms. In fact, we strengthen the known bounds for the general clique cutset decomposition and the minimal clique separator decomposition.

Graph ordering is the arrangement of a graph's vertices according to a certain logic and is a useful tool in optimization problems. Special types of vertices are often recognized in graph classes, for instance it is well-known every chordal graph contains a simplicial vertex. Vertex-ordering, based on such properties, have originated many linear time algorithms. We propose to define a new family named SE-Class such that every graph belonging to this family inherently contains a simplicial extreme, that is a vertex which is either simplicial or has exactly two neighbors which are non-adjacent. Our family lends itself to an ordering based on simplicial extreme vertices (named SEO) which we demonstrate to be advantageous for the coloring and maximum clique problems. In addition, we examine the relation of SE-Class to the family of (Even-Hole, Kite)-free graphs and show a linear time generation of SEO for (Even-Hole, Diamond, Claw)-free graphs. We showcase the applications of those two core tools, namely clique-based decomposition and vertex ordering, on the (Even-Hole, Kite)-free family.

Acknowledgements

First and foremost I would like to thank my supervisor, Professor Chính T. Hoàng, without whom this work could not have come to fruition. The expertise, creative thinking and enthusiastic guidance he has provided me with throughout this research were invaluable.

I would also like to acknowledge the Physics and Computer Science department at Wilfrid Laurier University and in particular, my course instructors who have equipped me with instrumental skills and a great starting point for the thesis. A special thanks is reserved to Professor Yang Liu for her help with preparing and publishing a paper in the field of machine learning.

Last, but not the least, I thank my family and friends for supporting me along the journey. Their encouragement and assistance instilled me with much confidence and motivation.

Contents

1 Introduction **1**

1.1 Definitions and Notations 1

1.2 Graph Problems 4

1.2.1 Chromatic Number and Clique Number 4

1.2.2 Approaches and Tools 4

1.3 Motivation 7

1.4 Organization of the Thesis 11

2 Background **12**

2.1 Perfect Graphs 12

2.1.1 The Strong Perfect Graph Theorem 14

2.1.2 Properties of Perfect Graphs 17

2.1.3 Problems on Perfect Graphs 17

2.2 Graph traversal 20

2.2.1 Fundamentals 21

2.2.2 LexBFS 23

2.3 Decomposition and Separability of Graphs 27

2.3.1 Clique Separator Decompostion 28

2.3.2 Other Cutset Decompositions 34

2.3.3 Partitioning 36

2.4 Ordered Graphs . . . 37
2.4.1 β-Perfect . . . 37
2.4.2 Perfectly Orderable Graphs . . . 38
2.5 Hole-free Graphs . . . 39

3 Clique-Cutset Decomposition 43
3.1 Decomposition Properties . . . 43
3.1.1 Cutsets Intersection . . . 47
3.2 Atom Properties . . . 49
3.2.1 On Non-maximal Atoms . . . 51
3.2.2 On the Number of Atoms . . . 53
3.3 Minimal clique cutset decomposition . . . 56
3.3.1 Decomposition Properties . . . 57
3.3.2 Atoms Properties . . . 61
3.3.3 On the Number of Atoms of the Decomposition . . . 62
3.4 Maximal clique cutset decomposition . . . 66
3.4.1 Decomposition Properties . . . 66
3.4.2 Atoms Properties . . . 66
3.4.3 On the Number of Atoms of the Decomposition . . . 68

4 SE-Class 71
4.1 SEO . . . 72
4.1.1 Generating SEO . . . 73
4.1.2 Verifing SEO . . . 73
4.1.3 Applications . . . 77
4.2 Structural Results . . . 79
4.2.1 (Even-Hole, Claw)-free . . . 79
4.2.2 (Even-Hole, Kite)-free . . . 81

4.2.3 (Even-Hole, Claw, Diamond)-free 83
4.3 SEO and LexBFS 92

5 Case study: Coloring and certifying (Even-Hole, Kite)-free 99
5.1 The Structure of k-critical Graphs 99
5.2 Certifying Algorithm for k-colorability 102
5.2.1 (Even-Hole)-free SE-Class Graphs 103
5.2.2 (Even-Hole, Kite)-free Graphs 105

6 Conclusions and Open Problems 109
6.1 Clique Cutset Decomposition 109
6.2 SE-Class 110

Bibliography 112

List of Tables

2.1 Perfect F-free Berge graphs 15
2.2 Partial Results for β-perfect graphs 38
2.3 Problems on hole-free classes 42

List of Figures

1.1 Some basic graphs examples 3

1.2 Important induced subgraphs 7

2.1 Graphs: Paw, Bull 14

2.2 Q graph 34

2.3 3-path configurations 40

3.1 Illustration of a decomposition path in $T(G)$ 44

3.2 Example: minimal clique separator vs. minimal clique cutset 45

3.3 Cutsets of decomposition blocks 46

3.4 Decomposition with mutually-exclusive clique cutsets 48

3.5 Clique cutset "preserving" decomposition 49

3.8 A clique cutset decomposition resulting in non-linear number of atoms 54

3.9 Two different maximal clique cutset decompositions 67

3.10 A "larger" clique cutset decomposition 68

4.1 SE-Class membership example 71

4.2 Example of a bad path 79

4.3 Minimally non-SE-Class (Even-Hole, Claw)-free graphs 81

4.4 Spear 84

4.5 Types of spears 85

4.6 Edges in a minimal-spear 88

4.7 Illustration for Lemma 4.2.16 89

4.8 A SE-Class graph for which a LexBFS ordering is not SEO 93

6.1 A SE-Class graph containing a structure 111

Chapter 1

Introduction

We start this chapter by introducing fundamental definitions and notations in Section 1.1. Further terms and concepts will appear throughout this thesis, wherever appropriate. In Section 1.2 we present common graph problems and in Section 1.3 we describe our particular topics of interests. Section 1.4 concludes the introduction by giving an outline of this thesis.

1.1 Definitions and Notations

A *graph* $G = (V, E)$ is a structure composed of the *vertex-set* V and the *edge-set* E. G contains an edge between $u \in V$ and $v \in V$ if $uv \in E$ (we then also refer to u and v as *adjacent*). A *simple* graph is a graph for which the edge-set E consists of unordered pairs of unique vertices. A simple graph has no *multiple edges* (there is at most one edges between any two vertices) and no *loops* (there is no edge from a vertex to itself). All graphs in this work are simple and *finite* (meaning, the number of vertices is bounded). We use the common notations $n = |V|$ and $m = |E|$ for the number of vertices and the number of edges, respectively (the function $|X|$ here means the size of a set X). The notation $N_G(V)$ means the *neighborhood* of v, that is the set of all vertices v is adjacent to in G (we may omit the subscript when the

context is clear). To describe the set consisting of v and its neighborhood we use $N_G[V] = N_G(V) \cup \{v\}$. A graph $H = (V', E')$ is said to be a subgraph of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. A special type of subgraph is an *induced subgraph* which we describe by $G[V']$ where $V' \subseteq V$. Let $H = (V', E') = G[V']$, then the edge-set E' is constrained as follows: for $\{u, v\} \subseteq V'$ we have $uv \in E'$ only if $uv \in E$. $G[V']$ is a properly induced subgraph if $V' \subset V$. All subgraphs in this thesis are induced subgraphs. A *graph family* (or graph class) $\mathcal{C}$ is a set of graphs in which all members hold a certain property. Note that graph families need not be finite. A graph property is said to be *hereditary* if for any $G \in \mathcal{C}$ every induced subgraph H of G also belongs to $\mathcal{C}$, i.e. $H \in \mathcal{C}$. Restricted graph classes are defined in the following way: let F be some graph, demand that every $G \in \mathcal{C}$ does not contain F as an induced subgraph. Accordingly we name this family F-free. Similarly, let $\mathcal{F}$ be a (possible infinite) set of graphs then G is $\mathcal{F}$-free if it does not contain any $F \in \mathcal{F}$. Properties of the form $\mathcal{F}$-free are hereditary.

We shall start by presenting some basic graph classes:

- *Paths* - P_n is a graph on n vertices $V = \{v_1, ..., v_n\}$ which contains the edges $v_i v_{i+1} \in E$ for $1 \leq i \leq (n-1)$. Any other edge, $v_i v_j \in E$ where $|j - i| > 1$, is called a chord. When referring to a path connecting two vertices v and u, v and u are the *endpoints* of the path and all other vertices are called *interior*.

- *Cycles* - A cycle C_n is a path P_n, on at least 4 vertices, with the addition of the edge $v_n v_1 \in E$. It can be visualized as a closed path. A chord $v_i v_j \in E$ in a cycle is an edge for which $|j - i| > 1$ and also $v_i v_j \neq v_n v_1$. *Holes* are cycles without chords.

- *Wheels* - A wheel W_n is composed of a hole on n vertices and a single *universal* vertex, that is a vertex $u \in V$ which is adjacent to all other $v \in V$.

- *Cliques* - A clique K_n is a set of n vertices which are all universal. Alternatively,

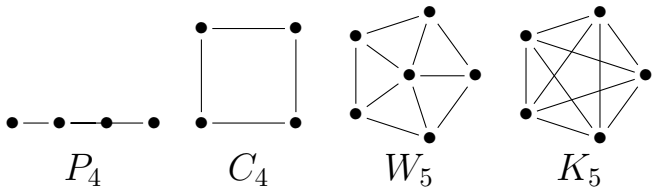


Figure 1.1: Some basic graphs examples

$v_iv_j \in E$ for any $1 \leq i, j \leq n, i \neq j$.

A graph G is said to be *connected* if for every pair $\{v, u\} \subseteq V$ there exists an induced path with v and u as its endpoints. For a given graph $G = (V, E)$, with $\{u, v\} \subseteq V$, we define its *complement* $\overline{G} = (V, E')$ by applying the rule $uv \in E'$ if and only if $uv \notin E$. The prefix *anti* in this paper refers to the complement graph (e.g. anti-C_5). An *independent set* is a set of vertices $S \subset V$ with no edges amongst them, in other words for every $\{u, v\} \subseteq S$ we have $uv \notin E$. Hence, an independent set of $\overline{G}$ translates into a clique of G.

We will use the notation $G - X$ where $X \subset V$ as a shorthand for $G[V - X]$. Let $v \in V - X$, we say v is X-complete if it is adjacent to every $u \in X$ (alternatively, $X \subseteq N(v)$), v is X-null if v is not adjacent to any vertex of X (i.e. $X \cap N(v) = \emptyset$) and v is X-partial is it is neither X-complete nor X-null. An edge is X-complete if both its endpoints are X-complete.

Finally, let us describe two important graph classes:

- *Bipartite graphs* - A graph is bipartite if its vertices can be partitioned into two independent sets. A *complete bipartite graph*, denoted $K_{x,y}$, is a bipartite graph with independent sets X and Y, with respective sizes x and y and where for every pair $\{v, u\}$ with $v \in X$ and $u \in Y$ we have $vu \in E$.

- *Chordal graphs* - A graph G is called chordal if it contains no holes (also known as triangulated [84] and rigid circuit graph [41]).

1.2 Graph Problems

Going as far back as 1736, to the famous Seven Bridges of Königsberg riddle, the field of graph theory spawn many problems [4]. Two fundamental ones concern the optimal vertex-coloring and the maximum clique.

1.2.1 Chromatic Number and Clique Number

A vertex-coloring is the assignment of colors to all vertices $v \in V$, such that every two adjacent vertices $\{u, v\} \subseteq V$ are given different colors. A graph is k-colorable if it can be colored utilizing k colors. Since this task can always be accomplished with n colors we are in fact interested in the smallest number of colors required. This number is denoted $\chi(G)$ and is known as the *chromatic number* of the graph, a vertex-coloring using only $\chi(G)$ colors is referred to as optimal coloring. The *clique number* of G, also denoted as $\omega(G)$, is the number of vertices in a set S which induces the largest clique of G. The real-world applications of coloring include scheduling [70] and channel frequency assignment [44]. Finding cliques in a graph is commonly motivated by network analysis in domains such as sociology and biology [47, 43].

1.2.2 Approaches and Tools

For an arbitrary graph G, it is well known that testing whether it is k-chromatic and finding a clique of size t are NP-complete [59, 49]. However, many algorithms have been proposed to improve the worst-case bound. Berge [4] originally suggested two naive approaches to find the chromatic number: 1) A top down method where we first color G and then try to eliminate redundant colors; 2) A bottom up method in which we attempt to color the graph with all $k \in \{3, ..., n\}$ colors ($k = 1$ means the graph has no edges and $k = 2$ implies G is bipartite). Alternatively, one may attempt to solve the clique cover problem on $\overline{G}$. The clique cover, denoted $\theta(G)$, is the minimum

number of cliques required to cover the vertex-set V. Since every clique in $\overline{G}$ is an independent set in G we obtain the optimal coloring by assigning each clique a unique color. The fastest algorithm currently known runs in time $O(2.2461^n)$. It utilizes set partitioning to find the chromatic number [7]. The clique number of G is closely linked with the *maximum independent set* (denoted by $\alpha(G)$) of $\overline{G}$ (in a similar way $\chi(G)$ and $\theta(\overline{G})$ are related). A naive algorithm to find $\alpha(G)$ is to check all 2^n subsets of G. Tarjan and Trojanowski [92] presented the first significant improvement over the basic method by constructing a $O(2^{n/3})$ time-complexity algorithm. A number of later works were able to slightly improved this result [58, 82, 83].
Due to the hardness of the general case other courses of action have been extensively explored.

Definition 1.2.1 *An **ordering** $\sigma = \{v_1, ..., v_n\}$ is an ordered list of all vertices of a graph G. G_i is produced by inducing a graph on G using the set S from σ starting at index i (i.e. $\{v_i, ..., v_n\}$). The notation $deg_\sigma(v_i)$ in this context stands for the degree of vertex v_i in G_i and $N_\sigma(v_i)$ is the neighbors of v_i in G_i.*

One example of a heuristic tactic is greedy algorithms. These are a family of algorithms which attempt to maximize (or minimize) some target value. This type of algorithms make their choices based on the "current" state, therefore they are prone to "get stuck" is local maximums (or local minimums). Regardless, they are easy to implement and for certain problems they perform properly. For instance finding a maximal clique is straightforward using a greedy algorithm, we may start at any vertex and then add all vertices which increase our current clique. A greedy coloring algorithm performs in the following way, it traverses the graph in some prescribed order and assigns every vertex the smallest available color. Greedy coloring is guaranteed to produce a coloring with at most $\Delta(G) + 1$ colors (where Δ is the largest degree of any $v \in V$). Generally speaking, the greedy algorithm is not able to find the maximum clique or produce an optimal coloring. However, for particular

families of graphs, given an appropriate ordering a greedy algorithm will perform accurately. We will see several examples in Section 2.4.

Another way to address the NP-completeness barrier is to reduce the problem space by restricting the allowed input graphs. Choosing to limit the discussion to specific graph families motivates developing structure theories and auxiliary tools such as vertex ordering and decomposition. Let us demonstrate the benefits of ordering on the chordal graph family. A *simplicial vertex* is a vertex whose neighborhood is a clique. It is well known, from structural analysis, that every chordal graph contains a simplicial vertex (as a matter of fact a stronger theorem shows every chordal graph G which is not a clique contains two non-adjacent simplicial vertices) [41]. Using this knowledge we can build an ordering for G by repeatedly removing simplicial vertices (note the hereditary property guarantees the existence of a simplicial vertex in any induced subgraph). The order we create is known in literature as a *perfect elimination order* (PEO) and can be used to optimally color G and identify $\omega(G)$ in linear time. It follows that computing $\chi(G)$ and $\omega(G)$ in this case can be achieved in $O(\textit{generating PEO}) + O(n+m)$ time. In Section 2.2.2 we show a PEO for a chordal graph G can be generated in linear time (Theorem 2.2.7), making the entire problem linear-time solvable. This is a very elegant result as opposed to the general NP-hard case. Decomposition is a technique designed to split or reduce a graph by identifying specific structures. Decompositions are the foundation of many divide and conquer graph algorithms. The objective of those algorithms is to solve the problem on smaller "simpler" graphs and later "stitch" together the complete solution. This often proves more efficient than trying to tackle the original graph directly as we demonstrate in Chapter 5.

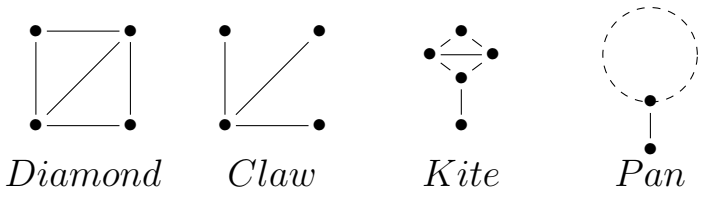


Figure 1.2: Important induced subgraphs

1.3 Motivation

The well awaited proof of the Strong Perfect Graph Theorem in [19] was a significant milestone in the study of graphs. Furthermore, it rendered many open problems easy to solve and consequently motived a new line of research focusing on (Even-Hole)-free or (Odd-Hole)-free graphs. Our study is primarily inspired by prior results in this vein of work. Table 2.3 (in Section 2.5) suggests the (Odd-Hole)-free class is harder to crack, hence we pay more attention to (Even-Hole)-free graphs.

Figure 1.2 depicts some important graphs which will appear in this work. A *Diamond* is the graph produced by removing one edge from a K_4. A *Claw*, with center v, is the bipartite graph $K_{1,3}$ where v is the single-vertex set. A *Kite* is a diamond with a vertex attached to it as illustrated in figure 1.2. A *Pan* is a hole with an auxiliary vertex. We note, that the Kite generalizes the Diamond and the Pan generalizes the Claw. We now define a number of observable graph properties.

Definition 1.3.1 *Let $G = (V, E)$ be a graph. A set of vertices $C \subset V$ is called a* ***cutset*** *if and only if the graph $G - C$ can be partitioned into two vertex-sets A and B such that there are no edges between A and B. C is a* ***clique cutset*** *if it is a cutset and the subgraph induced by C is a clique.*

Definition 1.3.2 *Let $G = (V, E)$ be a graph, a vertex $v \in V$ is a* ***simplicial extreme*** *if the neighborhood of v is a clique or the degree of v is 2 (i.e. the neighborhood is exactly two non-adjacent vertices).*

Definition 1.3.3 *Let G_1 and G_2 be two graphs. The graph G is **join** of G_1 and G_2 (denoted $G = G_1 \oplus G_2$) if for every $v \in V(G_1)$ and $u \in V(G_2)$, vu is an edge in G (i.e. $vu \in E(G)$).*

Kloks et al. [63] established the following nice structural theorem.

Theorem 1.3.4 *[63] If G is an (Even-Hole, Diamond)-free graph, then one of the following holds:*

- *G is a clique, or*
- *G contains two non-adjacent simplicial extremes.*

The existance of simplicial extreme vertices implies a $O(n^2m)$ time-complexity for coloring (Even-Hole, Diamond)-free graphs. In [48] this work was extended to encompass (Even-hole, Kite)-free graphs, but the theorem in that paper is in fact stronger as is it applies to (C_4, Kite)-free graphs.

Theorem 1.3.5 *[48] Let G be a connected (C_4, Kite)-free graph. Then one of the following holds.*

- *G is diamond-free, or*
- *G is the join of a clique and a diamond-free graph, or*
- *G contains a clique cutset*

Another result given in this work states that:

Theorem 1.3.6 *[48] (Even-Hole, Kite)-free graphs are β-perfect.*

The final structural result we bring here regards (Even-Hole,Pan)-free graphs and requires one additional definition.

Definition 1.3.7 *A **Circular-Arc graph** G is the mapping of a set A of arcs on some circle, such that every vertex represents an arc, and two vertices of G are adjacent if the two corresponding arcs intersect on the circle. A **Unit Circular-Arc graph** is a circular-arc graph for which all arcs have the same length.*

Theorem 1.3.8 *[13] If G is (Even-Hole, Pan)-free, then*

- *G is a clique, or*
- *G contains a clique cutset, or*
- *G is a unit circular-arc graph, or*
- *G is the join of a unit circular-arc graph and a clique.*

Fraser et al. [48] and Cameron et al. [13] present poly-time algorithms to color (Even-Hole, Kite)-free and (Even-Hole, Pan)-free graphs with run times of $O(n^3m)$ and $O(n^{2.5}+nm)$ respectively. Both coloring algorithms exploit Tarjan's [91] decomposition technique and rely on the fact that an efficient decomposition produces at most $n-1$ atoms (see Section 2.3.1). Examining the presented results, we extract two properties to be at the center of our thesis, namely: clique cutsets and simplicial extremes.

The idea to utilize clique cutsets to decompose a graph is described by Gavril [51]. A *prime graph* is a natural concept in decompositions. It takes different meanings with respect to the preformed decomposition and we formally define and redefine it numerous times in Section 2.3. For the purpose of discussing the clique cutset decomposition we say a prime graph is a graph that does not contain a clique cutset. A *maximal prime subgraph* (mp-subgraph) H of G is an induced subgraph of G which is prime and is maximal with respect to the number of vertices. An *atom* of a decomposition is a prime subgraph produced by the decomposition process, we remark that an atom is not necessarily a mp-subgraph. Many have studied this decomposition,

including Whitesides [101] and Diestel [39]. Leimer [65] proved that using minimal clique separators produces the optimal (smallest) number of atoms. So far most efforts in this line of work were focused on optimizing the decomposition algorithm and minimizing the number of atoms. We are however interested in a comprehensive theoretical characterization of this type of decomposition. Our main results for this chapter are:

- Theorem 3.3.17 tightens the maximum bound, for a minimal clique separators decomposition, to $n - |H| + 1$. Where H is the largest mp-subgraph of G.
- Theorem 3.2.7 refines the upper bound, on the number of atoms, for any clique cutset decomposition. We show the number of atoms is at most $\frac{n^2}{4}$. This is a more precise result than the previously reported n^2 estimation [51].

The second feature we set out to explore is the presence of simplicial extremes. Simplicial extremes only slightly differ from simplicial vertices and encourage us to consider them in orderings (this is reminiscent of the PEO). In fact we show in Section 4.1.3 that once a simplicial-extreme order (SEO) is generated for a graph, assuming such order exists, both the chromatic number and the clique number can be found in linear time (Theorems 4.1.5 and 4.1.6). We define a new graph family $\mathbb{C}$ with the property: $G \in \mathbb{C}$ if for any $X \subseteq V$ there is a simplicial extreme in $G[X]$. From [63] we know $\mathbb{C}$ contains (Even-Hole, Diamond)-free graphs. We ask the following questions:

- Which other graph families are in $\mathbb{C}$?
- How can we generate and verify a SEO order?
- For which graph classes every LexBFS produces a SEO?

Our major findings are:

- Corollary 4.2.6 shows that an (Even-Hole, Kite)-free graph G is either in $\mathbb{C}$ or G contains a wheel.

- Theorem 4.3.3 demonstrates that every LexBFS ordering for an (Even-Hole, Diamond, Claw)-free graph is a SEO.

1.4 Organization of the Thesis

Chapter 2 presents a survey of related works, starting from a historical review on the study of perfect graphs. Perfect graphs are the cornerstone to many of the techniques and topics in the remainder of that chapter including lexicographical breadth first search (LexBFS), various decomposition methods, vertex-orderings and the currently ongoing research on hole-free graphs. In Chapter 3 we dive into the clique cutset decomposition and attempt to characterize multiple aspects of it, from the decomposition itself to properties of the atoms and their quantity. We begin with a general decomposition, meaning we place no constraints over the clique cutsets chosen at each step. Later we consider extreme cases including the minimum and minimal clique cutset decomposition as well as the maximum and maximal clique cutset decomposition. We propose a new class named *SE-Class* in Chapter 4, which is defined by the ideal that any graph G in this class has a simplicial extreme and so does every induced subgraph of G. We commence by demonstrating the benefits of our prescribed property for solving the coloring and maximum clique problem. Then we examine the (Even-Hole, Claw)-free and (Even-Hole, Kite)-free classes to determine whether they belong to SE-Class. Finally we identify a structure we name *spear* in (Even-Hole, Claw, Diamond)-free graphs (a subclass of SE-Class) which helps us prove that any LexBFS on such graphs creates a SEO. The purpose of Chapter 5 is to showcase how clique cutset decomposition along with structural results on (Even-Hole, Kite)-free graphs allow us to design a certifying algorithm for k-colorability of graphs in this family. Lastly, in Chapter 6 we summarize and pose the questions left unresolved in this work.

Chapter 2

Background

In this chapter we present an overview of related graph families along with a broad literature survey of tools essential to our work. We start by introducing the class of perfect graphs in Section 2.1. This graph family was first introduced in the 1960s by Berge and has drawn much attention since. It is also the prime motivator behind many of the techniques covered in this chapter. Section 2.2 presents graph traversal algorithms, focusing on LexBFS. In Section 2.3 we survey multiple decomposition techniques. Section 2.4 demonstrates several graph families which are defined via special orderings. And lastly, we examine the known results on hole-free graphs in Section 2.5.

2.1 Perfect Graphs

In the late 1950s Berge set out to explore new combinatorial properties of graphs. He proposed two new graph families which he named γ-perfect and α-perfect, we now know these classes simply as perfect graphs [3]. The original definition of γ-perfect graphs was identical to the one given here.

Definition 2.1.1 *A graph G is **perfect** if for every induced subgraph H of G the*

property $\chi(H) = \omega(H)$ *holds.*

On the other hand, α-perfect graphs were defined by the hereditary property $\alpha(H) = \theta(H)$ for every induced subgraph H of G. We have pointed out in Section 1.2.2 that $\alpha(G) = \omega(\overline{G})$ and $\theta(G) = \chi(\overline{G})$ thus α-perfection of G implies γ-perfection of $\overline{G}$. Several prominent graph classes were recognized to be perfect soon after Berge's definitions. For instance, Berge noted bipartite graphs and chordal graphs (as proven by Hajnal and Suranyi) were perfect [3, 10]. However, two non-perfect infinite families, became quickly known. Specifically, graphs containing induced odd-holes and graph with induced odd-antiholes are not perfect. An odd-hole G on $2k+1$ vertices ($k \geq 2$) has $2 = \omega(G) \neq \chi(G) = 3$ and its complement $\overline{G}$ has $k = \omega(\overline{G}) \neq \chi(\overline{G}) = k + 1$. These findings motivated Berge to formulate the next two conjectures.

1. A graph G is perfect if and only if its complement is perfect.

2. A graph G is perfect if and only if it does not contain an odd-hole or an odd-antihole.

These conjectures are commonly referred to as the *Weak Perfect Graph Conjecture* (WPGC) and the *Strong Perfect Graph Conjecture* (SPGC), respectively. The family of graphs described by the SPGC is also known as *Berge* graphs.

Definition 2.1.2 *A graph G is* ***Berge*** *if it contains no induced odd-hole and no induced odd-antihole.*

Lovász [67] managed to prove the WPGC using the *replication lemma* and with that made the dual terminology of α-perfect and γ-perfect obsolete.

Lemma 2.1.3 *[67] When substituting perfect graphs for some vertices of a perfect graph the obtained graph is also perfect.*

Theorem 2.1.4 *[67] (Perfect Graph Theorem) A graph G is perfect if and only if $\overline{G}$ is perfect.*

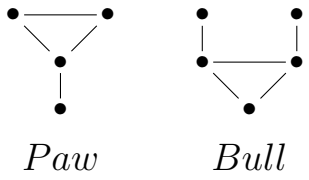


Figure 2.1: Graphs: Paw, Bull

2.1.1 The Strong Perfect Graph Theorem

Unlike the WPGC, the SPGC resisted proof for almost four decates up utill the early 2000s when it was finally proven by Chudnovsky et al. [19]. It drew quite a bit of attention over those years and cultivated many new techniques throughout the various attempts to prove it. It is important to note one side of the SPGC is straightforward: any graph which is not Berge is not perfect, as by definition it contains an induced subgraph which obstructs perfection. For that reason, all efforts focused on proving every Berge graph is also perfect.

Many of the results cited here use special types of cutsets. Section 2.3 is dedicated to decompositions and provides complete definitions for all relevant terms. We mark with (*) terms which will be properly defined later. A *Paw* (see Fig. 2.1) is a connected graph on four vertices which consists of a triangle and exactly one more edge. A *Bull* is the graph on five vertices depicted in Fig. 2.1 and is self-complementary (meaning $G = \overline{G}$).

One endeavor aimed to characterize perfect graphs by means of restricted-graph families. In fact, the perfection of all F-free Berge graphs with F being any connected graph on four-vertices was gradually established (see Table 2.1). Yet it is clear such an approach cannot settle the SPGC.

At the same time a parallel effort was carried out and involved decomposing (and in certain cases composing [35]) perfect graphs. The results of the decompositions were used to characterize a *minimal imperfect graph* (w.r.t. the number of vertices). In a more precise manner we say a *minimal imperfect graph* is a graph G which is not

F-free Berge	Source
P_4-free	Seinsche (1974) [86]
Claw-free	Parthasarathy and Ravindra (1976) [81]
K_4-free	Tucker (1977) [97]
Diamond-free	Tucker (1987) [99]
Paw-free	Olariu (1988) [79] based on [73]
C_4-free	Conforti et al. (2004) [27]

Table 2.1: Account of establishing perfection for F-free Berge graphs

perfect but every properly induced subgraph H of G is. We will discuss notable results for minimally imperfect graphs, but beforehand we must define a few properties.

Definition 2.1.5

*An **Even-Pair** in $G = (V, E)$ is a pair of two non-adjacent vertices $\{v_1, v_2\} \subseteq V$ such that all induced paths with endpoints v_1, v_2 have an even number of edges.*

*An **Antitwin** in $G = (V, E)$ is a pair of vertices $\{v_1, v_2\} \subseteq V$ for which every vertex $u \in (G - \{v_1, v_2\})$ is adjacent to exactly one of $\{v_1, v_2\}$.*

*A **Homogeneous pair** in $G = (V, E)$ is a pair of disjoint vertex-sets A, B ($A \subset V$, $B \subset V$) such that:*

1. *all A-partial vertices are in B and all B-partial vertices are in A,*
2. *at least one of A, B has size greater than two,*
3. *$|V(G - (A \cup B))| \geq 2$*

Cornuéjols and Cunningham studied perfection-preserving compositions and discovered the following result.

Lemma 2.1.6 *[35] No minimal imperfect graph contains a 2-join (*).*

Meyniel proved that:

Lemma 2.1.7 *(Even Pair Lemma) [74] No minimal imperfect graph contains an even pair.*

The next result by Chvátal inspired Olariu's theorem for pan-free Berge graphs. It also serves to demonstrate how observations on minimal imperfect graphs allowed researchers to further identify restricted perfect graph classes.

Lemma 2.1.8 *(Star Cutset Lemma) [23] No minimal imperfect graph contains a star cutset (*).*

Theorem 2.1.9 *[80] The Strong Perfect Graph Conjecture holds true for pan-free graphs.*

Another example is the Antitwin Lemma which was generalized by Chvátal and helped proving bull-free Berge graphs are perfect.

Lemma 2.1.10 *(Antitwin Lemma) [78] No minimal imperfect graph contains antitwins.*

Lemma 2.1.11 *(Homogeneous Pair Lemma) [24] No minimal imperfect graph contains a homogeneous pair.*

Theorem 2.1.12 *[24] The Strong Perfect Graph Conjecture holds true for bull-free graphs.*

Definition 2.1.13 *Two non-adjacent vertices u_1, u_2 of a graph G form a three-pair if all chordless paths of G joining u_1 to u_2 have exactly three edges.*

The antitwin structure was also utilized by Hoàng[56], who noted every even-pair in G forms a three-pair in G or in $\overline{G}$, thus leading to the following result:

Lemma 2.1.14 *(Three-Pair Lemma) [56] No minimal imperfect graph contains a three-pair.*

The final proof to the *Strong Perfect Graph Teorem* (SPGT) was a combination of the two approaches [19]. It was inspired by a proposition for decomposing perfect graphs mentioned in [27]. The authors conjectured that every Berge graph can be classified into some well-defined graph class (called *basic*) or it can be decomposed in a specific manner (referred to as a *useful* decomposition). The idea behind the proof was to show that a minimal imprefect graph does not admit any of the *useful* decompositions, thus it is *basic*. However, all basic graphs are perfect hence we arrive at a contradiction, meaning no Berge graph can be minimally imperfect. One of the adaptations required for this proof was the definition of balanced skew partitions(*).

2.1.2 Properties of Perfect Graphs

Following the previous discussion we are already aware of a powerful property of perfect graphs, namely a graph G is perfect if and only if it is Berge. We mention below two results of particular interest.

Theorem 2.1.15 *[73] A graph is perfect if each of its odd cycles with at least five vertices contains at least two chords.*

Theorem 2.1.16 *[19, 16] For every perfect graph G, either G is basic, or one of G, $\overline{G}$ admits a proper 2-join (*), or G admits a balanced skew partition (*).*

2.1.3 Problems on Perfect Graphs

Perfect graphs are nice in the sense that many NP-complete problems can be solved in polynomial time for this class. The definition of perfect graphs and the equality $\omega(G) = \alpha(\overline{G})$ (symmetrically, $\chi(G) = \theta(G)$) means a poly-time solution to one of the four problems entails an efficient algorithm for the rest, and these in fact exist [53]. To reap those benefits, it is desirable to recognize perfect graphs algorithmically in

poly-time as well. Fortunately, due to [18] we are now able to recognize perfect graph in $O(n^9)$ time.

Recognizing Perfect Graphs

Based on the SPGT, one can test if a graph is perfect by testing to see if it is Berge. A direct approach would be to follow the definition of a Berge graph and search for odd-holes, unfortunately there is no poly-time algorithm known to perform this task [14]. For this reason we are compelled to search for other structures resulting in odd-holes or odd-antiholes.

Definition 2.1.17 *$G = (V, E)$ has a **Pyramid** if $\{x_1, x_2, x_3\} \subset V$ is a triangle and there are three induced paths from x_1, x_2, x_3 to a vertex $y \in V$, P_1, P_2, P_3 respectively. At most one of P_1, P_2, P_3 has length one and the union of every two paths induces a hole.*

Definition 2.1.18 *$G = (V, E)$ has a **Jewel** if it has a cycle $C_5 = v_1, v_2, v_3, v_4, v_5$ with the following conditions: $v_1v_3 \notin E$, $v_1v_4 \notin E$, $v_2v_3 \notin E$ and there is some path P with endpoints v_1, v_4 and no edges between its interior vertices and $\{v_2, v_3, v_5\}$.*

Theorem 2.1.19 *If G contains a pyramid or a jewel it contains an odd hole.*

Proof. A pyramid necessarily contains two paths with the same parity. Assume, without loss of generality, P_1 and P_2 have the same parity, then $G[P_1 \cup P_2]$ is a odd hole. For a Jewel, assume P has an even length then $G[P \cup \{v_2, v_3\}]$ is a odd hole, otherwise $G[P \cup \{v_5\}]$ is a odd hole. □

The algorithm devised in [18] handles odd-holes with a special property called *amendable*. Let C be a hole in G, a vertex $v \in V(G - C)$ is called *C-major* if $V(N(v) \cap C) \not\subseteq V(H)$ where H is any induced P_2 of C. A hole C without C-major

vertices can be easily identified. In order to obtain such hole a "cleaning" technique, first introduced in [28], is used. Roughly speaking, *amendable holes* are holes which can be cleaned. To be more specific, a *cleaner* of a hole C is a vertex-set $X \subset V$ which contains all C-major vertices. A set X is anticonnected if $\overline{G}[X]$ is connected. Additionally, C is called *amendable* if it is the shortest odd-hole in G on at least 7 vertices and for every anticonnected set X of C-major vertices, there is an X-complete edge in C. The general idea is to generate a polynomial number k of cleaners, such that $S = X_1, ..., X_i, ..., X_k$, and test the graph $G[V - X_i]$ for clean holes. For practical reasons, [18] have applied some optimizations to this approach and are working with a variation of the cleaners, called *near-cleaners*. The algorithm is composed of three routines and is executed on both G and $\overline{G}$. There are two bottlenecks in this algorithm resulting in the $O(n^9)$ time-complexity, the first one is detecting a pyramid and the second one is detecting an amendable odd-hole (an order of $O(n^5)$ near-cleaners are generated and the time-complexity to detect a hole using a near-cleaner is $O(n^4)$).

Algorithm 1 Berge Graph Recognition

```
1: Detect one of five structures (two of which are the pyramid and the the jewel)
2: Generate O(n^5) near-cleaner sets.
3: for each near-cleaner set do
4:     check if a odd-hole is found
5: end for
```

Coloring Perfect Graphs

In the 1980s Grötschel at el. [53] demonstrated how the *ellipsoid method* can be applied to solve the coloring problem for perfect graphs in poly-time. For that end, they utilized the theta function $\Theta(G)$ introduced by Lovász in [68]. Their algorithm evaluates $\Theta(G)$, and by perfection we have $\Theta(G) = \alpha(G) = \theta(G) = \chi(\overline{G}) = \omega(\overline{G})$. Recently, there have been several attempts to devise a combinatorial coloring algorithm. Combinatorial in this context means an algorithm which relies on graph

searches, decomposition or linear programing [94]. The motivation of these studies is to solve those problems with the gathered knowledge about decompositions in perfect graphs. So far, most such algorithms have been restricted with respect to the allowed decompositions. For instance, [96] focused on perfect graphs which can be decomposed using 2-joins(*). In their work they showed that for a perfect graph G with no balanced skew partition(*), no connected non-path 2-join(*) in the complement and no homogeneous pair a maximum stable set and a maximum clique can be found in $O(n^6)$ time. Subsequently, an optimal coloring can be produced in $O(n^7)$ time. This work was extended in [20], where the restriction was loosened to accept any perfect graph with no balanced skew partition(*) and a $O(n^5)$ time algorithm to retrieve the maximum stable set and maximum clique was presented. The complexity for optimal coloring, on the other hand, remained the same. A further step in this direction was taken in [21] in which a combinatorial algorithm for coloring any perfect graph on n vertices in time $O(n^{(\omega(G)+1)^2})$ is given. This is technically polynomial assuming the graph has a bounded clique number.

2.2 Graph traversal

A graph traversal (or graph search [103]) of $G = (V, E)$ is the process of visiting all the nodes V in some order, two principal algorithms to accomplish this task are *Breadth First Search*(BFS) and *Depth First Search*(DFS). The logic guiding each search process produces a vertex-ordering of G. In this section we trace the development of *Lexicographic Breadth First Search* (LexBFS), a prominent graph traversal algorithm and an extension of the classic BFS.

Definition 2.2.1 *Let $G = (V, E)$ be a graph and $\{v, u\} \subseteq V$. The **distance** between v and u, denoted $d(v, u)$, is the length of the shortest chordless path with endpoints v and u (in terms of number of edges).*

2.2.1 Fundamentals

The *adjacency list* is the optimal graph representation (i.e. data structure) for BFS and DFS and is key for their linear time execution ($O(n+m)$). Conceptually BFS and DFS are very much alike: they both explore the graph gradually, in the sense that the algorithm will only reach a new vertex v if it has already encountered at least one of its neighbors (besides the first node, obviously). Furthermore, we can visualize the traversal pattern as a tree, as we always reach a vertex from a previously seen neighbor we can think of it as the parent node. The major difference between BFS and DFS, as implied by their names, lies in the order in which we choose the "next" vertex. For BFS we use a *first in first out* (FIFO) strategy, every time we process a vertex we *enqueue* all of its "new" neighbors and reach them in due time. Assume $s \in V$ is the vertex from which we start the search, BFS logic guarantees a vertex $v \in V$ will be processed before a vertex $u \in V$ if $d(s,v) < d(s,u)$. This property, combined with the traversal tree enables us to find the shortest path from s to v. BFS is indeed an appropriate and efficient solution for finding shortest paths and is widely used for this purpose. Other applications of BFS include cycle detection, spanning trees, and graph class recognition [30]. Algorithm 2 outlines the steps of BFS.

Algorithm 2 BFS

```
1: initialize queue with start vertex s
2: initialize a boolean array seen and mark s as seen
3: while queue not empty do
4:    pop v from queue
5:    for u ∈ N(v) do
6:       if u not seen then
7:          mark u as seen
8:          push u to queue
9:       end if
10:   end for
11: end while
```

Unlike the previous algoritm, DFS employs a "deep-dive" approach. The algo-

rithm follows a path of unvisited vertices until it reaches some vertex whose entire neighborhood has been visited, then it backtracks to the latest seen unvisited vertex and repeats the process (see Algorithm 3). Such behavior is achieved by a *last in first out* (LIFO) technique, characteristic to a *stack* data structure. Similar to BFS, DFS can be used to solve problems like cycle detection and spanning trees however it was popularized for settling connectivity problems [89].

Algorithm 3 DFS

```
1: initialize stack with start vertex s
2: initialize a boolean array visited
3: while stack not empty do
4:    pop v from stack
5:    if v not visited then
6:       for u ∈ N(v) do
7:          if u not visited then
8:             push u to stack
9:          end if
10:      end for
11:      mark v as visited
12:   end if
13: end while
```

Before commensing the discussion on LexBFS, we would like to mention as a side note another commonly used traversal variation, namely the *Maximum Cardinality Search* (MCS). Unlike BFS which progresses to the earliest seen and unvisited vertex or DFS which chooses the latest seen yet unvisited one, MCS decides on the next vertex based on a *seen-count*, the vertex with the most neighbors in the "visited pile" is picked next. MCS was introduced by Tarjan and Yannakakis [93] as a simplification of LexBFS for the tasks of chordality testing and checking acyclicity of *hypergraphs* (these generalize graphs by allowing edges to contain more than two vertices [2]).

2.2.2 LexBFS

Origin and Chordal Graphs

LexBFS first appeared in [85] where the authors studied the problem of *Minimum Fill-In*. This problem is closely related to *Gaussian elimination* which is a technique from linear algebra to solve a system of linear equations, the method simplifies the coefficients matrix M (of size $n \times n$) by eliminating the variables $x_1, ..., x_n$ from as many rows as possible. The fill-in of a graph corresponds to the new non-zero elements produced by the elimination process (a comprehensive review on this representation can be found in [90]). To discuss the problem we need few additional definitions.

Definition 2.2.2 *Let $G = (V, E)$ be a graph and $v \in V$.*

- *The **deficiency** of v, denoted $D(v)$ is a set of edges corresponding to all the non-edges in the neighborhood of v. Formally, $D(v) = \{u_1u_2 | \{u_1, u_2\} \subseteq N(v), u_1u_2 \notin E\}$.*

- *The graph G_v is a **v-elimination** graph, if it represents the elimination process of v. This graph is generated by adding all the edges $D(v)$ and removing v, i.e. $G_v = (V', E')$ with $V' = V - \{v\}$ and $E' = E(G[V']) \cup D(v)$.*

- *An **elimination process** with order σ is the process of eliminating all vertices of G according to σ, using the i-th elimination graph as the input for the $\{i+1\}$ elimination step.*

- *The **fill-in** of G with respect to σ is the result of the elimination process and is defined as $F(G_\sigma) = \bigcup_{i=1}^{n-1} D(\sigma_i)$ (the deficiency $D(\sigma_i)$ is computed here on the elimination graph of the $(i-1)$-th step).*

- *The **elimination graph** of $G = (V, E)$ with ordering σ is $G^*_\sigma = (V, E \cup F(G_\sigma))$.*

The minimum fill-in problem requires finding an ordering σ such that $F(G_\sigma)$ contains the smallest number of edges with respect to any other ordering of V. The authors of [85] conjecture this problem might be NP-complete (as was actually proven later in [104]) and therefore they focus on minimal fill-in instead.

Definition 2.2.3 *A **minimal fill-in** is a fill-in $F(G_\sigma)$ such that for any other ordering α, $F(G_\alpha) \not\subset F(G_\sigma)$. An ordering σ for which $F(G_\sigma)$ is a minimal fill-in is called a minimal elimination order (MEO). A **perfect elimination order**(PEO) is an ordering σ for which $F(G_\sigma) = \emptyset$. Graphs that admit PEO are **perfect elimination graphs**.*

Clearly every elimination graph G^*_σ is a perfect elimination graph since the order σ is a perfect elimination order of this graph (it cannot contribute any more edges to a fill-in). In addition we note that PEO is both a MEO and minimum elimination order. The following theorem establishes a PEO is an efficient recognition technique for chordal graphs.

Theorem 2.2.4 *[84] A graph G is a perfect elimination graph if and only if it is chordal.*

For that reason G^*_σ is called a *triangulation* of G or a *minimal triangulation* if σ is a MEO. A particularly interesting observation about a PEO is that it implies that every σ_i is simplicial in $G[\{\sigma_i, ..., \sigma_n\}]$, making it a powerful tool for solving graph problems on chordal graphs [50]. The next result defines a sufficient condition for a minimal triangulation.

Theorem 2.2.5 *[85] Let $G = (V, E)$ be a graph and $G' = (V, E \cup F)$ be chordal. Then G is a minimal triangulation if and only if each $f \in F$ is a unique chord of a C_4 in G'.*

Algorithm 4 LexM

```
initialize labels list of size n
for i : n → 1 do
    choose the vertex v with the largest label
    set σ_i = v
    for unnumbered u ∈ V such that there is a path P = {v, u_1, ..u_k, u} with u_j
    unnumbered and labels[u_j] < labels[u] do
        labels[u] += i {e.g. add i to the set labels[u]}
    end for
end for
```

Rose et al. [85] showed Algorithm 4 produces an order that satisfies the requirement in Theorem 2.2.5, meaning every order produced by Lex-M is MEO. They also demonstrated a $O(nm)$ time implementation of Lex-M and that constructing a minimal triangulation using the produced order can be done in linear-time. A further result was due to the next observation. Note, from this point onwards we use G_i as per Definition 1.2.1.

Observation 2.2.6 *[85] Let $G' = G^*_\sigma$ be the elimination graph of $G = (V, E)$. The label for every $v = \sigma_i$ contains all of the neighbors of v in G'_i. In other words, let $L(v)$ be the final label of v, then $L(v) = N_{G'}(v) \cap V(G'_i)$.*

For a MEO of a chordal graph G (and equally, as we have noted, a PEO) Observation 2.2.6 implies that once we pick the vertex v with the largest label it is sufficient to only update its neighbors. Thus Rose, Tarjan and Lueker were able to give a simplified linear-time algorithm to produce a PEO assuming one exists. Their algorithm, Lex-P, is now commonly called LexBFS and is detailed in Algorithm 5.

Theorem 2.2.7 *[85] Chordal graphs can be recognized in time $O(n + m)$*

Proof. To check that G is chordal by Theorem 2.2.4 it is enough to check that G has a PEO. First, execute LexBFS on G in $O(n + m)$ time to find an elimination order $\sigma = \{v_1, ..., v_n\}$. If G is chordal then σ is inevitably a PEO, the following is

Algorithm 5 LexBFS

```
initialize labels list of size n
for i : n → 1 do
    choose the vertex v with the largest label
    set σ_i = v
    for unnumbered u ∈ N(v) do
        labels[u] += i {e.g. add i to the set labels[u]}
    end for
end for
```

an explanation of how to check that in linear time. Initialize a list *bba* ("better be adjacent") of size n. Now we examine σ from left to right. For each v_i $(1 \leq i \leq (n-1))$, start by checking that it satisfies its adjacency requirements, i.e. $bba[i] \subseteq N_\sigma(v_i)$. Then, identify the smallest $v_j \in N_\sigma(v_i)$ (w.r.t. the index in σ) and assign $bba[j] = set(bba[j] \cup (N_\sigma(v_i) - \{v_j\}))$. □

Further Applications

Since the original paper more variations and applications for LexBFS were introduced. The research branches out to other graph families so we briefly define them first. We say a path P *misses* a vertex v if $N(v) \cap V(P) = \emptyset$.

Definition 2.2.8 *A vertex v is **semisimplicial** if v is not an interior vertex of an induced P_4. An ordering $\sigma = \{v_1, ..., v_n\}$ of V is a **semiperfect elimination ordering** if v_i is semisimplicial in G_i. The term **HHD** is short for the graphs: hole, **House** ($\overline{P_5}$) and **Domino** (a graph composed of two C_4s with a shared edge).*

Definition 2.2.9 *The vertices v_1, v_2 are **unrelated** w.r.t. v if there exist P_1, P_2 with endpoints v, v_1 and v, v_2 respectively such that P_1 misses v_2 and P_2 misses v_1. The vertex v is called **admissible** if there are no unrelated vertices w.r.t. v. An ordering σ is an **admissible elimination ordering** if v_i is admissible in G_i. An independent set $\{v_1, v_2, v_3\} \subset V$ is an **Asteroidal Triple** (AT) if between every pair there is a path P which misses the third vertex.*

LexBFS is an effective tool to recognize graph families beyond chordal graphs. As demonstrated by several works, specific restricted graph classes are characterized by special orderings which are produced by any LexBFS.

Theorem 2.2.10 *[57] G is an HHD-free graph if and only if every ordering produced by LexBFS is a semiperfect elimination ordering.*

Theorem 2.2.11 *[33] G is an AT-free graph if and only if every ordering produced by LexBFS is an admissible elimination ordering.*

In addition, it was noted in [55] that the orderings produced by LexBFS embody a special case of partitioning. This observation instigated algorithms based on multiple "sweeps" such as LexBFS+ and LexBFS$^-$. Let σ be a LexBFS ordering of G. LexBFS+ breaks the ties in line 3 of Algorithm 5 according to σ and is used in [34] to recognize *Interval graphs* (these are AT-free chordal graphs [66]) in a simple linear fashion. LexBFS$^-$ produces an ordering of $\overline{G}$ using σ and eliminates the need to explicitly represent $\overline{G}$. It was also used in [12] to efficiently recognize *Cographs* (graphs with no induced P_4).

2.3 Decomposition and Separability of Graphs

Decompositions are a substantial subfield of graph theory. This research area includes modular, split and star-cutset decompositions, to name a few. Roughly speaking the connecting thread of all techniques is partitioning the graph in order to simplify it. Some methods, as does the one we plan to study, exploit cutsets and separate the graph into multiple smaller graphs. A similar, yet more informative, way to define a cutset is as a *separator*.

Definition 2.3.1 *Let $G = (V, E)$ be a graph and $u, v \in V$. A set S is a* ***uv-separator*** *of G if the vertices u and v belong to different components A and B in $G - S$. Every uv-separator is also a cutset with $u \in A$ and $v \in B$.*

In this chapter we focus on the clique separator decomposition and provide an overview of some other popular decompositions. We group the various decompositions into two types based on whether cutsets are utilized.

Definition 2.3.2

- *A **decomposition step** is an operation on graph G resulting in a set of smaller graphs (with respect to number of vertices).*
- ***Decomposition blocks** are the graphs produced by a decomposition step.*
- *A **decomposition** is the process of repeatedly performing decomposition steps until the decomposition method is no longer applicable.*
- *A **decomposition tree** is a logical (and visual) representation of a decomposition.*
- *An **atom** is the graph corresponding to a leaf of a decomposition tree.*

In some cases we mention *composition*, this is the opposite operation of a decomposition step. Simply put it is an operation of "gluing" or "bonding" two graphs together.

2.3.1 Clique Separator Decompostion

A uv-separator S is a *clique separator* (and therefore a *clique cutset*) if S induces a clique. A graph G is called *decomposable* if it contains a clique cutset. Let us begin by redefining some of the general terms with respect to the clique separator decomposition. All subgraphs in this section are induced subgraphs.

Definition 2.3.3

- *Let G be a decomposable graph and C be a clique cutset of G. A decomposition step is the partitioning of $V(G-C)$ into two sets A and B, such that there is*

no edge between A and B. We define the decomposition blocks G_1, G_2 as follows $G_1 = G[A \cup C]$ and $G_2 = G[B \cup C]$. We denote a decomposition step by the function $DS(G, C, G_1, G_2)$.

- *The decomposition tree $T(G)$ is a binary tree representing a decomposition. Each inner node is labeled according to the decomposition step it represents and (apart from the root node) corresponds to a previous decomposition block. The leaves of the tree are non-decomposable subgraphs of G and are known as **atoms**. We use the notation atoms($T(G)$) for the set of subgraphs at the bottom of a decomposition tree.*

- *Let H be a subgraph of G. H is a **prime-subgraph** of G if and only if H is a connected subgraph which has no clique cutsets. H is a maximal prime-subgraph (**mp-subgraph**) if and only if H is a prime-subgraph and there exists no H' such that $H \subset H'$ and H' is a prime-subgraph. We use mp-subgraphs(G) to refer to the set of all mp-subgraphs of G.*

The clique separator decomposition, similar to most other decompositions we shall present, was motivated by the research on perfect graphs. Clique cutsets were noted as "useful" to the study of perfection since a composition of two perfect graphs using a clique preserves perfection (i.e. this operation results in a perfect graph) [94]. At the same time, certain perfect graph families are famously decomposable. For instance, it is well known that every minimal cutset of a chordal graph G is a clique [41, 84]. Gavril [51] tried to aggregate graphs with clique cutsets and special types of atoms (Type 1: the join of a biparite graph and a clique, Type 2: k-partite graphs) under a class named *Clique Separable Graphs*. He presented an algorithm for recognizing graphs in this family together with a method to find a maximum clique and minimum coloring (utilizing the decomposition tree).

For an arbitrary graph G, an $O(nm)$ time algorithm to detect a clique cutset was given by Whitesides [101]. A complete clique cutset decomposition is achieved by re-running the algorithm on all decomposition blocks, which are roughly bounded by n^2 [51], and amounts to $O(n^3m)$ time. Tarjan [91] noticed the advantage of a MEO to this problem and suggested an alternative algorithm. His algorithm also requires $O(nm)$ time to find a clique cutset, however this is also the time required to find *all* the clique cutsets. Tarjan generalized Garvil's algorithms for maximum clique and minimum coloring and answered Garvil's open question regarding maximum stable set. Even though the proposed algorithm was shown to produce at most $n-1$ atoms, Tarjan raised the question regarding the uniqueness of atoms in an arbitrary clique cutset decomposition. This problem was tackled by Leimar [65] who introduced the P-decomposition, which is a clique cutset decomposition with the property that at each decomposition step $DS(G, C, G_1, G_2)$, C must be a minimal clique separator of G. Let us define that more precisely.

Definition 2.3.4 *Let $G = (V, E)$ be a graph. S is a **minimal clique separator** if and only if S is a clique cutset and there are $\{u, v\} \in V$ such that S is a uv-separator and there exists no $S' \subset S$ which is also a uv-separator.*

Liemer [65] proved the following two theorems and thus demonstrated the smallest possible set of atoms of any clique separator decomposition is exactly the set of mp-subgraphs(G).

Theorem 2.3.5 *[65] Let G be a decomposable graph and $T(G)$ be any decomposition of G. Then mp-subgraphs$(G) \subseteq$ atoms$(T(G))$.*

Theorem 2.3.6 *[65] Let G be a decomposable graph and $T(G)$ be any minimal clique separator decomposition of G. Then mp-subgraphs$(G) =$ atoms$(T(G))$.*

Finding Clique Cutsets

Whiteside's [101] algorithm for finding clique cutsets is a bottom up approach. As described in Algorithm 6, we iteratively grow a non-decomposable subgraph S of G. We start from a subgraph S which is hole of G (hence, clearly prime) and enlarge it until a clique cutset is found or we reach $S = G$ (recall, a graph with no holes is chordal and any cutset of such graph is a clique cutset). Once a clique cutset C if found the algorithm may be repeated on the decomposition blocks of $DS(G, C, G_1, G_2)$.

Algorithm 6 Find clique cutset - Whitesides

```
1: initialize S with a hole of G
2: while |S| < |G| and clique cutset not found do
3:    Let C be a connected component of G − S
4:    Compute R = N_G(C) ∩ S
5:    if R is a clique then
6:       R is a clique cutset
7:    else
8:       Find a path P between two non-adjacent vertices {u, v} ⊆ R with P ⊆ C
9:       Increase S = S ∪ P
10:   end if
11: end while
```

Tarjan's algorithm uses a minimal triangulation (presented in Section 2.2.2) of G and relies of the observation that for a MEO σ any clique cutset of G^*_σ is a clique cutset of G. This property is a result of the next theorem.

Theorem 2.3.7 *[91] Let $G = (V, E)$ be a graph and σ be a MEO of G generated by Lex-M (see Algorithm 4). For any decomposition by clique separators, every edge $uv \in F(G_\sigma)$, is such that a unique atom contains both u and v.*

Using the fact that σ is a PEO of G^*_σ and that every cutset of a chordal graph is a clique [41, 84], by scanning the vertices of G according to σ and checking whether $N_\sigma(v_i)$ is a clique we find all clique cutsets of G. The correctness of this algorithm is proven in [91]. Leimer [65] gave a comprehensive mathematical analysis of Tarjan's algorithm and the minimal clique separator decomposition. He proved that

the optimal (smallest) number of atoms is produced when the MEO is ordered in a particular way, called *D-numbering*. He showed that Lex-M does in fact generate such an order, while other minimal traingulation algorithms may not (e.g. [77]). He also suggested an optimization to the second step of Tarjan's algorithm. Instead of trying to apply the decomposition step to all vertices in σ, Leimer was able to extract better candidates. A recent variation of this idea was presented by Berry [6], she demonstrated a MCS-based minimal triangulation algorithm which also produces an effective *D-numbering* order.

Applications of Clique Cutset Decomposition

As previously mentioned, for a decomposable graph G, the decomposition tree can be used to find an optimal coloring, a maximum weighted clique and a maximum stable set. The following propositions assume we are able to solve those problems on the atoms of G. We denote by $O(F(Atoms))$ the time complexity required to do so (that is $O(n)\cdot$ *time per atom*). It is important however to mention these problems may remain NP-hard for the atoms. We rely on the fact that Tarjan's decomposition produces a right skewed binary tree with at most $n-1$ atoms and at most $n-2$ inner nodes.

Proposition 2.3.8 *[91] A graph G can be colored in time $O(n^2) + O(F(Atoms))$.*

Proof. First we color all leaves in time $O(F(Atoms))$. We now work our way up the decomposition tree, starting at the lowest inner node. The general coloring step consists of coloring the current node utilizing its two optimally colored children. Let the current inner node correspond to $DG(G', C', G'_1, G'_2)$ and let $k = max(\chi(G'_1), \chi(G'_2))$. Then we can color G' with k colors by identifying the colors of G'_1 and G'_2 with respect to C'. Perform this repeatedly until we reach the root of the tree. We have $O(n)$ such compositions to perform and the cost of recoloring is also $O(n)$, resulting $O(n^2)$ time. □

Proposition 2.3.9 *[91] A a maximum-weighted clique of G can be found in time $O(F(Atoms))$.*

Proof. Any clique K of G is obviously prime and we expect to find any prime subgraph of G in one of its atoms (see Observation 3.1.3). Therefore, this problem reduces to finding the maximum weighted clique for all atoms and then merely choosing the largest one. □

The key idea to computing the maximum-weighted independent set of G is that any clique cutset may contribute at most one vertex to any independent set of G. The method proposed here requires us to solve $O(n)$ variations of the problem on each atom. Hence, in this case $O(F(Atoms))$ is of order $O(n^2)\cdot$ *time per atom.*

Proposition 2.3.10 *[91] A a maximum-weight independent set of G can be found in time $O(n^2) + O(F(Atoms))$.*

Proof. We solve this problem recursively by traversing the skewed tree from the top down. The function $w(I)$ is the weight of a set I. Let $DS(G, C, G_1, G_2)$ be the decomposition step at the root of $T(G)$, and let us label the vertices of C by $\{v_1, ..., v_l\}$. According to Tarjan's decomposition one of G_1, G_2 is an atom, and let it be G_1. We start by computing the maximum-weight independent set of $(G_1 - C)$ denoted I, along with l more sets I_j $(1 \leq j \leq l)$ such that I_j is the maximum weight stable set of $G_1 - N_{G_1}[v_j]$. The next step is to adjust the weights of $v_j \in G_2$ so they reflect the potential gain or loss in choosing v_j as part of the maximum weight independent set of G_2 (w.r.t. G_1). We assign v_j the weight $w(v_j) + w(I_j) - w(I)$. Finally, we solve the problem on the updated G_2 recursively. Let $\alpha(G_2)$ be the maximum weighted stable set of G_2, if $v_j \in \alpha(G_2)$ then $\alpha(G) = I_j \cup \alpha(G_2)$ otherwise $\alpha(G) = I \cup \alpha(G_2)$. □

For certain graph families the maximum independent set problem is relatively easy to solve. We can therefore define classes whose atoms are in such families and obtain results of the following type.

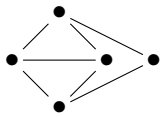


Figure 2.2: Q graph

Theorem 2.3.11 *[9] The maximum independent set problem can be solved in time $O(n^4m)$ on graphs whose atoms are (P_5, Q)-free. (see Fig. 2.2)*

Theorem 2.3.12 *[9] The maximum independent set problem can be solved in time $O(n^3m)$ on graphs whose atoms are (P_6, C_4)-free.*

2.3.2 Other Cutset Decompositions

In cutset-based decompositions a prime subgraph is an induced subgraph that does not admit the respective cutset. Cutsets chosen for decompositions usually exhibit special properties, as is the case with clique cutsets. Another example is the *stable cutset*. A stable cutset is a stable set S for which $G - S$ is disconnected, it was studied in relation to perfect graphs in [98, 32]. Unfortunately this type of cutset is not particularly useful since it is NP-complete to detect [11]. We start this survey from the *star cutset*, which is a generalization of the clique cutset and follow with the even more general *skew partition*. Recall, skew partition decomposition appeared in Theorem 2.1.16, which popularized it along with the *2-join*.

Definition 2.3.13 *A graph G has a **star cutset** C if there is a vertex $v \in C$ such that C is a cutset of G and v is adjacent to all vertices of $C - v$.*

Definition 2.3.14 *A **skew partition** is a partitioning of V into four nonempty sets: A, B, C, D such that every vertex in A is adjacent to every vertex in B and no vertex in C has a neighbor in D.*

A skew partition insures that the set $A \cup B$ is a cutset of G while $C \cup D$ is a cutset of $\overline{G}$. It easy to see the skew partition generalizes the star cutset which is a

skew partition where one of A, B has size one. The decomposition blocks of a skew partition decomposition are $G_1 = G[A \cup B \cup C]$ and $G_2 = G[A \cup B \cup D]$. In [23] it was conjectured that no minimal imperfect graph has a skew partition. However, so far this is only known to be true for the *balanced skew partition*, and was demonstrated in the SPGT proof [19].

Definition 2.3.15 *A **balanced skew partition** is a skew partition A, B, C, D such that:*

- *For every non-adjacent pair $\{u, v\} \subseteq A$ or $\{u, v\} \subseteq B$ any induced path with endpoints u, v and interior in C or D has an even number of edges.*
- *For every adjacent pair $\{u, v\} \subseteq C$ or $\{u, v\} \subseteq D$ any induced antipath with endpoints u, v and interior in A or B has an even number of edges.*

As for recognizing these types of cutsets, from [61] we know skew partitions can be found in time $O(n^4m)$ while recognizing balanced skew partitions is NP-hard [95]. Nonetheless, for a Berge graph the existence of a balanced skew partition can be certified in $O(n^5)$ time [95, 15].

The 2-join decomposition emerged from studies regarding perfection-preserving compositions. This is in fact a generalization of the *split-decomposition* (or *1-join*) and a special case of the *2-amalgam* structure [38, 37, 35].

Definition 2.3.16 *A connected graph G has a **2-join** if V can be partitioned into V_1, V_2 with vertex-disjoint non-empty subsets A_i, B_i in V_i $(i = 1, 2)$, such that:*

- *every vertex of A_1 is adjacent to every vertex of A_2 and every vertex of B_1 is adjacent to every vertex of B_2*
- *there are no other edges between V_1 and V_2 and $|V_i| \geq 3$ for $i = 1, 2$.*

In a 2-join decomposition the decomposition blocks are $G_1 = G[V_1 \cup P_2]$ and $G_2 = G[V_2 \cup P_1]$, where P_1 and P_2 are *marker paths* as defined in [18]. A *path 2-join* is a 2-join for which some $G[V_i]$ is an induced path. Appropriately, a *non-path 2-join* is a 2-join which is not a path 2-join, this was defined to facilitate detection. The fastest algorithm known to identify a non-path 2-join is given in [15] with execution time of $O(n^2m)$.

2.3.3 Partitioning

While cutset-based decompositions allow us to "trim" the graph, in certain cases we may rather benefit from extracting special subgraphs. The *Modular decomposition* is just that case. It is a well studied and commonly used approach (as evident from its many names: substitution decomposition, disjunctive decomposition and X-join [76]). We first need to define a module:

Definition 2.3.17 *Let $G = (V, E)$ be a graph. A **module** of G is a subset of vertices $M \subseteq V$ such that for every $\{v, u\} \subseteq M$ we have $N(v) - M = N(u) - M$.*

Modules also appear in literature as closed sets, clans, autonomous sets and partitive sets [54]. A module is called *trivial* if it is an empty set, a single vertex or the entire vertex-set V. A non-trivial module is called a *homogeneous set*. Here a graph is called prime if it contains no homogeneous sets. Let H be a homogeneous set of a graph G. In a modular decomposition, the decomposition blocks are the two graphs: H and $G[(V(G)-H)\cup\{h\}]$ for a vertex $h \in H$. Note that $G[(V(G)-H)\cup\{h_1\}]$ is isomorphic to $G[(V(G) - H) \cup \{h_2\}]$ for any two vertices h_1, h_2 in H. If either of the graphs H and $G[(V(G) - H) \cup \{h\}]$ is not prime, then it is recursively decomposed. At the bottom of the decomposition tree (i.e. the leaves) we find all the unique vertices. The modular decomposition can be performed in linear time [36, 72] making it a robust tool for optimization problems (as it has practically no cost). Applications for this decomposition may be found in [76].

2.4 Ordered Graphs

A central topic of this work is exploring a SEO, that is an ordering σ with the property that σ_i is simplicial extreme in G_i. This idea is clearly inspired by the classical PEO. In this section we examine two graph families which also demonstrate inherently useful orders, these are *β-perfect* graphs and *perfectly orderable graphs*.

2.4.1 β-Perfect

Let $G = (V, E)$ be a graph and $\delta(G)$ be the minimum degree of any $v \in V$. We define $\beta(G) = max(\delta(H) + 1)$ where H is an induced subgraph of G. The relation to the chromatic number is known to be $\chi(G) \leq \beta(G)$, by virtue of the analysis made in [88]. The following definition of β-perfect graphs is somewhat akin to that of the perfect graph class [69].

Definition 2.4.1 *A graph G is **β-perfect** if for every induced subgraph H we have $\chi(H) = \beta(H)$.*

A favorable trait of β-perfect graphs and a core reason for their conception is the fact they can be colored in linear time by a greedy algorithm and a *smallest last ordering* (that is an ordering σ in which σ_i has the smallest degree in G_i) [88, 71, 69]. It is easy to check that even holes are not β-perfect and as a matter of fact no β-perfect graph can contain an even hole. This observation is one of the motivators for the study of (Even-Hole)-free graphs. We note, on the other hand, that (Even-Hole)-free graphs are not necessarily β-perfect. Take the graph W_5 for example, it contains no even hole but it is not β-perfect as $4 = \chi(W_5) > \beta(W_5) = 3$. The following is an important structural result about β-perfect graphs.

Theorem 2.4.2 *[69] G is β-perfect if it contains no even-hole and no even antihole.*

Class	β-perfect	
(Even-hole, Diamond, Cap on six vertices)-free	Yes	2000 [46]
(Even-hole, Diamond, Net)-free	Yes	2002 [60]
(Even-hole, Diamond)-free	Yes	2009 [63]
(Even-hole, Kite)-free	Yes	2018 [48]
(Even-hole, Claw)-free	No	[60] gives forbidden structures
(Even-hole, Cap)-free	No	[100] gives an example

Table 2.2: Partial Results for β-perfect graphs. *No* means not necessarily β-perfect

A *Cap* is a hole with a single chord between two vertices at distance two. A *Net* is a graph on six vertices $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ where $\{v_1, v_2, v_3\}$ is a triangle and the other edges are exactly v_1v_4, v_2v_5, v_3v_6. The complete characterization of (Even-Hole)-free β-perfect graphs is still open, Table 2.2 depicts some partial results (in chronological order).

Lastly, the next theorem demonstrates the relation between β-perfect graphs and simplicial extremes.

Theorem 2.4.3 *[69] A minimal β-imperfect graph that is not an even hole, contains no simplicial extreme.*

2.4.2 Perfectly Orderable Graphs

The family of *perfectly orderable graphs* is characterized by the existence of a *perfect* ordering, let us define that.

Definition 2.4.4 *An ordering σ of a graph G is called **perfect** if for each induced subgraph H the greedy algorithm (see Section 1.2.2) produces an optimal coloring using the sub-order of σ that contains $V(H)$. If G has a perfect order then G is **perfectly orderable**.*

The special requirement in this definition encourages one to observe the corresponding oriented graph, that is the directed graph in which an edge uv has direction

$u \to v$ if u appears before v in σ (or $\sigma(u) < \sigma(v)$). Chvátal [22] presented the following analogous definition of perfectly orderable graphs and showed they are perfect.

Definition 2.4.5 *A graph G is perfectly orderable with order σ if it does not contain a chordless path $P = \{v_1, v_2, v_3, v_4\}$, with edges v_1v_2, v_2v_3, v_3v_4, such that $\sigma(v_1) < \sigma(v_2)$ and $\sigma(v_4) < \sigma(v_3)$.*

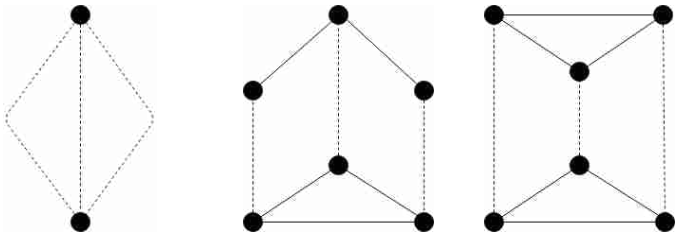
Theorem 2.4.6 *[22] A perfectly orderable graph is perfect.*

By definition, a graph G with a perfect order σ, can be optimally colored in linear time. The paper [25] gives a linear time algorithm to find a largest clique in a perfectly ordered graph. However, it is known that checking whether an arbitrary graph has a perfect order is NP-complete [75]. For this reason, much of the research concentrates on easily recognizable sub-families, including some famous classes such as chordal graphs and interval graphs.

2.5 Hole-free Graphs

Once the SPGC was settled and perfect graphs become fairly well characterized, it was only natural to extend the research to broader families of graphs, namely hole-free graphs. The class of (Odd-Hole)-free graphs is an obvious generalization of Berge graphs, at the same time graphs without even holes are an interesting counterpart and (as we have already seen) are closely related to the β-perfect family.
Within the study of (Even-Hole)-free and (Odd-Hole)-free graphs, the *3-path configurations* (abbriviated 3PCs) are predominant structures which are commonly encountered [29]. The 3PC structure-set, illustrated in Fig. 2.3, consists of the three types: $3PC(.,.)$ (*called theta* [17]), $3PC(\Delta,.)$ (*called pyramid* [18]) and $3PC(\Delta,\Delta)$ (*called prism*[17]). The pyramid was previously introduced in relation to perfect graph recognition (see Section 2.1.3). A graph G is said to have a theta, specifically $3PC(x,y)$,

if there are two vertices x and y with three paths P_1, P_2, P_3 between them. Graph G contains a prism with two disjoint triangles $\{x_1, x_2, x_3\}$ and $\{y_1, y_2, y_3\}$ if there exists a path P_1 from x_1 to y_1, a P_2 from x_2 to y_2 and a P_3 from x_3 to y_3 (to be more explicit we say G has a $3PC(x_1x_2x_3, y_1y_2y_3)$). For both theta and prism, any two paths must induce a hole, i.e. the condition $G[V(P_i) \cup V(P_j)]$ is a hole for $\{P_i, P_j\} \subset P_1, P_2, P_3$ ought to be satisfied.



(a) 3PC(.,.) (b) 3PC(Δ,.) (c) 3PC(Δ,Δ)

Figure 2.3: 3-path configurations. Solid lines are edges, dashed lines are paths with length at least one.

In a manner similar to that of Theorem 2.1.19 it can be shown an (Even-Hole)-free graph may not contain a theta or a prism. Let us demonstrate that for the 3PC(Δ, Δ).

Observation 2.5.1 *A graph G containing a 3PC(Δ, Δ) has an even hole.*

Proof. Let G contain a 3PC($x_1x_2x_3, y_1y_2y_3$), and let P_1 be a path from x_1 to y_1, P_2 be a path from x_2 to y_2 and P_3 be a path from x_3 to y_3. By the definition of 3PC we have the following holes: $H_1 = G[V(P1) \cup V(P2)]$, $H_2 = G[V(P_1) \cup V(P_3)]$ and $H_3 = G[V(P_2) \cup V(P_3)]$. For H_1 and H_2 to be odd the parity of $|P_1|$ should differ from that of $|P_2|, |P_3|$. Without loss of generality, we choose $|P_1|$ to be even, meaning $|P_2|$ and $|P_3|$ are odd. But now H_3 is an even hole. □

Following the methodology used to prove the SPGT, it became customary to describe graph classes as *basic* or decomposable via certain cutsets. This is also the best known characterization of (Even-Hole)-free and (Odd-Hole)-free graphs. We will not specify the exact basic classes here as they tend to be complex and are not

necessary for this thesis. Along those lines, a recent study into graphs with restricted *Trumper configurations* (these are the 3PCs and a variation of the wheel) have yielded nice decomposition and structural results [8]. Conforti et al. [26] gave the following characterization of graphs without odd holes.

Theorem 2.5.2 *[26] If G is an (Odd-Hole)-free graph, then G is either basic or G has a double star cutset or a 2-join.*

An equivalent way to express whether a graph has an odd or even hole is by *signing* it. A signed graph is a graph with the values 0 and 1 assigned to its edges. A graph is *odd-signable* (resp. *even-signable*) if there exists a signing such that every triangle has odd weight and every hole has odd (resp. even) weight. A graph is odd-signable if it is (Even-Hole)-free and is even-signable if it is (Odd-Hole)-free [29]. Da Silva and Vušković [87] described the structure of (Even-Hole)-free graphs in terms of odd-signable graphs and provided a $O(n^{19})$ time recognition algorithm for them (this was improved to $O(n^{11})$ time by [14]).

Theorem 2.5.3 *[87] A connected 4-hole-free odd-signable graph is either basic or it has a 2-join or a star cutset.*

Farber [45] noted that C_4-free graphs have a most $O(n^2)$ maximal cliques, this entails a polynomial algorithm to find the maximum clique. Another nice property of (Even-Hole)-free graphs relates the clique number to the chromatic number.

Theorem 2.5.4 *[1] If a graph G is (Even-Hole)-free, then G contains a vertex whose neighborhood can be partitioned into two cliques. In particular, G satisfies $\chi(G) \leq 2\omega(G) - 1$.*

Class	coloring	clique	independent set	clique cover
(Even-Hole)-free	?	P	?	NP-hard
(Odd-Hole)-free	NP-hard	NP-hard	?	NP-hard

Table 2.3: Complexity for problems on hole free classes [100, 64]

Table 2.3 summarizes the known results for the fundamental graph problems of coloring, maximum clique, maximum independent set and clique cover on (Even-Hole)-free and (Odd-Hole)-free graphs.

Chapter 3

Clique-Cutset Decomposition

In this chapter we conduct a thorough examination of the clique cutset decomposition. We rely on the definitions presented in Section 2.3.1 coupled with few extra notations. In general, decompositions are not unique since at each node of the decomposition tree we are at liberty to pick any cutset. To illustrate this claim, let $T(G)$ be a decomposition tree of G and N be a node labeled $DS(G', C, G'_1, G'_2)$. The cutset C is chosen out of a set containing all clique cutsets of G'. Our goal is to observe how this choice effects the corresponding subtree in terms of further available cutsets and atoms formed. Furthermore, we intend to examine various types of decompositions which impose different constraints over the choice of C. We start by examining the properties of a general clique cutset decomposition in Section 3.1 and Section 3.2. Section 3.3 performs a similar study for the minimal clique cutset decomposition and Section 3.4 analyzes the maximal clique cutset decomposition.

3.1 Decomposition Properties

Definition 3.1.1 *Let G be a graph and C be a clique cutset of G. A **decomposition path**, denoted $P(N)$, is a path in $T(G)$ from the root to a node N, that consists of all decomposition blocks on the way. For instance, a path to an atom A is $P(A) =$*

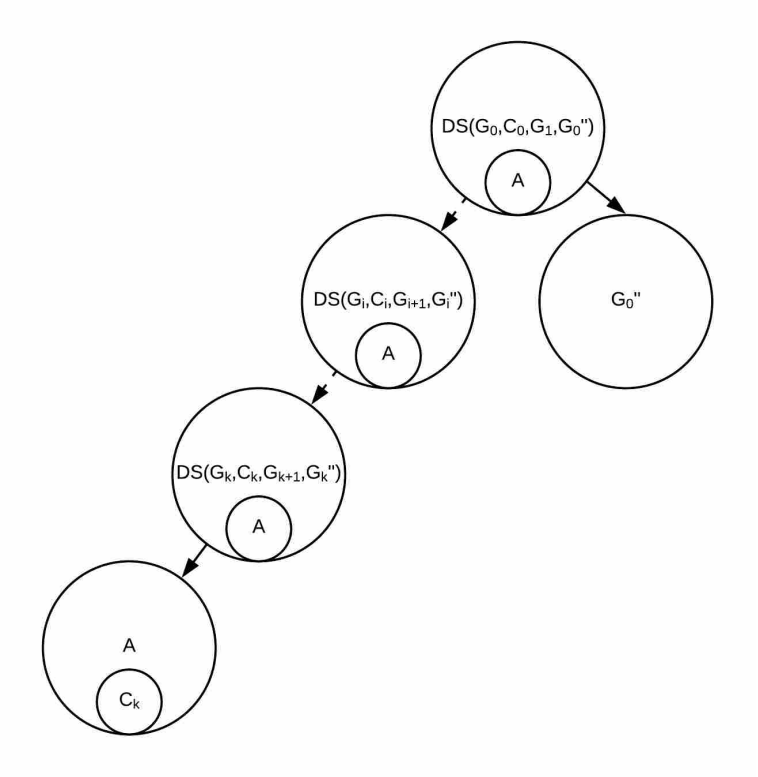


Figure 3.1: Illustration of a decomposition path in $T(G)$

$(G = G_0, ..., G_k, G_{k+1} = A)$. *The decomposition blocks of each* G_i *are denoted as* G'_i *and* G''_i, *and* $G'_i = G_{i+1}$ *(see Fig. 3.1).*

So far we used the terms cutset and separator almost interchangeably, however on some occasions we should be more careful. With the following observation we emphasis the difference between a minimal clique separator and a minimal clique cutset (sometimes called inclusion minimal separator [62]). To avoid confusion, we use the notation S for separators and C for cutsets.

Observation 3.1.2 *Every minimal clique cutset* C *of* G *is a minimal separator of* G, *the converse is not necessarily true (see Fig. 3.2). For every minimal clique separator* S *there exists a minimal clique cutset* C *such that* $C \subseteq S$. *Particularly, if* S *is not a minimal clique cutest then there is a minimal clique cutset* C *such that* $C \subset S$.

By definition, the atoms of any $T(G)$ are prime subgraphs of G. It is also trivial

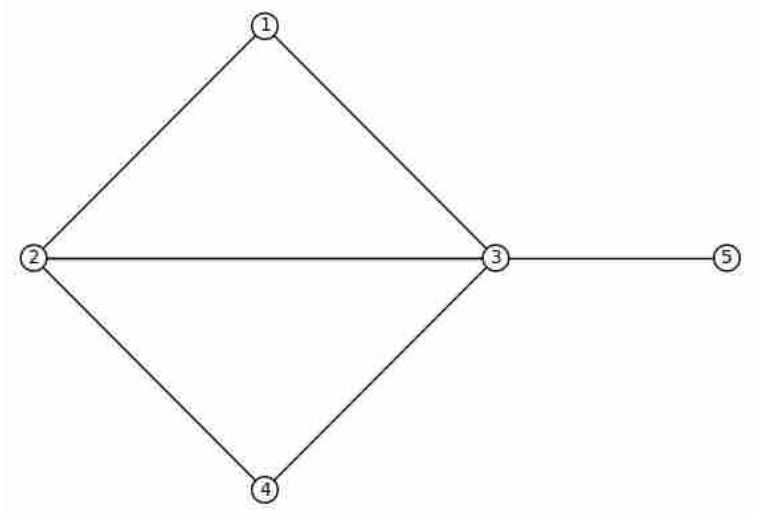


Figure 3.2: The clique {2,3} is a minimal (1,4)-separator but it is not a minimal clique cutset as {3} is a smaller cutset.

that any clique K of G is a prime subgraph of G. We start by observing the "location" of prime subgraphs in a decomposition tree.

Observation 3.1.3 *Let G be a graph and H be a prime subgraph of G. Consider $DS(G, C, G_1, G_2)$, if $H \not\subseteq C$ then $H \subseteq G_1$ or $H \subseteq G_2$.*

Proof. As H is not contained in C we can find a vertex $v \in V(G)$ such that $v \in V(H)$ and $v \notin V(C)$. Without loss of generality we assume that after the decomposition step $v \in V(G_1 - C)$. Let us consider two cases:

Case 1: H is a clique.

Since C is a clique cutset of G there exists no edge between v and any $u \in V(G_2 - C)$, so $H \cap (G_2 - C) = \emptyset$ and necessarily $H \subseteq G_1$.

Case 2: H is not a clique.

Assume there exists a vertex $u \in V(H)$ such that $u \in V(G_2 - C)$. Since any path in H with endpoints u, v must contain some vertex of C, we note $H \cap C$ is a non-empty clique. But that also means $H \cap C$ is a uv-separator and consequently H contains a clique cutset which is a contradiction (recall H is prime). So any $u \in V(H)$ must be in G_1 and we have $H \subseteq G_1$. □

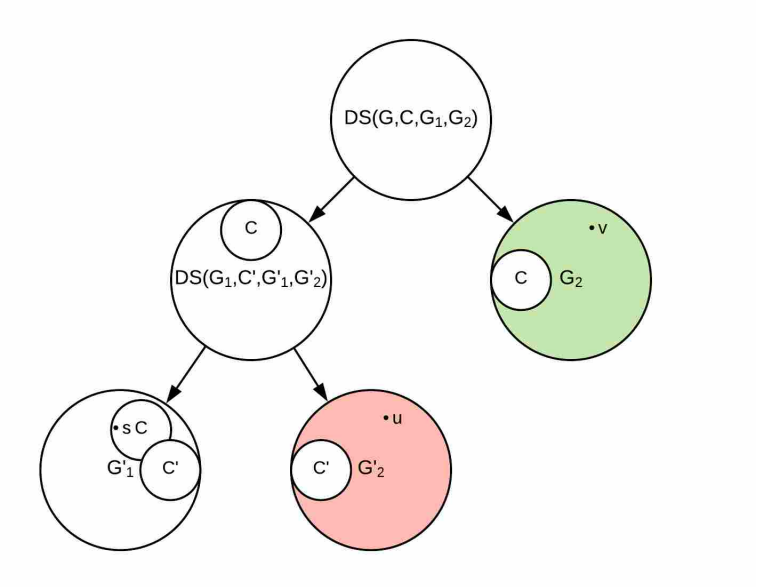


Figure 3.3: A clique cutset of a decomposition block is also a clique cutset of the parent graph.

As a general rule, clique cutsets have the property of "bubbling up" a decomposition tree (also mentioned in [51]). More precisely, let G' be the graph corresponding to some node N of $T(G)$ then any clique cutset of G' is a cutset of the graph corresponding to any ancestor of N (this is Observation 3.1.5 below). We start with the following observation.

Observation 3.1.4 *(known [51]) Let G be a decomposable graph, consider $DS(G, C, G_1, G_2)$. Let G' be one of the decomposition blocks (i.e. G_1 or G_2) with clique cutset C', then C' is also a clique cutset of G.*

Proof. Without loss of generality we assume $G' = G_1$. Now, note that $C \subset G_1$ and $C' \subset G_1$ (if $C' = G_1$ then C' is not a clique cutset of G'). For the case $C \subseteq C'$ it is clear that C' is a clique cutset of G. In particular, if $C \subsetneq C'$ then C' is a non-minimal clique cutset. So we only need to examine the case $C \not\subseteq C'$. Consider $DS(G_1, C', G'_1, G'_2)$ (Fig. 3.3 may serve as a visual aid). By observation 3.1.3, we assume w.l.o.g. that $C \subseteq G'_1$. We know $G'_2 - C' \subset G_1 - C$. Let $u \in V(G'_2 - C')$ and $v \in V(G_2 - C)$. Suppose there exists a shortest path P with endpoints u, v such that

P does not contain any vertex in C'. Let s be a vertex of P that belongs to $C - C'$. The subpath P' of P from u to s has $P' \subseteq G_1$ and since C' is a cutset of G_1 the path P' must contain a vertex in C'. However, since $P' \subset P$ we have $P \cap C' \neq \emptyset$, a contradiction. □

Observation 3.1.5 *Let G be a decomposable graph and N be a node of $T(G)$ with label $DS(G', C', G'_1, G'_2)$. Then C' is a clique cutset of all $G_i \in P(N)$.*

Proof. Let $G_k = G'$ in $P(N)$, then by Observation 3.1.4 C' is a clique cutset of G_{k-1}. Now by induction C' is a clique cut set of all $G_i \in P(0 \leq i \leq k)$. □

3.1.1 Cutsets Intersection

We denote by $\mathbb{C}(G)$ all the clique cutsets of a decomposable graph G. The following is a breakdown of the various possible intersections among members of $\mathbb{C}(G)$.

Observation 3.1.6 *Any two clique cutsets C_1 and C_2 of G with $C_1 \neq C_2$ may intersect in the following ways:*

1. $C_1 \cap C_2 = \emptyset$
2. $C_1 \cap C_2 \neq \emptyset$ *and none of C_1 and C_2 is contained in the other.*
3. $C_1 \subset C_2$
4. $C_2 \subset C_1$

Definition 3.1.7 *Let $\{C_1, C_2\} \subseteq \mathbb{C}(G)$. If the intersection of C_1 and C_2 is of type 1 or 2 in Observation 3.1.6, we call them* ***mutually-exclusive clique cutsets****.*

Let G' be a decomposition block of $DS(G, C)$, we are interested to know when will a cutset of G be in $\mathbb{C}(G')$ (we already know $\mathbb{C}(G') \subseteq \mathbb{C}(G)$). Clearly this is affected by the choice of C, which why we consider the intersections. One might hope

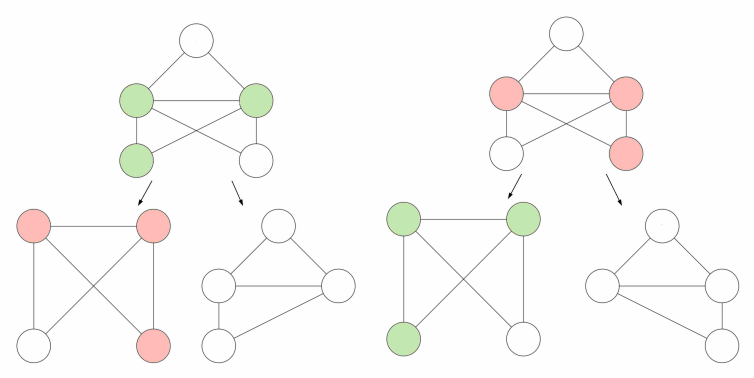


Figure 3.4: Two different decomposition steps using mutually-exclusive cutsets C_1 (rightside, colored green) and C_2 (on the left, colored red). C_2 is not a cutset for any of the decomposition blocks created by C_1 and vice versa.

that given two mutually-exclusive clique cutsets $\{C_1, C_2\} \subseteq \mathbb{C}(G)$, decomposing G by one of them will result in the other being a clique cutset of some decomposition block. Unfortunately, this is not so and Fig. 3.4 depicts a counterexample, but by imposing the condition in Proposition 3.1.8 below such behavior can be achieved.

Proposition 3.1.8 *Let C_1 and C_2 be two mutually-exclusive clique cutsets of G and $C' = C_1 \cap C_2$ is not a clique cutset. Consider $DS(G, C_1, G_1, G_2)$, then C_2 is a clique cutset of G_1 or G_2.*

Proof. By observation 3.1.3 we assume, w.l.o.g., $C_2 \subset G_1$. Since $C_1 \not\subset C_2$ there is a vertex $v_1 \in C_1 - C_2$. Consider $DS(G, C_2, G_1', G_2')$. Assume without loss of generality $C_1 \subseteq G_1'$. We may choose some vertex $v_2 \in V(G_2' - C_2)$ and note C_2 is a v_1v_2-separator (Fig. 3.5). Clearly, if $v_2 \in V(G_1)$ then C_2 is a clique cutset of G_1 and we are done. We only need to prove $v_2 \in V(G_1)$. Note, C' cannot be a clique cutset of G_2 since it is not a clique cutset of G (Observation 3.1.5). Suppose $v_2 \in V(G_2 - C_1)$ then we can find a path P with endpoints v_1, v_2 in $G_2 - C'$, otherwise (i.e. if all paths contain a vertex of C') C' is a cutset of G. Now, P is in $G_2 - C' \subset G - C_2$ (because $C_2 \subset G_1$), but this is a contradiction as removing C_2 should separate v_1, v_2. Hence, necessarily $v_2 \in V(G_1)$. $\square$

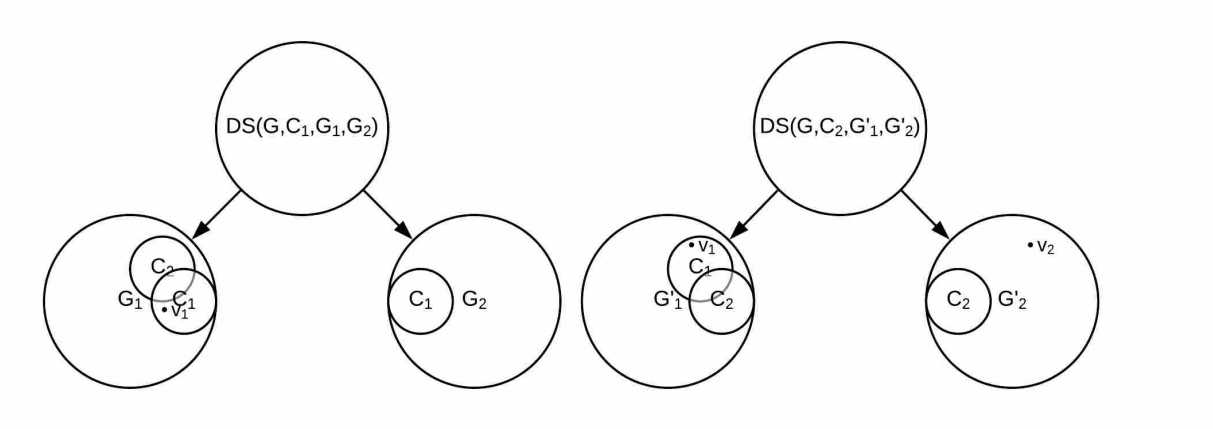


Figure 3.5: Illustration of Proposition 3.1.8. The intersection condition guarantees C_2 to be a clique cutset of G_1

The following lemma demonstrates every clique which lies "in between" clique cutsets is also a clique cutset.

Lemma 3.1.9 *Let C_1, C_2, C_3 be cliques of G such that $C_3 \subseteq C_2 \subseteq C_1$ and C_1, C_3 are cutsets of G. Then C_2 is a cutset of G.*

Proof. Let $H_1, ..., H_k$ be the components of $G - C_3$. If $C_2 = C_3$ or $C_2 = C_1$ then clearly C_2 is a cutset of G. So let us consider $C_3 \subset C_2 \subset C_1$. We know C_2 can intersect with at most one H_i because C_2 is a clique. Without loss of generality assume $C_2 \cap H_1 \neq \emptyset$. Suppose $H_1 - C_2 = \emptyset$, i.e. $H_1 \subset C_2$. Since $C_2 \subset C_1$ there exists a vertex $v \in H_j \cap C_1$ for some j. Now, there is no edge from v and $C_2 \cap H_1$, a contradiction (C_1 is a clique). So, $H_1 - C_2 \neq \emptyset$ and $G - C_2$ is disconnected. □

3.2 Atom Properties

In this section we proceed with our analysis by taking a closer look at the atoms. As usual, let G be a graph with clique cutset C and let G_1 and G_2 be the decomposition blocks of G using C. We denote the number of atoms created by a clique cutset decomposition $T(G)$ by $|atoms(T(G))|$ and denote the number of vertices in G by $|G|$. To start with, we recount the following essential equations.

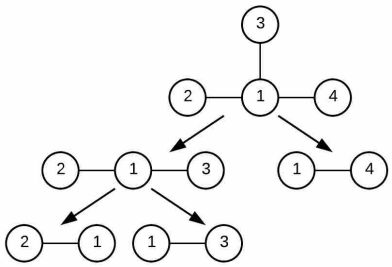


Figure 3.6: $\mathbb{L} = \{\{1\}, \{1\}\}$ and $\mathbb{C} = \{\{1\}\}$

- $|atoms(T(G))| = |atoms(T(G_1))| + |atoms(T(G_2))|$
- $|G| = |G_1| + |G_2| - |C|$
- $|C| < |G_1|$ and $|C| < |G_2|$
- $|G| - |C| \geq 2$

Here, $T(G_1)$ and $T(G_2)$ are the subtrees of $T(G)$ rooted at the second level. The decomposition tree $T(G)$ for any clique cutset decomposition is a full binary tree. It is well known that a full binary tree with I inner nodes has exactly $I + 1$ leaves. We use this property to derive the next two observations.

Observation 3.2.1 *Let G be a decomposable graph and $T(G)$ be some decomposition tree of G. We denote by $\mathbb{L}$ the multi-set of all clique cutsets used in $T(G)$ and by $\mathbb{C}$ the set of all clique cutsets used in $T(G)$ (see Fig. 3.6).*

- $|atoms(T(G))| = |\mathbb{L}| + 1$.
- $|atoms(T(G))| \geq |\mathbb{C}| + 1$.

Proof. Clearly, every clique cutset $C \in \mathbb{L}$ corresponds to an inner node of $T(G)$. As $T(G)$ is a full binary tree, it has $|\mathbb{L}| + 1$ leaves which represent $|atoms(T(G))|$. Every $C \in \mathbb{C}$ may appear in $\mathbb{L}$ more than once and therefore it gives a lower bound. □

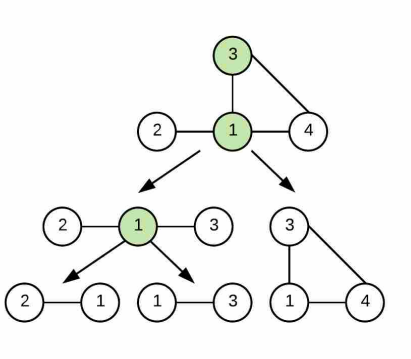


Figure 3.7: A decomposition which produces a non-mp-atom: the atom $\{1, 3\}$ is not a mp-subgraph as is belongs to larger prime subgraph $\{1, 3, 4\}$

3.2.1 On Non-maximal Atoms

Clique cutsets are obviously prime. It is therefore natural to wonder whether a clique cutset of G can possibly be an atom of some $T(G)$. The answer is yes as implied by Proposition 3.2.4 below. The next two observations demonstrate straightforward relations between atoms and clique cutsets.

Observation 3.2.2 *Let G be a decomposable graph and let A be an atom of some $T(G)$. Then there is a clique C such that $C \subset A$ and C is a clique cutset of G.*

Proof. We examine the decomposition path $P(A)$ of $T(G)$. G_k is the parent of A in $T(G)$ and let C_k be the clique cutset of G_k. Now $C_k \subset A$ and from Observation 3.1.5 C_k is also a clique cutset of $G_0 = G$. □

Observation 3.2.3 *(known [5]) Let G be a decomposable graph and C be a minimal clique cutset of G. For any $T(G)$ we have $C \notin atoms(T(G))$.*

Proof. Assume $C \in atoms(T(G))$. Then there exists $C' \subset C$ which is also a cutset of G (from Observation 3.2.2), but this contradicts the minimality of C. □

We recall that for an arbitrary clique cutset decomposition an atom $A \in atoms(T(G))$ does not need to be a mp-subgraph of G. An example is illustrated in Fig. 3.7. We will now show any such atom is a non-minimal clique cutset of G.

Proposition 3.2.4 *Let G be a decomposable graph and $A \in atoms(T(G))$. If $A \notin$ mp-subgraphs(G) then A is a non-minimal clique cutset of G.*

Proof. This proposition contains two claims, specifically, that A is a clique and A is a minimal-cutset. We start by proving A is a clique. Since A is not a mp-subgraph of G we know there must be a mp-subgraph A' such that $A \subset A'$. In addition, from Theorem 2.3.5, A' is an atom. Consider the lowest common ancestor N of A and A' along $P(A)$ and $P(A')$ (this node must exists as $A \neq A'$ and both paths originate at the root node $G = G_0$). Let the label of N be $DS(G', C', G'_1, G'_2)$. Without loss of generality let $A \subset G'_1$ and $A' \subset G'_2$. Because $A \subset A'$ we know $A \subseteq C'$ and therefore A is a clique. In Observation 3.2.2 we have established every atom contains a clique cutset of G, let us denote the clique cutset contained in A by C_A. So, we have $C_A \subset A \subseteq C'$, where both C_A and C' are cutsets of G. It follows from Lemma 3.1.9 that A is a clique cutset of G. Finally, A is not a minimal clique cutset since it contains C_A, a smaller cutset. □

It is possible to extract another piece of information regarding a non-mp-atom A. Not only is A a non-minimal clique cutset but apparently it is also *used* at some step of the decomposition.

Proposition 3.2.5 *Let G be a decomposable graph and $A \in atoms(T(G))$. If $A \notin$ mp-subgraphs(G) then A is a cutset of some $G' \in P(A)$.*

Proof. Let $\mathbb{C}_P$ be the set of all clique cutsets used along $P(A)$. Assume $A \notin \mathbb{C}_P$, meaning for every cutset $C \in \mathbb{C}_P$ we have $A \neq C$, or in other words either $A \not\subseteq C$ or $A \subset C$. Let H_i be a prime subgraph of G_i such that $A \subset H_i$ (such H exists for $G_0 = G$ by definition). We intend to show any of the two intersection options for A and C_i results in the existence of H_{i+1} in G_{i+1}. For $A \not\subseteq C_i$ we have $H_i \not\subseteq C_i$ and from Observation 3.1.3 we get $H_{i+1} = H_i \subseteq G_{i+1}$. The second case, $A \subset C_i$ implies we can define $H_{i+1} = C_i \subset G_{i+1}$. So far we have shown that $A \subset H_i \subset G_i$ for all i,

however this is a contradiction to $G_{k+1} = A$. Hence, there must be some $G' \in P(A)$ for which A is used as a cutset. $\square$

3.2.2 On the Number of Atoms

The famous Tarjan Decomposition was covered in Section 2.3, one of its important features is the fact it produces at most $n-1$ atoms. In fact it produces the minimum possible number of atoms as it only utilizes minimal clique separators. We will try to strengthen this bound (just the bound, surely we can not expect less atoms) by only allowing *minimum* clique cutsets. We denote by $T_m(G)$ a clique cutset decomposition of G where at each node only a minimum clique cutset may be chosen.

Theorem 3.2.6 *Let G be a decomposable graph and C be a minimum clique cutset of G, then $|atoms(T_m(G))| \leq |G| - |C|$.*

Proof. Let the decomposition at the root node be $DS(G, C, G_1, G_2)$. By strong induction we assume the theorem is correct for any induced proper subgraph of G. Let us consider the atoms of G_1 and G_2.

Case 1: G_1 and G_2 are both atoms. Here we find $atoms(G) = 2 \leq |G| - |C|$ and the theorem is correct.

Case 2: G_1 and G_2 are non-atoms. Let C_1 and C_2 be the minimum clique cutsets of G_1 and G_2 respectively. By the induction hypothesis: $|atoms(T_m(G_1))| \leq |G_1| - |C_1|$ and $|atoms(T_m(G_2))| \leq |G_2| - |C_2|$. We know $|atoms(T_m(G))| = |atoms(T_m(G_1))| + |atoms(T_m(G_2))| \leq |G_1| - |C_1| + |G_2| - |C_2|$. Now, C_1 and C_2 are also clique cutsets of G (Observation 3.1.4) but since C is a minimum clique cutset we have $|C_1| \geq |C|$ and $|C_2| \geq |C|$. From here: $|atoms(T_m(G))| \leq |G_1| + |G_2| - 2|C| \Rightarrow |atoms(T_m(G))| \leq |G| - |C|$.

Case 3: One of G_1, G_2 is an atom. Without loss of generality we assume G_1 is the atom and G_2 has a clique cutset C_2. So, $|atoms(T_m(G))| = 1 + |atoms(T_m(G_2))| \leq$

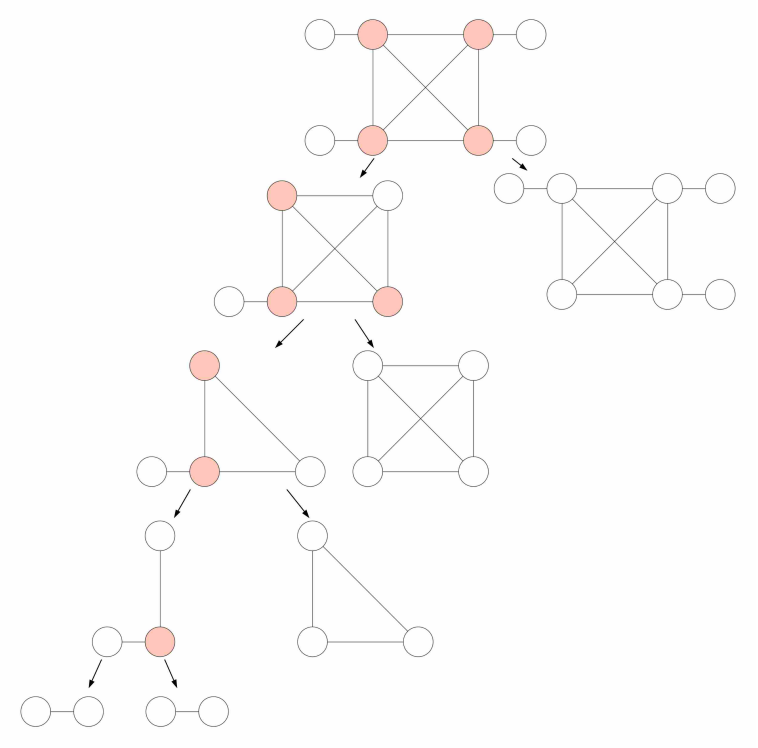


Figure 3.8: A clique cutset decomposition resulting in non-linear number of atoms.

$1 + |G_2| - |C_2| \leq 1 + |G_2| - |C|$. Obviously, $|G_2| < |G|$ and we get $|atoms(T_m(G))| \leq |G| - |C|$. □

While the minimum clique cutset decomposition (which we have just seen) and the minimal clique separator decomposition produce a linear number of atoms (w.r.t. $|G|$) this is not guaranteed for an arbitrary decomposition by clique cutsets. For instance, consider a graph G composed of a clique K_l and a set of l disjoint vertices S. Draw an edge between $u \in S$ and $v \in V(K_l)$ if v has no neighbor in S and u has no other neighbor in K_l. Now, K_l is a clique cutset of G. Take $u \in V(S)$ and decompose G into $G_1 = G[V(K_l) \cup \{u\}]$ and $G_2 = G[V - \{u\}]$. It is possible to decompose G_1 into l atoms by repeatedly choosing the maximal clique cutset (as illustrated in Fig. 3.8). Applying this process recursively to G_2 will result in a total of $l^2 = \frac{|G|^2}{4}$ atoms.

The quadratic order of atoms in our extreme example is not surprising. This limit was also reported in [51, 102]. Gavril proved this by noting every inner node of $T(G)$

corresponds to at least one non-edge of G and as G may have at most n^2 non-edges this figure serves to bound the number of decompositions along with the number of atoms (as per Observation 3.2.1). We will now demonstrate how this estimate can be strengthened with respect to the largest clique of G.

Theorem 3.2.7 *Let G a decomposable graph and let K be a maximum clique of G. Let $T(G)$ be any clique cutset decomposition of G. Then $|atoms(T(G))| \leq |K|(|G| - |K|)$*

Proof. By virtue of G being decomposable, we derive: $|G| \geq 3$, $|K| \geq 2$, $|G| - |K| \geq 1$ and $|K|(|G| - |K|) \geq 2$. Now, let C be any clique cutset of G. Consider $DS(G, C, G_1, G_2)$ and assume, without loss of generality, that $K \subseteq G_1$ (Observation 3.1.3). Let K' be the maximum clique of G_2, clearly $|C| \leq |K'| \leq |K|$. Once again we consider the cases for G_1 and G_2 and employ induction.

Case 1: G_1 and G_2 are atoms. Then $|atoms(T(G))| = 2 \leq |K|(|G| - |K|)$.

Case 2: G_1 and G_2 are both decomposable. By induction we say: $|atoms(T(G))| = |atoms(T(G_1))| + |atoms(T(G_2))| \leq |K|(|G_1| - |K|) + |K'|(|G_2| - |K'|)$. Using the size properties of K and K' we write: $|atoms(T(G))| \leq |K|(|G_1| + |G_2| - |K| - |K'|) \leq |K|(|G| + |C| - |K| - |K'|) \leq |K|(|G| - |K|) - |K|(|K'| - |C|) \leq |K|(|G| - |K|)$.

Case 3: G_1 is decomposable and G_2 is an atom. Here $|atoms(T(G))| = |atoms(T(G_1))| + 1 \leq |K|(|G_1| - |K|) + 1 \leq |K|(|G - 1| - |K|) + 1 \leq |K|(|G| - |K|) - |K| + 1 \leq |K|(|G| - |K|)$.

Case 4: G_1 is an atom and G_2 is decomposable. In this case $|atoms(T(G))| = 1 + |atoms(T(G_2))| \leq 1 + |K'|(|G_2| - |K'|)$. Since $K \subseteq G_1$ we know:

- $|G_1| \geq |K|$
- $|G_2| = |G| - |G_1| + |C| \leq |G| - |K| + |C|$
- $|C| - |K'| \leq 0$

We examine the cases for $|K|$ and $|K'|$:

Case 4.1: $|G_1| > |K|$. Obviously then $|G_1| \geq |K|+1 \Rightarrow |G_2| \leq |G|-(|K|+1)+|C|$. Plug this into the atoms equation, $|atoms(T(G))| \leq 1+|K'|(|G|-(|K|+1)+|C|-|K'|) \leq 1+|K|(|G|-|K|)-|K'|+|K'|(|C|-|K'|)$. Now, we may disregard the last factor (since it is non-positive) and as $|K'| \geq 1$ we get $|atoms(T(G))| \leq |K|(|G|-|K|)$.

Case 4.2: $|G_1| = |K|$ and $|K'| = |K|$. For this case $|C| < |K| \Rightarrow |C|-|K| > 0$. We find the required bound by simply rearranging our preliminary equation $|atoms((T(G))| \leq 1+|K'|(|G|-|K|+|C|-|K'|) \leq |K|(|G|-|K|)+1+|K|(|C|-|K|) \leq |K|(|G|-|K|)$.

Case 4.3: $|G_1| = |K|$ and $|K'| < |K|$. First we rewrite $|atoms(T(G))| \leq 1+|K'|(|G|-|K|+|C|-|K'|) \leq |K'|(|G|-|K|)+1+|K'|(|C|-|K'|)$.
Similar to the previous case $|K'|(|C|-|K'|) \leq 0$ so instead we state $|atoms(T(G))| \leq |K'|(|G|-|K|)+1$. Note, $|G|-|K| > 0$ and $|K'| \leq |K|-1$. These observations lead to $|atoms(T(G))| \leq (|K|-1)(|G|-|K|)+1 \leq |K|(|G|-|K|)+1-(|G|-|K|) \leq |K|(|G|-|K|)$.

□

It is only called for to treat the equation in Theorem 3.2.7 as a maximum problem for $|K|$. By doing so we reach the next corollary. It is also worth noting this result is not-incidentally in line with our example.

Corollary 3.2.8 *Let G a decomposable graph and K be the largest clique of G. The maximum number of atoms for any $T(G)$ is achieved when $|K| = \frac{|G|}{2}$ and is bounded by $\frac{|G|^2}{4}$.* □

3.3 Minimal clique cutset decomposition

A minimal clique cutset decomposition is a clique cutset decomposition in which we are only allowed to decompose using minimal clique cutsets. In Theorem 3.2.6 we

have obtained a result for a minimum clique cutset decomposition, this is clearly a special case of the minimal clique cutset decomposition. We now aim to expand and provide more results for this type of decomposition. Through this section all decompositions implied by $T(G)$ are minimal clique cutset decompositions.

Definition 3.3.1 *Let S be a minimal clique separator of G. The graph $G[V-S]$ contains at least two connected components H_1, H_2 such $N_G(H_1) = N_G(H_2) = S$. These components are called **full components** of S.*

3.3.1 Decomposition Properties

Observation 3.3.2 *Any two minimal clique cutsets C_1 and C_2 of G with $C_1 \neq C_2$ are mutually-exclusive clique cutsets. As they may only intersect in the following ways:*

1. $C_1 \cap C_2 = \emptyset$
2. $C_1 \cap C_2 \neq \emptyset$ *and none of C_1 and C_2 is contained in the other.*

An important property of the minimal clique cutset decomposition, which follows from the next lemma, is that all minimal clique cutsets of G are utilized in any decomposition $T(G)$. We stress this property holds regardless of the order in which we choose them.

Lemma 3.3.3 *Let C_1 and C_2 be two minimal clique cutsets of G. Consider $DS(G, C_1, G_1, G_2)$. Then C_2 will be a clique cutset of either G_1 or G_2.*

Proof. From the Observation 3.3.2 C_1 and C_2 are mutually-exclusive cutsets and from minimality $C' = C_1 \cap C_2$ is not a clique cutset of G. The proof is now immediate from Proposition 3.1.8. □

Theorem 3.3.4 *(known [6]) Let $\mathbb{C}$ be the set of all minimal clique cutsets of a decomposable graph G. Then for every $C \in \mathbb{C}$ there is an inner node N of $T(G)$ which corresponds to C.*

Proof. By induction on the size of G. Let the root node of T(G) be labeled $DS(G, C, G_1, G_2)$. Now from Lemma 3.3.3 any $C' \in \mathbb{C} - \{C\}$ will be either a mini mal clique cutset of G_1 or G_2. Assume without loss of generality that C' is a minimal clique cutset of G_1. Let $\mathbb{C}_1$ be the set of all minimal clique cutsets of G_1 (we know $C' \in \mathbb{C}_1$). Now by induction $T(G_1)$ will include some node N_1 in which the corresponding graph is decomposed by with C'. Trivially any node of $T(G_1)$ is a node of $T(G)$ and we have $N \in T(G)$ which corresponds to C'. □

A charectaristic similar to that of Lemma 3.3.3 is showcased by the minimal clique separator decomposition [6]. And since every minimal clique cutset is a minimal clique separator it is intuitive to assume the minimal clique cutset decomposition is a variation of the the minimal clique separator decomposition. This is in fact so. But at the same time we also know not every minimal clique separator is a minimal clique cutset, to verify the analogy of the decompositions we plan to show all minimal clique separators are used as minimal clique cutsets in the minimal clique cutset decomposition.

Lemma 3.3.5 *(known [5]) Let G be a decomposable graph and $\mathbb{C}$ be the set of all minimal clique cutsets of G. Let S be a minimal clique separator of G with $S \notin \mathbb{C}$. Consider $DS(G, C, G_1, G_2)$. Then S is also a minimal clique separator of G_1 or G_2.*

Proof. Obviously S may not be a subset of any $C \in \mathbb{C}$ (i.e. $S \not\subseteq C$) so we know that S is a subgraph of G_1 or G_2 (by Observation 3.1.3). Assume without loss of generality $S \subseteq G_1$. We will now demonstrate that S is also a minimal clique separator of G_1. Let $u_1, u_2 \in V$ such that S is a minimal u_1u_2-separator (so, $u_1 \notin S$, $u_2 \notin S$ and $\{u_1, u_2\} \not\subseteq C$).

Case 1: $u_1, u_2 \in V(G_1)$. Notice that $(G_1 - S) \subset (G - S)$ hence clearly S remains a minimal clique u_1u_2-separator in G_1.

Case 2: $u_1, u_2 \in V(G_2)$

Case 2.1: One of u_1, u_2 is in C.

Without loss of generality assume $u_1 \in C$ and $u_2 \in V(G_2 - C)$. Now, $u_1 \in C - S$ and u_2 in some full component H of C. We know every vertex of C has a neighbor in every full component (Definition 3.3.1) therefore we can find a path P with endpoints u_1, u_2 such that all interior vertices belong to H. Note that $G[H \cup \{u\}] \subseteq G_2 - S \subset G - S$ meaning we have a path from u_1 to u_2 in $G - S$, but this is a contradiction.

Case 2.2: $u_1, u_2 \in V(G_2 - C)$. We claim there is no path from u_1 to u_2 in $(G_2 - C)$ and denote by H_i be the components of $G_2 - C$ that contains u_i for $i = 1, 2$. Otherwise S is not a minimal clique u_1u_2-separator as $(G_2 - C) = (G - G_1) \subseteq (G - S)$. In other words C is a minimal u_1u_2-separator in G_2 (if C contains a smaller separator C' then C' is a clique cutset of G, a contradiction to the minimality of C). We also claim that $C \not\subset S$, for otherwise C is a smaller u_1u_2-separator (note, $C \neq S$ or S is a minimal clique cutset). Seeing that G_2 is connected there is a path with endpoints u_1, u_2 in G_2 (and any such path goes through C). We can now identify a vertex $v \in C - S$. Recall, every vertex of C has a neighbor in every full component. So, the following paths must exists: P_1 with endpoints u_1, v and all interior vertices belong to H_1 and P_2 with endpoints v, u_2 and all interior vertices belong to H_2. Note, both P_1, P_2 contain no vertices of S. Concatenating P_1 and P_2 results in a path with endpoints u_1, u_2 in $(G - S)$. Therefore, S is not a u_1, u_2-separator which is a contradiction.

We conclude that this case is not possible.

Case 3: $u_1 \in V(G_1)$ and $u_2 \in V(G_2)$. First, we remark that if $u_2 \in C$ we are in Case 1 and if $u_1 \in C$ we have Case 2. So, $u_1 \in V(G_1 - C)$ and $u_2 \in V(G_2 - C)$.

Now, note that C is a minimal clique u_1u_2-separator. From here we gather C and S

are mutually-exclusive clique cutsets (by definition $S \not\subseteq C$ and by the minimality of S we have $C \not\subseteq S$). Furthermore, the minimality of C means $C' = C \cap S$ cannot be a clique cutset of G, thus Proposition 3.1.8 assures us S is a clique cutset of G_1. To show that S is indeed a minimal clique separator in G_1 we would like to pin-point two vertices which S separates minimally. Let $v \in C - S$. Let us show S is a minimal u_1v-separator in G_1. Assume there is a path P with endpoints v, u_1 in $G_1 - S$ (i.e. P does not contain any vertex of S), then we have a path in $G - S$ with endpoints u_1, u_2 (simply join P with some path in G_2 which avoids S). This of course contradicts S being a minimal clique u_1u_2-separator and with that we know S is u_1v-separator in G_1. We are going to show S is a minimal separator. Assume it is not. Then there is some $S' \subset S$ such that v and u_1 are disconnected in $(G_1 - S')$. However, we know that there is a path P' from u_1 to u_2 in $G - S'$ (since $(G - S) \subset (G - S')$), which must contain some vertex c of $V(C)$. By taking the subset of P' from u_1 to c and adding the edge cv we get a path with endpoints u_1, v in $(G_1 - S')$. Hence, no S' can be a u_1v-separator in G_1 and S must be a minimal. □

Theorem 3.3.6 *A minimal clique cutset decomposition is a minimal clique separator decomposition.*

Proof. Let G be a decomposable graph and $T(G)$ be some minimal clique cutset decomposition of G. From Theorem 3.3.4 we know all the minimal clique cutsets of G will be used in $T(G)$. We intend to prove that every minimal clique separator of G will also be used in $T(G)$. Let S be a minimal clique separator of G which is not a minimal clique cutset. S is trivially prime and will be contained in some atom A. Consider the path $P(A) = (G = G_0, ..., G_k, A)$ in $T(G)$ (illustrated in Fig. 3.1), we know $S \subset G_i$ $(0 \leq i \leq k)$. Assume S is not a minimal clique cutset of G_i (we will arrive at a contradiction) then according to Lemma 3.3.5, S is a minimal clique separator of G_{i+1}. Hence $G_{i+1} \neq A$ as A contains no clique cutsets and equally no clique separators, a contradiction to Lemma 3.3.5. □

Corollary 3.3.7 *Every inner node N of $T(G)$ is decomposed by a minimal clique separator of G.* □

3.3.2 Atoms Properties

Recognizing the minimal clique cutset decomposition is a minimal clique separator decomposition indicates the atoms of such a decomposition are exactly the set of mp-subgraphs [5, 65]. However, it is possible to reach this conclusion without relying on this result, as we intend to do here.

Lemma 3.3.8 *Let A be an atom of $T(G)$. Then A is not a clique cutset for any inner node N of $T(G)$.*

Proof. Certainly, this lemma is correct when A is not a clique, so we assume it is. Let $P(A) = (G = G_0, ..., G_k, A)$ be the path to A in $T(G)$ (G_k is the parent of A and C_k is the minimal clique cutset used for G_k). We recall from Observation 3.1.3 that A is contained in all graphs along $P(A)$. If A is a clique cutset chosen for some inner node $N \in T(G)$ then $P(A)$ must go through N, or in other words we expect to find $G_i \in P(A)$ $(0 \leq i \leq k)$ for which $A = C_i$. Recall, $C_k \subset A$ and C_k is a clique cutset for all G_i $(0 \leq i \leq k)$ (Observation 3.1.5). So we have $A \not\subseteq C_i$ for all i or C_i is not a minimal cutset of G_i. □

The last result in combination with Theorem 3.3.6 also entails no atoms is a minimal clique separator.

Corollary 3.3.9 *(known [5]) No minimal clique separator S is an atom of $T(G)$.* □

It is now easy to be convinced all atoms of $T(G)$ are mp-subgraphs of G. Assume by contradiction that for some atom A we find $A \notin \textit{mp-subgraphs}(G)$, then by Proposition 3.2.5 A is a clique cutset of some $G' \in P(A)$. But according to Lemma 3.3.8 this cannot be, so A must be a mp-subgraph of G. This discussion can be summarized by the theorem below.

Theorem 3.3.10 *(known [65]) Let G be a decomposable graph and A be an atom of a clique minimal cutset decomposition $T(G)$. Then $A \in$ mp-subgraphs(G).* □

Corollary 3.3.11 *(known [65]) Let G be a decomposable graph. Then any minimal clique cutset decomposition $T(G)$ produces the same set of atoms.*

Proof. We know from Theorem 2.3.5 that *mp-subgraphs*$(G) \subseteq$ *atoms*$(T(G))$. Let $A \in$ *atoms*$(T(G))$ then according to Theorem 3.3.10, we have $A \in$ *mp-subgraphs*(G). No mp-subgraph can appear more than once in *atoms*$(T(G))$, due to Lemma 3.3.8 and Observation 3.1.3. So we have *atoms*$(T(G)) \subseteq$ *mp-subgraphs*(G), therefore *atoms*$(T(G))$ = *mp-subgraphs*(G). □

3.3.3 On the Number of Atoms of the Decomposition

With Corollary 3.3.11 we have established the uniqueness property of the minimal clique cutset decomposition. We no longer need to refer to a specific decomposition tree when discussing the atoms of a decomposable graph G. Hence, in this section, we simply use the notation $atoms_{min}(G)$ (we also reiterate $atoms_{min}(G) = mp - subgraphs(G)$). Quite similar to our attempt in Theorem 3.2.6, we now focus on uncovering a stronger bound for $|atoms_{min}(G)|$, that is better than $n - 1$. To help with this goal we are introducing the concept of dedicated vertices.

Definition 3.3.12 *Let A be an atom of G and v be a vertex in A. We say that v is a* ***dedicated vertex*** *if A is the only atom of G containing v. A set $D = \{v_1, v_2, ..., v_n\}$ is a* ***dedicated set*** *if $D \subset A$ and each $v_i \in D$ is a dedicated vertex. A clique K is a dedicated clique if $V(K)$ is a dedicated set.*

Observation 3.3.13 *A vertex $v \in V$ is dedicated if and only it is not included in any minimal clique separator of G.*

We plan to assert the existence of dedicated vertices in a minimal clique cutset decomposition. This property enables us to bind the number of atoms of specific subgraphs.

Lemma 3.3.14 *In every minimal clique cutset decomposition of a graph G there exists at least one dedicated vertex v (which is found in some atom A).*

Proof. If G has no clique cutsets all of $v \in V$ are trivially dedicated. For a decomposable graph G, we intend to construct a special minimal clique cutset decomposition $T(G)$ which will allow us to easily identify a dedicated vertex in some atom A. It has already been established that any decomposition $T'(G)$ will result in $A \in atoms(T'(G))$ (Corollary 3.3.11). As suggested by Observation 3.3.13 we are looking for a vertex v which is not contained in any minimal clique separator of G (analogously, v is not in any clique cutset used by $T(G)$). Denote $G_0 = G$ and let $DS(G = G_0, C_0, G'_0, G''_0)$ be the label for the root node of $T(G)$. As always, we recursively define $G_{i+1} = G'_i$. At each decomposition step $DS(G_i, C_i, G'_i, G''_i)$ we partition the vertices such that:

- C_i is not a clique cutset of G'_i
- $C_{i-1} \subset G''_i$

We say G'_i is the left child of G_i, and G''_i is the right child of G_i. This partitioning makes sure $C_i \neq C_{i-1}$ and is possible due to the properties of minimal clique cutsets (Observation 3.3.2). We denote by S_i the set of vertices used in all minimal clique cutsets up to step i, specifically $S_i = \bigcup_{j=0}^{i} C_j$. Our choices enforce $(G_{i+1} - C_i) \cap S_i = \emptyset$. Let $G_{k+1} = A$ be the left-most atom of this decomposition, then any vertex $v \in V(A - C_k)$ is a dedicated vertex. Such v has not been used in any C_i $(0 \leq i \leq k)$ and it necessarily exists. □

Lemma 3.3.15 *Let S be a dedicated set of G. Denote $G' = G - S$ then $|atoms_{min}(G)| \leq |atoms_{min}(G')| + 1$.*

Proof. First we address the case where G has no clique cutset. G' must have at least one atom and therefore $|atoms_{min}(G)| = 1 \leq 2$ is correct.
Let A be an atom of G such that $S \subset A$. We intend to show that $(atoms_{min}(G) - \{A\}) \subseteq atoms_{min}(G') \Rightarrow |atoms_{min}(G)| - 1 \leq |atoms_{min}(G')|$. Let $A' \in (atoms_{min}(G) - \{A\})$, from Theorem 3.3.10 we know $A' \in$ *mp-subgraphs*(G). By the definition of a dedicated set $S \cap A' = \emptyset$ and therefore $A' \subseteq G'$. We claim that A' is a mp-subgraph of G'. Assume it is not. Then there exists A'' such that $A' \subset A'' \in$ *mp-subgraphs*(G'), but now A'' is also prime graph of G and A' is not maximal, a contradiction. Thus we conclude $|atoms_{min}(G)| \leq |atoms_{min}(G')| + 1$. □

Let us first reprove the popular $|G| - 1$ bound using Lemma 3.3.15. Afterwards we will show $|atoms_{min}(G)| \leq |G| - |H| + 1$ where H is any prime subgraph of G (in particular H may be the largest prime subgraph).

Theorem 3.3.16 *(known [91]) Let G be a connected graph with $|G| \geq 2$. Then $|atoms_{min}(G)| \leq |G| - 1$.*

Proof. From Lemma 3.3.14 we know G has some dedicated vertex v. Let $G' = G - \{v\}$, G' is certainly connected as v is not a cutset (Observation 3.3.13). We prove the theorem by induction on the number of vertices using Lemma 3.3.15. It is trivial to verify the theorem for the base case $|G| = 2 \Rightarrow |atoms_{min}(G)| = 1$. Applying the induction hypothesis to G' we have $|atoms_{min}(G')| \leq |G'| - 1 \leq |G - 1| - 1$. Now, $|atoms_{min}(G)| \leq |atoms_{min}(G')| + 1 \leq |G| - 2 + 1 \leq |G| - 1$. □

Theorem 3.3.17 *Let H be a prime-subgraph of G. Then $|atoms_{min}(G)| \leq |G| - |H| + 1$.*

Proof. For any subgraph H of G, obviously $|H| \leq |G|$, therefore $|G| - |H| + 1 \geq 1$. If G is not decomposable then $|atoms_{min}(G)| = 1$. So we have $|atoms_{min}(G)| \leq$

$|G| - |H| + 1$.

We break the discussion into two parts based on whether H is a dedicated set or not.

Case 1: H is a dedicated set.

Let $G' = G - H$, we will show $2 \leq |G'|$. Let C be a minimal clique cutset of G. Notice $H \cap C = \emptyset$. Consider $DS(G, C, G_1, G_2)$, and assume without loss of generality $H \subset G_1$. Now $G_2 \subseteq G'$ and clearly $2 \leq |G_2| \leq |G'|$. Now, we can apply Theorem 3.3.16 to G' and get $|atoms_{min}(G')| \leq |G'| - 1 \leq |G| - |H| - 1$. Using Lemma 3.3.15 we find $|atoms_{min}(G)| \leq |atoms_{min}(G')| + 1 \leq |G| - |H| \leq |G| - |H| + 1$.

Case 2: H is a not dedicated set.

Observation 3.3.13 implies there is some minimal clique separator S for which $S \cap H \neq \emptyset$. Consider $DS(G, S, G_1, G_2)$ and partition the vertices so $H \subseteq G_1$. Notice, H is also a prime-subgraph of G_1 so by induction we say $|atoms_{min}(G_1)| \leq |G_1| - |H| + 1$. Clearly $S \subset G_2$ but from the Corollary 3.3.9 we know $S \notin atoms_{min}(G_2)$. Therefore there exists a prime subgraph M of G_2 which contains S ($S \subset M$). Using M and the induction hypothesis for G_2 we find $|atoms_{min}(G_2)| \leq |G_2| - |M| + 1 \leq |G_2| - |S|$. Combining the two tree branches gives us the desired outcome $|atoms_{min}(G)| = |atoms_{min}(G_1)| + |atoms_{min}(G_2)| \leq |G_1| - |H| + 1 + |G_2| - |S| \leq |G| - |H| + 1$. □

It is clear that for Theorem 3.3.17 the larger the prime subgraph H is, the tighter the bound. Unfortunately, we do not know of an efficient way to recognize the largest prime subgraph of an arbitrary G without actually performing a decomposition. However, since certain prime subgraphs are easily described (cliques, for instance) or are strictly not maximally prime (e.g. minimal cutsets) the theorem can be rephrased to reflect that.

Corollary 3.3.18 *Let G be a connected graph with $|G| \geq 2$ and K be any clique of G. Then $|atoms_{min}(G)| \leq |G| - |K| + 1$.* □

3.4 Maximal clique cutset decomposition

After exploring decompositions by minimal clique cutsets we shift our attention to the opposite extreme, namely maximal clique cutsets. We follow the same structure: first we examine the decomposition itself then we focus on properties of the atoms and their quantity.

3.4.1 Decomposition Properties

Observation 3.4.1 *Any two maximal clique cutsets C_1 and C_2 with $C_1 \neq C_2$ are mutually-exclusive clique cutsets.*

Lemma 3.4.2 *Let K be a clique of a decomposable graph G, and let C be a clique cutset of G, such that $C \subset K$. Then there exists a clique cutset $C' \subset K$ with $|C'| = |K - 1|$.*

Proof. Consider $DS(G, C, G_1, G_2)$. Assume without loss of generality $K \subseteq G_1$ and identify a vertex $u \in K - C$. We may choose any $v \in V(G_2 - C)$, and we have C is a uv-separator. Define $C' = K - \{u\}$, clearly C' is also a uv-separator and thus we have a clique cutset with size $|K - 1|$. □

3.4.2 Atoms Properties

Conveniently, the minimal clique cutset case guaranteed any such decomposition will produce the same exact set of atoms. We are not that lucky when it comes to the maximal clique cutset decomposition. Figure 3.9 contains an example for two maximal clique cutset decompositions of a graph G which result in different atom-sets.

Observation 3.4.3 *Not all maximal clique cutset decompositions produce the same number of atoms.*

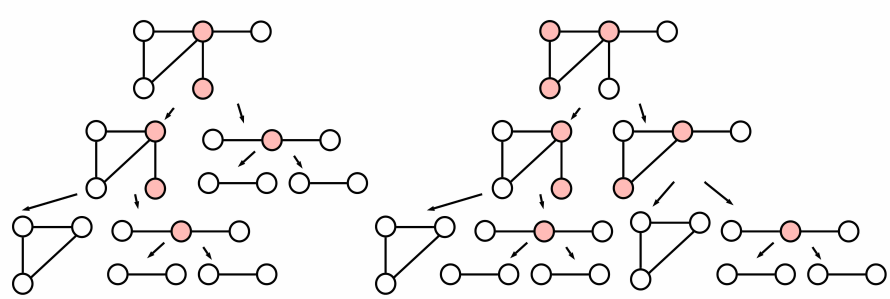


Figure 3.9: Two maximal clique cutset decompositions which result in different sets of atoms

In the following lemma we use *larger* in terms of the number of atoms the decomposition produces.

Lemma 3.4.4 *Let $T(G)$ be a decomposition tree of G with the root node labeled $DS(G, C, G_1, G_2)$. If C is not a maximal clique cutset, we can find a larger decomposition.*

Proof. Recall, $|atoms(T(G))| = |atoms(T(G_1))| + |atoms(T(G_2))|$. Choose C' a maximal clique cutset of G, with the property $C \subset C'$. Let us strategically construct a decomposition tree $T'(G)$ where the root is labeled $DS(G, C', G'_1, G'_2)$. We will show $|atoms(T'(G))| > |atoms(T(G))|$. See Fig. 3.10. Assume without loss of generality $C' \subset G_1$. We partition G'_1 and G'_2 such that $G'_1 = G_1$. Consequentially $G'_2 = G_2 \cup (C' - C)$. We know there exists $v \in C' - C$. Since $C' \subset G_1$ we know that C is a vu-separator for every $u \in V(G_2 - C)$. Therefore we may consider $DS(G'_2, C, G'_3, G'_4)$. By partitioning $V(G'_2)$ such that $G'_3 = C'$ we consequentially find $G'_4 = G_2$. We can now apply the previous decompositions of G_1, G_2, i.e. $T'(G'_1) = T(G_1)$ and $T'(G'_4) = T(G_2)$. Recall, G'_3 is a clique so $|atoms(T'(G'_2))| = |atoms(T'(G'_3))| + |atoms(T'(G'_4))| = 1 + |atoms(T(G_2))|$.
Overall we get $|atoms(T'(G))| = |atoms(T'(G'_1))| + |atoms(T'(G'_2))| = |atoms(T(G_1))| + |atoms(T(G_2))| + 1 = |atoms(T(G))| + 1$. □

Lemma 3.4.4 implies the following corollary.

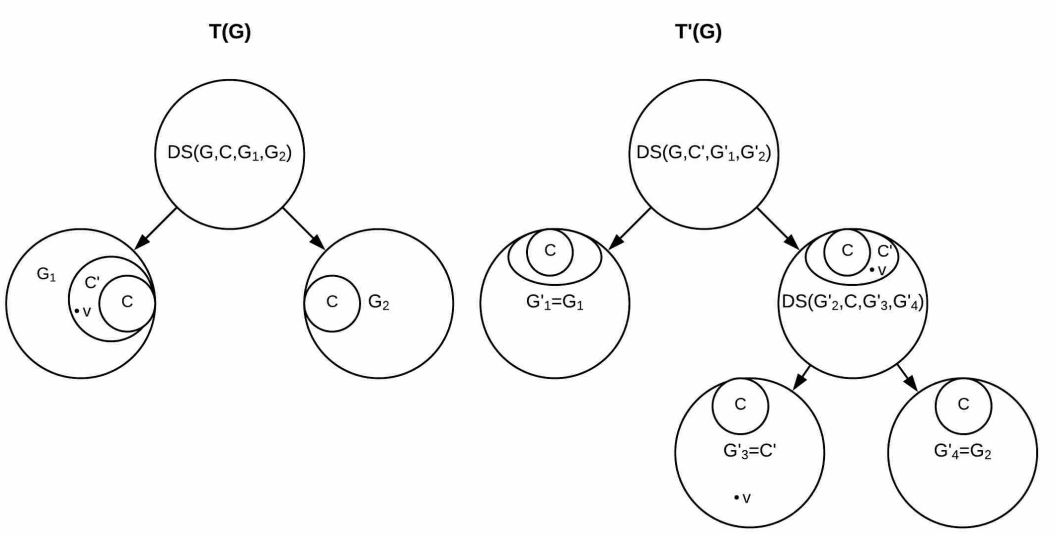


Figure 3.10: Depiction of how choosing a larger clique cutset (w.r.t. set-size) can produce more atoms. The decomposition on the right uses C' for which $C \subset C'$.

Corollary 3.4.5 *Let $T_{max}(G)$ be a decomposition of G which produces the largest possible number of atoms. Then every clique cutset used at every inner node of $T_{max}(G)$ is a maximal clique cutset.* □

3.4.3 On the Number of Atoms of the Decomposition

Corollary 3.2.8 gives the absolute "worst case" for the number of atoms which a clique cutset decomposition can produce. We have derived this result from Theorem 3.2.7 which relates $|atoms(T(G))|$ to the cliques of the corresponding graph G. Here we seek to uncover a similar relation with respect to *maximum* clique cutsets.

Observation 3.4.6 *A maximum clique cutset decomposition is a maximal clique cutset decomposition.*

Definition 3.4.7 *Let G be a decomposable graph and C be a clique cutset of G. The function $f(G, C)$ is the maximum number of connected components in $(G - C)$.*

It is plain to see that in general $f(G, C) \leq |G| - |C|$. The Star graph S_k is an example for $f(G, C) = |G| - |C| = k$. We denote by $T_M(G)$ a maximum clique cutset

decomposition of G.

Lemma 3.4.8 *Let G be a decomposable graph, C be a clique cutset of G and K be some clique of G such that $K \not\subseteq C$. Then $f(G,C) \leq |G| - |C \cup K| + 1$.*

Proof. Assume the graph $G - C$ has l connected components $H_1, ..., H_l$. Now, K is obviously connected and therefore it is completely contained in a some $C \cup H_i$ $(1 < i < l)$. Without loss of generality let us assume it is contained in $C \cup H_1$ (so $K \subseteq G[C \cup H_1]$). Trivially, H_1 is a single component so let us observe the subgraph $H = G[\{H_2, ..., H_l\}]$. We know H has $l - 1$ connected components, but it is also correct to say H has at most $|H|$ components. An even more detailed statement is $|H| = |G| - |C \cup H_1| \leq |G| - |C \cup K|$ (because $K \subseteq C \cup H_1$). Therefore $f(G,C) \leq 1 + |H| \leq 1 + |G| - |C \cup K|$. □

Theorem 3.4.9 *Let G be a decomposable graph and C be the maximum clique cutset of G. Then in a maximum clique cutset decomposition $T_M(G)$ we have $|atoms(T_M(G))| \leq |C|f(G,C)$.*

Proof. Consider $DS(G, C, G_1, G_2)$. For the case where G_1 and G_2 are both atoms the theorem holds since $|atoms(G)| = 2 \leq 2|C| \leq |C|f(G,C)$. Proving $|atoms(T_M(G_i))| \leq |C|(|G_i| - |C|)$ $(i = 1, 2)$ is sufficient as it leads to $|atoms(T_M(G))| = |atoms(T_M(G_1))| + |atoms(T_M(G_2))| \leq |C|(|G_1| - |C|) + |C|(|G_2| - |C|) \leq |C|(|G| - |C|)$. Let us therefore focus on G_1 and let C_1 be a maximum clique cutset of G_1, clearly $|C_1| \leq |C|$.

Case 1: $|C_1| = |C|$. By induction, it is immediate that $|atoms(T_M(G_1))| \leq |C|f(G_1, C) \leq |C|(|G_1| - |C|)$.

Case 2: $|C_1| < |C|$. We should consider whether or not C_1 is contained in C.

Case 2.1: $C_1 \subset C$. Lemma 3.4.2 implies $|C_1| = |C - 1|$, let u be the vertex left out of C_1 $(u \in C - C_1)$. Consider $DS(G_1, C_1, G', G'')$ and assume $C \subseteq G'$. We claim $G' = C$. Otherwise, C should have been the maximum clique cutset of G_1, as it would

be a separator for any $v_1 \in V(G' - C)$ and $v_2 \in V(G'' - C_1)$. Finally, using induction we find $|atoms(T_M(G_1))| \leq 1 + |atoms(T_M(G''))| \leq 1 + |C_1|f(G_1 - \{u\}, C_1) \leq 1 + |C - 1|(|G_1 - 1| - |C - 1|) \leq 1 + |C|(|G_1| - |C|)| - (|G_1| - C) \leq |C|(|G_1| - |C|)$.

Case 2.2: $C_1 \not\subset C$. In this case $f(G_1, C_1) \leq |G_1| - |C \cup C_1| + 1$ (Lemma 3.4.8). Now since $C_1 \not\subset C$ we know $|C \cup C_1| > |C|$, therefore $f(G_1, C_1) \leq |G_1| - |C|$. Employing the induction hypothesis $|atoms(T_M(G_1))| \leq |C_1|f(G_1, C_1) \leq |C|(|G_1| - |C|)$. □

Our last result for this chapter is another mean to bind the number of atoms of any clique cutset decomposition. We mention in certain cases it can serve as a better bound then the one presented in Theorem 3.2.7. In particular, let G be a decomposable graph, C be a maximum clique cutset of G and K be the maximum clique of G, if $C < K \leq \frac{|G|}{2}$ then Corollary 3.4.10 gives a tighter bound.

Corollary 3.4.10 *Let G be a decomposable graph and C be a maximum clique cutset. Then in a maximum clique cutset decomposition $|atoms(T_M(G))| \leq |C|(|G| - |C|)$.* □

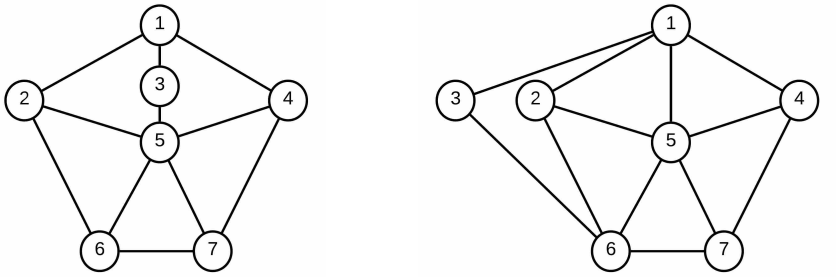
Chapter 4

SE-Class

We introduce a new graph family named *SE-Class*. Graphs in this class are defined as those that inherently contain simplicial extremes.

Definition 4.0.1 *A graph G is in **SE-Class** if any induced subgraph H of G contains a simplicial extreme. A **simplicial extreme order** (SEO) is an ordering σ such that v_i is a simplicial extreme in G_i.*

Figure 4.1a gives an example for a graph which belongs to SE-Class. The graph depicted in Fig. 4.1b does not belong to SE-Class. Note both graphs contain a simplicial extreme (vertex no. 3), however for the graph in Fig. 4.1b the induced subgraph on all other vertices has no simplicial extreme.



(a) A SE-Class graph (b) A non-SE-Class graph

Figure 4.1: SE-Class membership example

Clearly for any G in SE-Class we can construct a SEO, in this manner SE-Class generalizes chordal graphs (note that a PEO is a SEO). Additionally, we learn from Theorem 2.4.3 that a graph G of SE-Class which is also (Even-Hole)-free is β-perfect (the other direction is not true, take W_5 as an example).

Proposition 4.0.2 *Let G be a (Even-Hole)-free SE-Class graph. Then G is β-perfect.* □

But different from the perfect and β-perfect classes which forbid odd holes and even holes (respectively), graphs in SE-Class may contain any type of hole. Neither the odd nor the even hole is forbidden by the definition of SE-Class. In this chapter we explore the applications of a SEO as well as certain subfamilies of SE-Class. Our first result stems from the proof of Theorem 4.1.6 below (in Subsection 4.1.3) and stands in contrast to the known bound for (Even-Hole)-free graphs (Theorem 2.5.4). The following proposition is obvious:

Proposition 4.0.3 *Let G be a graph in SE-Class then $\chi(G) \leq max(\omega(G), 3)$.* □

This chapter is structured as follows: Section 4.1 explores SEO and its applications; Section 4.2 presents some structural results for related restricted graph families and in Section 4.3 we prove that a SEO for (Even-Hole, Claw, Diamond)-free graphs can be constructed using LexBFS in linear time.

4.1 SEO

In this section we focus on algorithmic aspects of SEO. We first demonstrate a naive SEO construction procedure followed by a linear time verification algorithm. Then we describe how a SEO of G can help us efficiently solve the coloring and maximum clique problems.

4.1.1 Generating SEO

To begin with, let us consider the complexity of finding a simplicial extreme vertex.

Lemma 4.1.1 *A simplicial extreme vertex can be found in $O(nm)$ time.*

Proof. A vertex v is trivially simplicial if it has a degree lower than 3. Otherwise, we need to examine whether its neighborhood induces a clique. This check can be performed using a single traversal of the graph in time $O(n+m)$. The worst case for finding a simplicial extreme requires testing all the vertices and therefore takes $O(nm)$ time. □

A straightforward approach, that consists of repeatedly finding and removing one simplicial extreme vertex, yields the next theorem.

Theorem 4.1.2 *SEO can be generated in $O(n^2m)$ time.* □

4.1.2 Verifing SEO

As per its definition, the vertices in a SEO may either be simplicial or have exactly two non-adjacent neighbors. The following observation gives a tool to differentiate between the two.

Observation 4.1.3 *Let σ be a SEO of G, then a vertex $v_i \in \sigma$ is simplicial if:*

- $deg_\sigma(v_i) = 1$
- $deg_\sigma(v_i) \geq 3$
- *there exists v_j with $j < i$, such that $deg_\sigma(v_j) = 3$ and v_i is the nearest larger neighbor of v_j in σ (i.e. in $\{v_j, ..., v_n\}$).*

Proof. Let σ be a SEO ordering of the vertices of G. The first two cases are trivial. A vertex v_i with $deg_\sigma(v_i) = 1$ is obviously simplicial. For v_i with $deg_\sigma(v_i) \geq 3$,

we know that v_i *must* be simplicial (or σ is not a SEO). Note that every v_i with $deg_\sigma(v_i) = 2$ is a simplicial extreme, so the task is to identify when such a vertex is required to be simplicial. Let us consider some v_i with $deg_\sigma(v_i) \geq 3$. Now, let $N_\sigma(v_i)$ be the neighborhood of v_i in G_i and let the three "smallest" (w.r.t. to the σ index) neighbors of v_i be v_k, v_l, v_m (with $k < l < m$). If $deg_\sigma(v_k) \geq 3$ then it is simplicial. Now, we can see that v_i forces v_k to be simplicial since v_k must be adjacent to all $N_\sigma(v_i) - \{v_k\}$. This justifies the third condition. □

Our next step is to meticulously design a linear SEO-verification algorithm. Meaning, given some ordering σ we would like to decide whether or not it is a SEO. For each $v_i \in \sigma$ $(1 \leq i \leq n-1)$ let $v_{i'}$ be the "smallest" neighbor of v_i in G_i. According to Observation 4.1.3, to test whether σ is a SEO, we need to verify the following for each v_i with $deg_\sigma(v_i) \geq 3$:

1. $v_{i'}$ is simplicial, and
2. $v_{i'}$ is adjacent to all $N_\sigma(v_i) - \{v_{i'}\}$

Algorithm 7 drafts an initial solution for this problem. The time to test whether $v_{i'}$ is simplicial is $O(n+m)$, hence the total complexity is governed by $O(nm)$ time.

Algorithm 7 Draft: Verify SEO

1: **for** $i \leftarrow 1$ to $n-1$ **do**
2: $v_{i'} \leftarrow$ smallest neighbor of v_i in G_i
3: task-1: test $v_{i'}$ is simplicial
4: task-2: test $v_{i'}w \in E$ for $w \in N_\sigma(v_i) - \{v_{i'}\}$
5: **end for**

To allow for a linear algorithm we employ the method developed in [85]. We scan σ from left to right (from smallest to largest) and exploit a similar strategy to that of perfect elimination order verification, namely deferring the check of simplicial vertices. We introduce two helper arrays: *bba* (stands for "better be adjacent") and *bbs* (short

for "better be simplicial"). The cell $bba[i]$ holds a set of vertices v_i is required to be adjacent to. The array bbs is a boolean array where $bbs[i] = true$ means v_i (with $deg_\sigma(v_i) \geq 2$) should be simplicial. Algorithm 8 demonstrates how both tasks can be accomplished when using an adjacency matrix.

Algorithm 8 Verify SEO (adjacency matrix)

Input: σ: an order of V
M: adjacency matrix of G
Output: $true$ if σ is SEO, $false$ otherwise

```
 1: for i ← 1 to n do
 2:    initialize bba[i] to empty set
 3:    initialize bbs[i] to false
 4: end for
 5: for i ← 1 to n − 1 do
 6:    if v_i is not adjacent to some vertex in bba[i] then
 7:       return false
 8:    end if
 9:    if deg_σ(v_i) ≥ 3 then
10:       v_i' ← smallest neighbor of v_i in G_i
11:       bbs[i] = bbs[i'] = true
12:       append N_σ(v_i) − {v_i'} to bba[i']
13:    else if deg_σ(v_i) = 2 and bbs[i] = true then
14:       {v_i', v_i''} = N_σ(v_i)
15:       if M[i'][i''] = 0 then
16:          return false
17:       end if
18:    end if
19: end for
20: return true
```

Task 1 for $v_{i'}$ is taken care of when the external loop reaches $v_{i'}$ (line 5). If $v_{i'}$ has $deg_\sigma(v_{i'}) \geq 3$ then the verification is implicit (it in fact relies on task 2). For $deg_\sigma(v_{i'}) = 2$ we perform the test immediately (that is at the time of processing $v_{i'}$) by looking at the adjacency matrix (line 15). This algorithm assumes the entries in the matrix correspond to the ordering σ but any other mapping method will work just as well. We pay no attention to vertices with $deg_\sigma(v_{i'}) = 1$ as they are trivially simplicial. Notice that we only append to $bba[i']$ when $deg_\sigma(v_i) \geq 3$, meaning all non-empty sets in bba contain at least two vertices (line 12). Task 2 is performed on

line 6 and requires $O(deg_\sigma(v_{i'}))$ for each $v_{i'}$. Overall, Algorithm 8 runs in linear time (i.e. $O(n+m)$) and requires $O(n^2)$ space due to the size of the adjacency matrix.

We can improve upon the space requirement of Algorithm 8 by avoiding the use of the adjacency matrix, as we do in Algorithm 9. One main difference is that now $bba[i]$ is allowed to contain a single vertex. However, we still make sure only simplicial vertices contribute to bba (line 13). While the time-complexity remains $O(n+m)$ the space-complexity can now be reduced to $O(m)$.

Algorithm 9 Verify SEO

Input: σ: an order of V

Output: $true$ if σ is SEO, $false$ otherwise

1: **for** $i \leftarrow 1$ to n **do**
2: initialize $bba[i]$ to empty set
3: initialize $bbs[i]$ to $false$
4: **end for**
5: **for** $i \leftarrow 1$ to $n-1$ **do**
6: **if** v_i is not adjacent to some vertex in $bba[i]$ **then**
7: return $false$
8: **end if**
9: $v_{i'} \leftarrow$ smallest neighbor of v_i in G_i
10: **if** $deg_\sigma(v_i) \geq 3$ **then**
11: $bbs[i] = bbs[i'] = true$
12: **end if**
13: **if** $bbs[i] = true$ **then**
14: append $N_\sigma(v_i) - \{v_{i'}\}$ to $bba[i']$
15: **end if**
16: **end for**
17: **return** $true$

For Algorithm 9 we are not using the adjacency matrix of G. However, we are still able to perform the adjacency check for v_i (line 6) in time $O(deg(v_i))$. Let us explain how this is achieved. Using the idea presented by Golumbic [52], we initialize a boolean array *test* of size n to $false$. For every v_i we iterate over $N(v_i)$ and set $test[j]$ to *true* for all $v_j \in N(v_i)$. This operation takes $O(deg(v_i))$ time. Now, there can be at most $deg(v_i)$ vertices in $bba[i]$ (or σ is not a SEO and we return). Checking whether $uv_i \in E$ for each $u \in bba[i]$ takes $O(1)$ using *test*. Finally, we reset *test* by changing all $deg(v_i)$ cells back to $false$. This procedure amounts to $O(deg(v_i))$ time for each $v_i \in V$. Overall, adjacency testing in Algorithm 9 requires $O(n+m)$ time.

The next theorem summarizes the discussion so far.

Theorem 4.1.4 *A SEO can be verified in $O(n+m)$ time.* □

4.1.3 Applications

Let G be a graph with a SEO σ. We will now demonstrate how can σ help identify the maximum clique of G.

Theorem 4.1.5 *Given a graph G and a SEO on G, a maximum clique can be found in $O(n+m)$ time.*

Proof. Clearly, whenever a vertex v_i is simplicial in G_i, it is part of a clique of size $deg_\sigma(v_i)+1$. On the other hand, if v_i is not simplicial, the largest clique containing it in G_i is an edge. So we define a *clique-size* function for $v_i \in V$ as:

$$cs(v_i) = \begin{cases} deg_{\sigma(v_i)} + 1, & \text{if } v_i \text{ is simplicial} \\ 2, & \text{otherwise} \end{cases}$$

The maximum clique of G is now given by $G[\{u\} \cup N_\sigma(u)]$ where $u \in V$ is the vertex with the largest clique-size, specifically $cs(u) = \max_{v_i \in V}(cs(v_i))$. The only

challenge left is to determine whether v_i is simplicial in G_i. We can easily accomplish that by constructing an indicator array using the following method. Initialize two arrays of size n: *simplicial* and *candidates*. Scan σ from left to right. If v_i has $deg_\sigma(v_i) \neq 2$ we mark it as simplicial and plainly move on. Otherwise, let $v_{i'}$ and $v_{i''}$ be the two neighbors of v_i in G_i. Without loss of generality assume $i' < i''$. Now, append the tuple $(i, v_{i''})$ to the list in $candidates[i']$. We think of this as a "request" of $v_{i'}$ to check whether v_i is simplicial. Note, every v_i can produce at most one "request", so $\sum_{i=1}^{n} |candidates[i]| \leq n$. We determine if v_i is simplicial when processing $v_{i'}$, if $v_{i''} \in N_\sigma(v_{i'})$ we set $simplicial[i]$ to $true$ otherwise v_i is not simplicial. We perform this adjacency check in a similar manner to that of Algorithm 9. □

Lastly, we show a greedy algorithm using a SEO produces an optimal coloring.

Theorem 4.1.6 *Given a graph G and a SEO on G, then G can be optimally colored in $O(n+m)$ time.*

Proof. Our first step is to check if G is bipartite. If it is we are done. Otherwise, we note that G requires at least 3 colors and we start to scan σ from right to left. We know v_i has $deg_\sigma(v_i)$ neighbors in G_i and respectively $N_\sigma[v_i]$ needs at most $deg_\sigma(v_i)+1$ unique colors (the available colors are $\{0, ..., deg_\sigma(v_i)\}$). For every vertex v_i we initialize all cells of a boolean array $free$ (of size $deg_\sigma(v_i)+1$) to $true$. We update $free[j]$ to $false$ if any $u \in N_\sigma(v_i)$ was assigned color j. If a neighbor of v_i has a color which is larger than $deg_\sigma(v_i)$ we disregard it. Finally, we choose the smallest free color and assign it to v_i.

Assume by contradiction such coloring is not optimal, this means at some vertex v_i we use a color l that is not necessary (meaning G_i could have been colored with $l-1$ colors). We should consider what type of vertex v_i is. Assume v_i is simplicial in G_i this implies $deg_\sigma(v_i) = l-1$, so G_i clearly needs at least l colors, a contradiction. Now we know v_i is not simplicial, and it may only be assigned one of the colors (0,1,2).

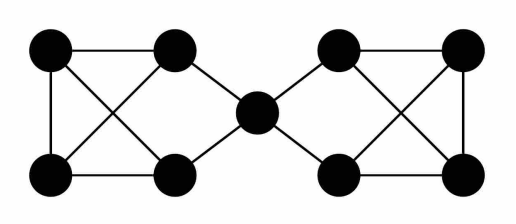


Figure 4.2: Example of a bad path

However, we have already established it is not possible to color G with less than 3 colors. So we could have not made a mistake and the coloring is optimal. □

4.2 Structural Results

We already know SE-Class contains (Even-Hole, Diamond)-free graphs, this is evident from Theorem 1.3.4. We'd like to learn which other families SE-Class contains. Another way to ask this question is: which graph families admit a SEO? Or, relatedly, when does a graph not admit a SEO?

The latter question motivates us to point out an obvious obstruction to SEO. Let G be a path $G_k = \{v_1, ..., v_k\}$ with $k \geq 5$, assume no vertex of G_k is simplicial and the degree of each vertex is at least 3 (i.e. $deg(v_i) \geq 3$). Then, clearly, we cannot eliminate any vertex of G_k. Thus G_k has no SEO and we call such structure a *bad path* (see Fig. 4.2). In addition, if $v_1v_k \in E(G_k)$ then there is a cycle of size k and we shall call it a *bad cycle*. The wheel W_k for instance is a bad cycle, it is also minimally non-SE-Class. Obviously, any graph which contains a bad path or a bad cycle cannot have a SEO. It is of course possible more obstructing structures exist.

4.2.1 (Even-Hole, Claw)-free

Cameron et al. [13] presented the *buoy* and the following structural result for (Even-Hole, Pan)-free graphs, these generalize the (Even-Hole, Claw)-free family.

Definition 4.2.1 *Let G be a connected graph. G has a **l-buoy** ($l \geq 5$) B with vertex-*

*sets $B_0, ..., B_{l-1}$ (called **bags**) such that $G[B_i]$ is a clique and every vertex $v \in B_i$ has a neighbor in B_{i-1} and B_{i+1}, and v has no neighbor in any other bag of B. A buoy of G is called **full buoy** if it includes all vertices of G.*

Theorem 4.2.2 *[13] If G is a connected graph and every atom of G is (Even-Hole, Pan)-free, then*

- *G is a clique, or*
- *G contains a clique cutset, or*
- *for every maximal buoy B of G, either B is a full buoy of G, or G is the join of B and a clique.*

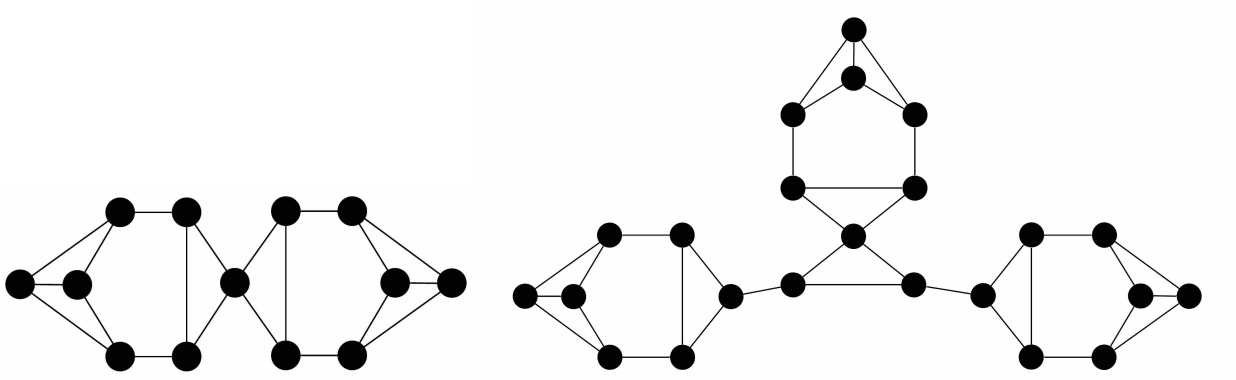
A *Hamiltonian cycle* (HC) in a graph G is a cycle of size n. We observe that for every buoy B on k vertices, a cycle C_k can be found. In particular, if B is a full buoy of G then G has a HC. Our proposition is build upon the latter Theorem.

Proposition 4.2.3 *Let G be a (Even-Hole, Claw)-free graph with no clique cutset. Then G is in SE-Class or G has a bad cycle.*

Proof. The claim trivially holds for any chordal graph so let us assume G has a hole, and moreover G contain a buoy. Let B be the maximal buoy of G.

Case 1: G is a join of B and a clique. We denote $G = B \oplus K$, where K is the clique. We know B has an induced hole F with length at least 5. Take any vertex $v \in V(K)$ and define $G' = G[V(F) \cup \{v\}]$. Every $u \in V(G')$ has $deg_{G'}(v) \geq 3$ and is not simplicial. Thus G' is a bad cycle and any G of this form is not in SE-Class.

Case 2: B is a full buoy of G. By definition buoys have no simplicial vertices and every vertex has a degree of at least 2. If for all $v \in V$ we have $deg(v) > 2$, then G is a bad cycle (notice, G has a HC). On the other hand, if $deg(v) = 2$ for some v then v is a simplicial extreme and we have a bag with size 1. The graph $G - \{v\}$ is chordal and therefore in SE-Class. Hence G is in SE-Class. □



(a) Not a bad cycle (b) Not a bad path

Figure 4.3: Minimally non-SE-Class (Even-Hole, Claw)-free graphs

Figure 4.3 depicts two minimally non-SE-Class (Even-Hole, Claw)-free graphs. Both examples have clique cutsets. We can see such graphs need not contain a bad cycle nor a bad path.

4.2.2 (Even-Hole, Kite)-free

Next, using Theorem 1.3.5 we prove that every (Even-Hole, Kite)-free graph is in SE-Class or has a wheel.

Proposition 4.2.4 *Let G be a (Even-Hole, Kite)-free graph that is not a clique and has no clique cutset, then G has a wheel or G has two non-adjacent simplicial extremes.*

Proof. This property is already known to us for a (Even-Hole, Diamond)-free graph G which is not a clique from Theorem 1.3.4. By Theorem 1.3.5, we're left to handle the case in which G is the join of a clique and a (Even-Hole, Diamond)-free graph. We denote $G = K \oplus H$, H is (Even-Hole, Diamond)-free and K is a clique. Note that H may not be a clique, otherwise G is a clique. Suppose H is chordal. Then H contains two non-adjacent simplicial vertices that remain simplicial in G. So, H contains a hole F. Now, take any vertex $u \in V(K)$ and observe $G[V(F) \cup \{u\}]$ is a wheel. □

Finally, we are ready to show the following:

Theorem 4.2.5 *Let G be (Even-Hole, Kite)-free then either*

- *G is a clique, or*
- *G has two non-adjacent simplicial extremes, or*
- *G has a wheel*

Proof. Proposition 4.2.4 handled the case in which G has no clique cutset, here we assume G has one. We also assume the theorem is true by induction. Let C be a clique cutset of G and G_1, G_2 be the decomposition blocks. We will show that G either contains a wheel or two non-adjacent simplicial extremes (one in $G_1 - C$ and the other in $G_2 - C$). We discuss G_1 but our logic is applicable to G_2 all the same. We apply the induction hypothesis to G_1.

Case 1: If G_1 is a clique then all vertices in $G_1 - C$ are simplicial.

Case 2: If G_1 has two non-adjacent simplicial extremes, then at least one of them must be in $G_1 - C$.

Case 3: If G_1 has a wheel then so does G. □

The last theorem also implies no minimal non-SE-Class (Even-Hole, Kite)-free graph can have a clique cutset. Simultaneously, we have proven every non-SE-Class (Even-Hole, Kite)-free graph has a bad cycle (specifically, a wheel). The wheel is in fact the only minimal obstruction for this class.

Corollary 4.2.6 *Let G be (Even-Hole, Kite)-free then G is in SE-Class if and only if G contains no wheel.* □

Detecting wheels is NP-complete [40] but fortunately we do not have to do that directly. The following theorem provides an efficient mean to recognize (Even-Hole,

Kite)-free graphs which belong to SE-Class using the minimal clique separators decomposition.

Theorem 4.2.7 *Let G be a (Even-Hole, Kite)-free graph. Then G is in SE-Class if and only if every $A \in atoms(G)$ is diamond free.*

Proof. Let us start by observing two things about the wheel: 1) it contains diamonds; 2) it has no clique cutset. Therefore, if a wheel W is present in G it will be included in some atom A of G (i.e $W \subseteq A$). Let us assume all atoms are diamond free, then no atom contains a wheel and accordingly G is wheel free. By Corollary 4.2.6, G is in SE-Class. We now analyze the opposite direction and assume G is in SE-Class. Clearly all atoms of G are (Even-Hole, Kite)-free and have no clique cutset. By Theorem 1.3.5 every atom A is either diamond free or is the join of a clique and a (Even-Hole, Diamond)-free graph, the first option is exactly what we want so let us take a closer look at the second one. Let us label the clique with K and the diamond free subgraph with H such that $A = K \oplus H$. If H contains a hole, then A has a wheel and thus G is not in SE-Class, a contradiction. Hence, H has no holes, or in other words H is chordal. From here it follows that A is chordal with no clique cutset (the universal vertices of K cannot by part of a hole), which is precisely a clique [84] and obviously diamond free. □

4.2.3 (Even-Hole, Claw, Diamond)-free

Following all our previous discussions we are well aware (Even-Hole, Claw, Diamond)-free graphs are a subclass of SE-Class. Our interest in this class stems from an attempt to efficiently construct a SEO, we will elaborate on this in Section 4.3.

For now, let us introduce the *spear*, a relaxed variation of the $3PC(\Delta, .)$ (Fig. 4.4). Similarly to the 3PC definition, a graph G has a spear$(x_1x_2x_3, y)$ if $\{x_1, x_2, x_3\}$ is a triangle and there are three paths P_1, P_2, P_3 from x_1, x_2, x_3 to y respectively.

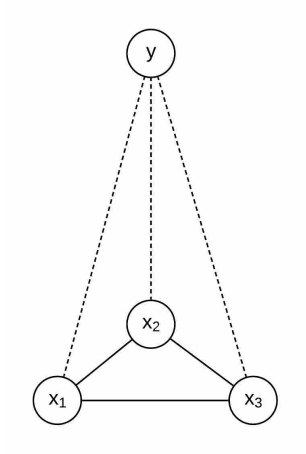


Figure 4.4: Spear

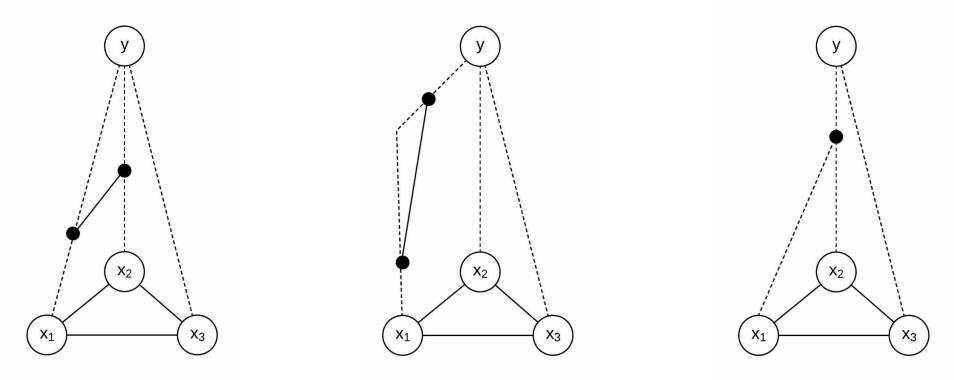
In a spear we do not require $G' = G[V(P_i) \cup V(P_j)]$ for $\{P_i, P_j\} \subset \{P_1, P_2, P_3\}$ to be a hole. Strictly speaking G' is a cycle in which chords are allowed or in other words P_i and P_j should both contain y but be otherwise *vertex-disjoint*. We emphasis another difference this definition makes. While the $3PC(\Delta, .)$ condition imposes that at most one of the paths may have length 1, this requirement does not apply to a spear (specifically, K_4 is a spear).

Definition 4.2.8 *A strong-spear is a spear for which the paths P_1, P_2, P_3 are all chordless.*

Definition 4.2.9 *A weak-spear is a spear for which the paths P_1, P_2, P_3 may not be chordless.*

Definition 4.2.10 *A near-spear is defined in a following manner. A graph G has a near-spear($x_1x_2x_3, y$) if there exist:*

- *$\{x_1, x_2, x_3\}$ is a triangle*
- *P_1 is a chordless path from x_1 to y and $x_2, x_3 \notin P_1$*
- *P_2 is a chordless path from x_2 to y and $x_1, x_3 \notin P_2$*
- *P_3 is a chordless path from x_3 to y and $x_1, x_2 \notin P_3$*



(a) strong spear (b) weak-spear (c) near-spear

Figure 4.5: Examples of different types of spears. Dashed lines represent paths and solid lines are edges.

Examples of the different types are illustrated in Fig. 4.5. Note that a near-spear allows paths $\{P_i, P_j\} \subset \{P_1, P_2, P_3\}$ to intersect and even partially merge (different from the strong-spear and weak-spear).

In the context of a spear(abc, d) or a near-spear(abc, d), where $\{a, b, c\}$ is a triangle and P_1, P_2, P_3 are the paths from a, b, c to d respectively, we use the following notations:

- A chordless path from some vertex $z \in \{a, b, c\}$ to d is denoted Z.
- The path $Z = \{z_0, ..., z_{z'}\}$ has z' vertices, such that $z_0 = z$ and $z_{z'} = d$.
- The notation $length(Z)$ is the number of edges of Z.
- We denote by $Z(i, j)$ the subpath of Z from vertex z_i to z_j.

We will now show both weak-spear and near-spear contain a strong-spear. We start by showing a weak-spear can always be transformed into a strong-spear.

Observation 4.2.11 *Let P be some path of G with endpoints x, y. Then P has an induced subpath P' such that P' is chordless and has endpoints x, y.*

Proof. By induction on the number of vertices. Let P be the path $\{v_0 = x, v_1, ..., v_k = y\}$. Assume P has a chord from v_i to v_j where $i < j$. Define $P' = P[\{v_0, ..., v_i\} \cup$

$\{v_j, ..., v_k\}]$. Now, P' is an induced subpath of P. By induction P' contains an induced chordless path P'' with endpoint x, y. P'' is also an induced subpath of P. □

Observation 4.2.12 *Every weak-spear contains an induced strong-spear.*

Proof. Let H be a weak-spear(abc, d) with paths P_1, P_2 and P_3. By Observation 4.2.11, P_1, P_2, P_3 can be replaced by chordless paths A, B, C respectively and we have a strong-spear(abc, d). □

Now, let G be a near-spear(abc, d), the *spear-size* is defined as the sum of the path-lengths $S(G) = length(A) + length(B) + length(C)$. A *minimal-size near-spear* G is a near-spear such that no induced subgraph G' of G is a near-spear(abc, d') with $S(G') < S(G)$ for some $d' \in V$. We intend to show the minimal-size near-spear does not contain any other type of spear (in particular, the strong-spear), as proper induced subgraph.

Lemma 4.2.13 *Let G be a minimal-size near-spear(abc, d). Then G is a strong-spear(abc, d).*

Proof. Assume G is not a strong-spear, then there exists an intersection between two of A, B, C. Without loss of generality we choose A to intersect with B. Assume $v \in A$ and also $v \in B$. Observe that unless $v = d$ we can identify the following three paths:

- $P_1 = A(0, v)$
- $P_2 = B(0, v)$
- $P_3 = A(v, d) \cup C$

The paths P_1, P_2 are clearly chordless and we may assume P_3 to be chordless as well (Observation 4.2.11). Note that $length(B(v, d)) \geq 1$ and with this we have a smaller-size near-spear(abc, v). But this is a contradiction to the minimality of G, hence G must be a strong-spear. □

Corollary 4.2.14 *Let G be a near-spear(abc, d). Then G contains a strong-spear(abc, v) for some $v \in G$.*

Proof. From Lemma 4.2.13 we know that if G is a minimal-size near-spear, it is a strong-spear. Assume G is not minimal-size near-spear then by definition there exists an induced subgraph G' of G which is a "smaller" near-spear. We apply the same argument recursively to G' until it is a minimal-size near-spear and therefore also a strong-spear. □

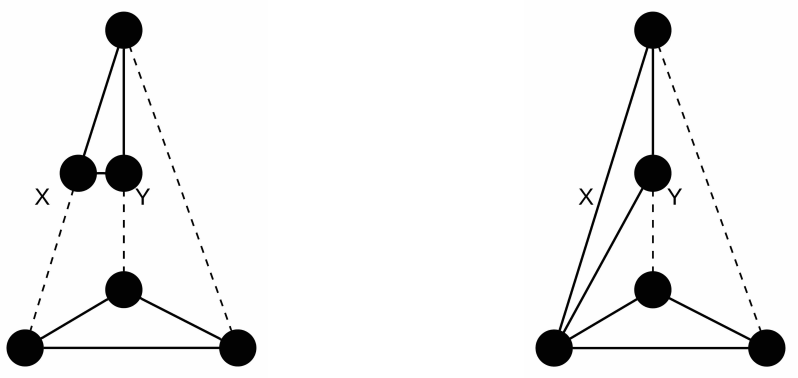
Let us recap our findings so far, from Observation 4.2.12 and Corollary 4.2.14 we know both weak-spear and near-spear contain a strong-spear. Clearly any strong-spear is also a weak-spear and a near-spear. Additionally, we have demonstrated a minimal-size near-spear is a strong-spear, so we conclude that a minimal-size spear is a strong-spear containing no other strong spear. We now simplify this construct and reformulate it as a *minimal-spear*. For the remainder of this chapter a spear implies a strong-spear and the term "smaller spear" is henceforward defined with respect to the number of vertices.

Definition 4.2.15 *A **minimal-spear** is a spear which does not contain another properly induced spear.*

We proceed by examining the structure of a minimal-spear. We show that a minimal-spear without an induced claw or diamond is either a K_4 or has special types of edges.

Lemma 4.2.16 *Let G be a minimal-spear(abc, d). Then either:*

- *G has an induced diamond or a claw*
- *G is a K_4*



(a) T_1 (b) T_2

Figure 4.6: Types of edges in a minimal-spear which is not a K_4 and is (Diamond, Claw)-free. Solid lines are edges, dashed lines are chordless paths

- *For any* $\{X, Y\} \subset \{A, B, C\}$*, if* $x_i y_j$ *is an edge such that* x_i *is an interior vertex of* X *or* y_j *is an interior vertex of* Y*, then* $i = x' - 1, j = y' - 1$ *(*T_1*) or* $i = 0, j = y' - 1$ *(*T_2*) (see Fig. 4.6).*

Proof. Our proof is by induction on the total number of vertices of the spear. We first consider the lengths of the paths A, B, C in terms of edges. We note that if all three paths have length 1 then G is precisely the K_4 on $\{a, b, c, d\}$. For the case where exactly two of the paths have length 1, it's easy to see a diamond is formed on $\{a, b, c, d\}$. We now focus on the case in which at most one path has length 1.
Let X and Y be the two distinct paths of the spear, $\{X, Y\} \subset \{A, B, C\}$. Let $H(X, Y)$ be the cycle formed by X, Y and the edge xy. If H is chordless for all possible X, Y, then d is the center of a claw $\{d, a_{a'-1}, b_{b'-1}, c_{c'-1}\}$ (such G is in fact a pyramid). Without loss of generality we may assume $H(A, B)$ has a chord $a_i b_j$ with $i > 0$ or $j > 0$, or both. We define the subpaths (Fig. 4.7):

- $S_1 = A(0, i)$, $S_2 = A(i, a')$
- $S_3 = B(0, j)$, $S_4 = B(j, b')$

Recall both A and B are chordless paths. This means no chord of H can have an end in d, i.e. $a_i \neq d$ and $b_j \neq d$. It follows that $length(S_2) \geq 1$ and $length(S_4) \geq 1$.

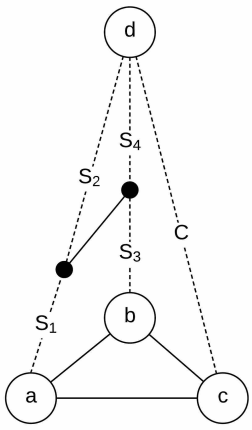


Figure 4.7: Illustration for Lemma 4.2.16

Since $i > 0$ or $j > 0$, we known $length(S_1) > 0$ or $length(S_2) > 0$.

Case 1: $length(S_1) \geq 1$ and $length(S_3) \geq 1$. We observe that for this case if $length(S_2) \geq 2$ then we have $G' = \text{weak-spear}(abc, b_j)$ with the paths: $P_1 = S_1 \cup a_i b_j, P_2 = S_3, P_3 = C \cup S_4$. We know G' contains a smaller spear (by Observation 4.2.12) which does not contain $a_{a'-1}$. But this is a contradiction to the minimality of G. Similarly, if $length(S_4) \geq 2$ then $G' = \text{weak-spear}(abc, a_i)$ with paths: $P_1 = S_1, P_2 = S_3 \cup a_i b_j, P_3 = C \cup S_2$ has a smaller spear. Therefore it must be $length(S_2) = length(S_4) = 1$. Consequently, we are left with exactly one possible edge $a_i b_j$ that is $i = a' - 1, j = b' - 1$. So this edge is of type T_1.

Case 2: $length(S_1) = 0$ or $length(S_3) = 0$. From symmetry we shall discuss $length(S_1) = 0$ (A and B may be swapped to fit this discussion). Our condition implies $a_i = a_0$. We first address the case where $length(S_3) = 1$ (i.e. $b_j = b_1$). We note there must be an edge between b_1 and c or $\{a, b, c, b_1\}$ is a diamond. But then $\{a, b, c, b_1\}$ is a K_4 which obviously is a smaller spear, a contradiction. So we may assume $length(S_3) \geq 2$. As in the previous discussion having $length(S_2) \geq 2$ leads to a smaller weak-spear(abc, b_j) with paths: $P_1 = a_0 b_j, P_2 = S_3, P_3 = C \cup S_4$ (accordingly, a smaller spear). Consequently, we may assume $length(S_2) = 1$, which implies $length(A) = 1$ and $length(B) \geq 2$. We have $length(S_4) = 1$, for otherwise there is a

claw $\{a, d, b_j, b\}$. Therefore the chord $a_i b_j$ is of type T_2. □

The following analysis suggests no (Even-Hole, Claw, Diamond)-free graph G may contain a minimal-spear which is not a K_4.

Lemma 4.2.17 *Let G be a minimal-spear(abc, d). Then either G is a K_4 or G contains as induced subgraph one of the following:*

- *even hole*
- *diamond*
- *claw*

Proof. Relying on Lemma 4.2.16, a minimal-spear which is not a K_4 and is (Diamond, Claw)-free should have an edge of type T_1 or T_2 for some $\{X, Y\} \subset \{A, B, C\}$. Without loss of generality we choose $X = A, Y = B$.

Case 1: A, B have an edge of type T_1 labeled $e_1 = a_{a'-1} b_{b'-1} \in E$.

Case 1.1: There is no edge between A and C (besides ac) and no edge between B and C (besides bc). Then G has a prism, specifically 3PC$(abc, da_{a'-1} b_{b'-1})$, and subsequently G has an even hole (Observation 2.5.1).

Case 1.2: There is an edge e_2 between A and C (besides ac) or B and C (besides bc). By symmetry we say the edge is between A and C. We should now consider what is the type of e_2.

Case 1.2.1: e_2 is of type T_1, meaning $e_2 = a_{a'-1} c_{c'-1} \in E$. We have $G' = \text{spear}(abc, a_{a'-1})$ with $P_1 = A(0, a'-1), P_2 = B(0, b'-1) \cup e_1, P_3 = C(0, c'-1) \cup e_2$. G' is a smaller spear than G since $d \notin V(G')$.

Case 1.2.2: e_2 is of type T_2 where $e_2 = a_0 c_{c'-1} \in E$.
Such e_2 requires $length(A) = 1$, but we know $length(A) \geq 2$ (because of e_1). So we cannot have this case.

Case 1.2.3: e_2 is of type T_2 where $e_2 = a_{a'-1}c_0 \in E$.

Here $G' = \text{spear}(abc, a_{a'-1})$ is a smaller spear with $P_1 = A(0, a'-1), P_2 = B(0, b'-1) \cup e_1, P_3 = e_2$ since we leave out d (i.e. $d \notin V(G')$).

To summarize, Case 1.1 results in an even hole and all three subcases of 1.2 contradict the minimality of G. We may now assume any edge between $\{X, Y\} \subset \{A, B, C\}$ (which is not xy) is of type T_2.

Case 2: A, B have an edge of type T_2 with $e_1 = a_0 b_{b'-1} \in E$.

Case 2.1: There is no edge between A and C (besides ac) and no edge between B and C (besides bc).

First we call attention to the fact that $length(B(0, b'-1))$ is odd, otherwise $B(0, b'-1) \cup e_1 \cup a_0 b_0$ is an even hole, subsequently $length(B)$ is even. Additionally, $length(C)$ is odd, otherwise $C \cup da_0 \cup a_0 c_0$ is an even hole. However now we have an even hole on $B \cup C \cup b_0 c_0$.

Case 2.2: There is an edge e_2 of type T_2 between A and C.

Since $length(A) = 1$ we can only have the edge $e_2 = a_0 c_{c'-1} \in E$. Now, we have $b_{b'-1}c_{c'-1} \in E$, otherwise there is a diamond on $\{d, a_0, b_{b'-1}, c_{c'-1}\}$. Now, there exists a smaller spear $G' = \text{spear}(abc, c_{c'-1})$ with paths: $P_1 = e_2, P_2 = B(0, b'-1) \cup b_{b'-1}c_{c'-1}, P_3 = C(0, c'-1)$ (note that $d \notin V(G')$). This contradicts the minimality of G.

Case 2.3: There is an edge e_2 of type T_2 between B and C.

The edge between B and C may only be of type T_1, as $length(A) = 1$ means B, C must have length greater than 1, a contradiction. □

Corollary 4.2.18 *Let G be a (Even-Hole, Claw, Diamond)-free minimal-spear(abc, d) then G is a K_4.* □

Up to this point we have established the structure of the minimal spear itself. We are now equipped with the tools to tackle graphs that contain spears.

Lemma 4.2.19 *Let $G' = spear(abc, d)$ be an induced subgraph of G. Then either G' has a K_4 containing $\{a, b, c\}$ or G contains as induced subgraph one of the following:*

- *even hole*
- *diamond*
- *claw*

Proof. We identify an induced minimal-spear within G'. Let $G'' = \text{minimal-spear}(abc, v)$ be an induced subgraph of G'. From Lemma 4.2.17 G'' is either the K_4 on $\{a, b, c, v\}$ or it has an induced even hole, diamond or claw. □

Our final and most general result regards graphs with near-spears. It will be of use to us in the next section.

Theorem 4.2.20 *Let G be a graph containing $G' =$near-spear(abc, d). Then either the graph induced on A, B, C has a K_4 which contains $\{a, b, c\}$ or G contains as induced subgraph one of the following:*

- *even hole*
- *diamond*
- *claw*

Proof. This follows effortlessly from Lemma 4.2.19 and Corollary 4.2.14. We know G' has an induced subgraph $G'' = \text{spear}(abc, v)$ and that G'' either contains as induced subgraph an even hole, a diamond or a claw or has a K_4 which contains $\{a, b, c\}$. □

4.3 SEO and LexBFS

In Section 4.1 we have displayed linear time applications of SEO towards coloring and maximum clique problems. The bottleneck, however, lies in generating such order.

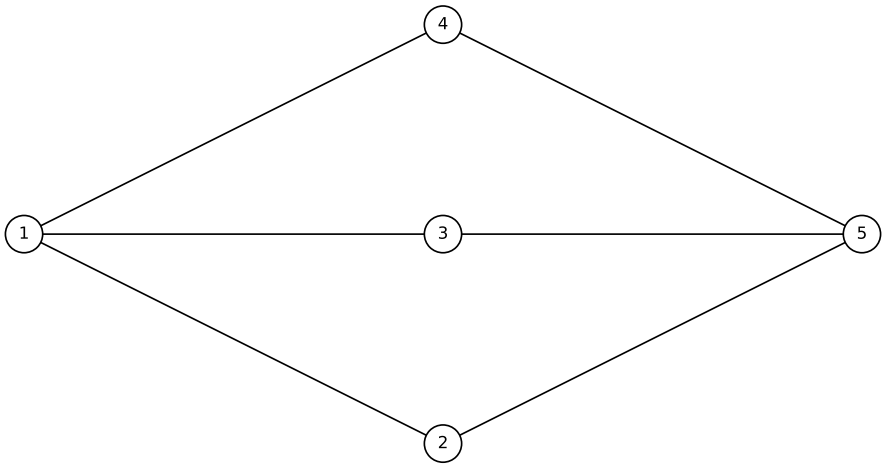


Figure 4.8: An example of a SE-Class graph for which a LexBFS ordering is not SEO. The vertex numbers represent the LexBFS ordering.

We have mentioned a naive method to generate a SEO in $O(n^2m)$ time, but a linear time construction would be much more profitable. Especially, it would be nice if some traversal algorithm could produce a SEO. In this section we are considering LexBFS as a candidate for this task, predominantly due to its ability to generate PEOs (as discussed in Section 2.2.2). Obviously, not every LexBFS ordering for any G in SE-Class is a SEO (an example of this is depicted in Fig. 4.8). What we are in fact seeking is a subclass of SE-Class for which every LexBFS produces a SEO. We aim to show this is so for the (Even-Hole, Claw, Diamond)-free family.

Let us introduce some new ordering notations first. Let $\sigma = \{\sigma_1, ..., \sigma_n\}$ be an ordering of the graph $G = (V, E)$. Let $v = \sigma_i$ ($v \in V$) for some $i \in \{1, ..., n\}$. We obtain the index of the vertex v in σ by $\sigma(v)$, expressly $\sigma(v) = i$. We name the start vertex of a LexBFS traversal s and accordingly $s = \sigma_n$. The following is a well known property of a LexBFS order.

Definition 4.3.1 *(known [42]) Let σ be a LexBFS ordering of G and $\{a, b, c\} \subseteq V$ with $\sigma(a) < \sigma(b) < \sigma(c)$. If $ac \in E$ and $bc \notin E$ then there exists $d \in V$ with $\sigma(c) < \sigma(d)$ which is adjacent to b but not to a.*

We start of by proving a property of the LexBFS order, a similar proof (under

the name *The Prior Path Lemma*) was given in [31]. We denote a path from some $v = \sigma_i$ to s with $P_\sigma(v)$.

Lemma 4.3.2 *(known [31]) Let σ be a LexBFS order of a connected graph G and $v = \sigma_i$. There exists a path $P_\sigma(v)$ from v to $s = \sigma_n$, which lies completely in G_i. Specifically, $P_\sigma(v)$ does not include any vertex $w = \sigma_j$ where $j < i$.*

Proof. By induction on the number of vertices. We may assume $vs \notin E$, for otherwise we are done. Let v' be the largest neighbor of v with $\sigma(v') > \sigma(v)$. Vertex v' exists by property of LexBFS (namely, we may only choose vertices which have been "seen" before). By induction, there is a path $P_\sigma(v')$. Now, $P_\sigma(v) = P_\sigma(v') \cup \{v, v'\}$ is the desired path. □

We now utilize the results obtained in Section 4.2.3 for (Even-Hole, Claw, Diamond)-free graphs.

Theorem 4.3.3 *Every LexBFS order on a (Even-Hole, Claw, Diamond)-free graph G is a SEO.*

Proof. Assume by contradiction there exists a vertex $\sigma_i = v$ which is not a simplicial extreme in G_i ($i \in \{1, ..., n\}$). Note that $deg_\sigma(v) \geq 3$ otherwise v is trivially a simplicial extreme. By requiring v to be non-simplicial we imply there exists at least one non-edge in $N_\sigma(v)$. Let u_1, u_2, u_3 be three neighbors of v in G_i with at least one non-edge among them. If the set $\{u_1, u_2, u_3\}$ does not contain an edge, then v is the center of a claw. If the set $\{u_1, u_2, u_3\}$ contains exactly one non-edge, then $\{v, u_1, u_2, u_3\}$ is a diamond. Hence we conclude that for v to satisfy our precondition there is, among u_1, u_2 and u_3, just one edge and exactly two non-edges.
We now choose u_3 to be the largest, with respect to the order σ, neighbor of v which has a non-edge with some $x \in N_\sigma(v)$. Observe, $\sigma(x) < \sigma(u_3)$. Consider another $y \in N_\sigma(v)$ (i.e. $y \neq x$ and $y \neq u_3$). Now $\{x, y, u_3\}$ is a set with at least one

non-edge. By the previous discussion the set $\{x, y, u_3\}$ has exactly one edge. So, y cannot be adjacent to both x and u_3. Additionally, $\sigma(y) < \sigma(u_3)$ or y would have been chosen as u_3 (the largest neighbor of v with a non-edge in $N_\sigma(v)$). In fact, this indicates u_3 is the largest neighbor of v. We now relabel x, y with u_1, u_2 such that $\sigma(u_1) < \sigma(u_2) < \sigma(u_3)$.

We examine all three adjacency cases. We intend to show, for all cases, a near-spear(abc, d) such that the graph induced on A, B, C has no K_4 which contains $\{a, b, c\}$.

Case 1: $u_2u_3 \in E$.

In this case we have $u_1u_3 \notin E$ and by the property of LexBFS there must be a u_4 with $\sigma(u_3) < \sigma(u_4)$ such that $u_1u_4 \in E$. Observe that $u_2u_4 \notin E$ otherwise there is a C_4 on $\{v, u_1, u_4, u_2\}$ (recall, $vu_4 \notin E$ and $u_1u_2 \notin E$). Now by LexBFS property there is a vertex u_5 with $u_2u_5 \in E$ and $\sigma(u_4) < \sigma(u_5)$. With this analysis we have a near-spear(vu_2u_3, s) in G_i:

- $P_1 = vu_1 \cup u_1u_4 \cup P_\sigma(u_4)$ $(u_2, u_3 \notin P_1)$
- $P_2 = u_2u_5 \cup P_\sigma(u_5)$ $(v, u_3 \notin P_2)$
- $P_3 = P_\sigma(u_3)$ $(v, u_2 \notin P_3)$

We know $\{v, u_1, u_2, u_3\}$ is not a K_4. Note, there in no induced K_4 containing $\{v, u_2, u_3\}$ on P_1, P_2, P_3 (recall, u_3 is the largest neighbor of v).

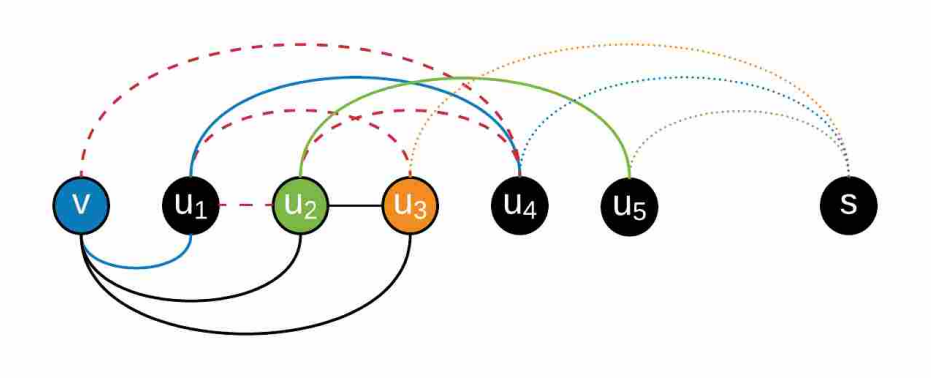


Figure 4.9: Solid lines are edges, red dashed lines are non-edges and blue,green and orange lines correspond to P_1, P_2, P_3.

Case 2: $u_1u_3 \in E$.

Here $u_2u_3 \notin E$ so there exists a u_4 with $\sigma(u_3) < \sigma(u_4)$ such that $u_2u_4 \in E$. Now the non-edge u_1u_2 implies there exists u_5 such that $\sigma(u_2) < \sigma(u_5)$ ($u_3 \neq u_5$ because $u_3v \in E$). If $\sigma(u_3) < \sigma(u_5)$, we define $P_2 = u_1u_5 \cup P_\sigma(u_5)$. Suppose $\sigma(u_5) < \sigma(u_3)$. We note $u_3u_5 \notin E$ (or $\{v, u_1, u_3, u_5\}$ is a diamond) and therefore there must be u_6 with $u_5u_6 \in E$ and $\sigma(u_3) < \sigma(u_6)$. Now, we define $P_2 = u_1u_5 \cup u_5u_6 \cup P_\sigma(u_6)$.

We can now identify the near-spear(vu_1u_3, s) with the paths:

- $P_1 = vu_2 \cup u_2u_4 \cup P_\sigma(u_4)$ $(u_1, u_3 \notin P_1)$
- P_2 defined as above. Note, $v, u_3 \notin P_2$.
- $P_3 = P_\sigma(u_3)$ $(v, u_1 \notin P_3)$

Here, $\{v, u_1, u_2, u_3\}$ and $\{vu_1u_3u_5\}$ are not a K_4. Also, no vertex on P_3 (besides u_3) is adjacent to v. So there in no induced K_4 containing $\{v, u_1, u_3\}$ on P_1, P_2, P_3.

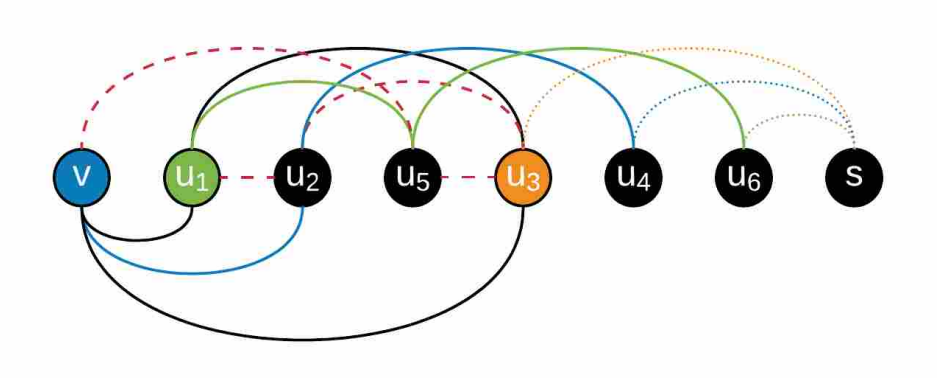


Figure 4.10: Solid lines are edges, red dashed lines are non-edges and blue,green and orange lines correspond to P_1, P_2, P_3. Note the positions of u_5 and u_6 are not strictly defined, these are placed only with respect to $\sigma(u_2) < \sigma(u_5)$ and $\sigma(u_3) < \sigma(u_6)$.

Case 3: $u_1u_2 \in E$.

As $u_1u_3 \notin E$ we have u_4 which is adjacent to u_1 and $\sigma(u_3) < \sigma(u_4)$. Now, u_4 may not be adjacent to u_2 or there is a diamond $\{v, u_1, u_2, u_4\}$. From here we have u_5, such that $\sigma(u_4) < \sigma(u_5)$ and $u_2u_5 \in E$. We have the following near-spear(vu_1u_2, s):

- $P_1 = vu_3 \cup P_\sigma(u_3)$ $(u_1, u_2 \notin P_1)$
- $P_2 = u_1u_4 \cup P_\sigma(u_4)$ $(v, u_2 \notin P_2)$
- $P_3 = u_2u_5 \cup P_\sigma(u_5)$ $(v, u_1 \notin P_3)$

Note, the sets: $\{v, u_1, u_2, u_3\}, \{v, u_1, u_2, u_4\}$ and $\{v, u_1, u_2, u_5\}$ are not K_4 (recall $vu_4 \notin E$ and $vu_5 \notin E$). So there is no induced K_4 on P_1, P_2, P_3.

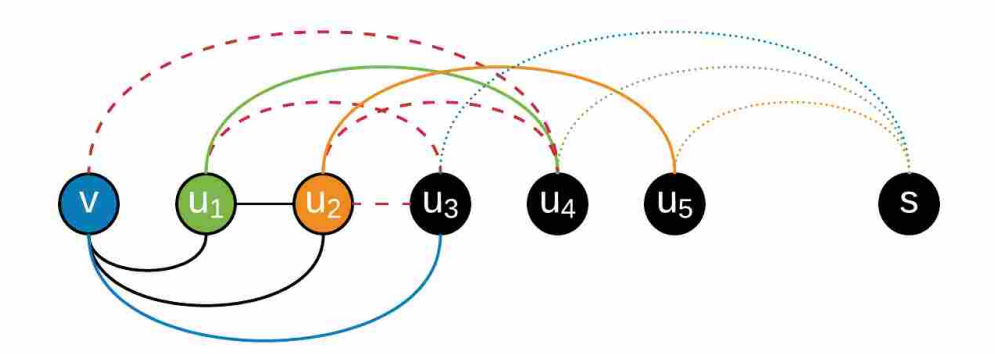


Figure 4.11: Solid lines are edges, red dashed lines are non-edges and blue,green and orange lines correspond to P_1, P_2, P_3.

For all cases, by Theorem 4.2.20, G has an induced even hole, diamond or claw. However, it is a contradiction to the definition of G. $\square$

Chapter 5

Case study: Coloring and certifying (Even-Hole, Kite)-free

This chapter ties together the two major topics of this thesis and outlines an application for the techniques we have presented. We say a graph G is k-critical if k colors are required to color G but any proper induced subgraph H of G can be colored with less than k colors. As the title of this chapter suggests our focus is (Even-Hole, Kite)-free graphs. In Section 5.1 we examine the structure of such k-critical graphs, then in Section 5.2 we devise a $O(n^3m)$ time certifying algorithm which extracts a k-critical induced subgraph from any (Even-Hole, Kite)-free graph.

5.1 The Structure of k-critical Graphs

Let us first establish some elementary properties regarding k-critical graphs. We should also note every antihole $\overline{C}_n$ with $n > 5$ contains an induced C_4, meaning graphs which are (Even-Hole, Odd-Hole)-free are perfect.

Observation 5.1.1 *Let G be a 3-critical (Even-Hole, Triangle)-free graph then G is an odd hole.*

Proof. We observe the absence of a triangle implies $\omega(G) \leq 2$. Now, if G does not contain any odd hole then G is perfect, but then G can clearly be colored by at most two colors (as $\omega(G) = \chi(G)$) which is a contradiction. Let H be an induced odd hole of G, assume there exists some $v \in V(G)$ such that $v \notin V(H)$ then $G[V - \{v\}]$ requires 3 colors as well and thus G is not 3-critical. □

Observation 5.1.2 *Let $G = (V, E)$ be any graph containing a clique H with k vertices then G is k-critical if and only if $G = H$.*

Proof. The "if" part is trivial, a clique H on k vertices is obviously k-critical. As for the other direction, assume there exists a vertex $v \in V(G)$ such that $v \notin V(H)$. We can easily see the graph $G[V - \{v\}]$ requires k colors which contradicts the criticality of G. □

This following lemma is related to Proposition 2.3.8.

Lemma 5.1.3 *A k-critical graph G does not contain a clique cutset.*

Proof. By contradiction. Let G be a k-critical graph which contains a clique cutset C and consider $DS(G, C, G_1, G_2)$. We recall that a coloring of G can be obtained from a coloring of G_1 and G_2 such that $\chi(G) = max(\chi(G_1), \chi(G_2)) = k$ (proof of Proposition 2.3.8). Without loss of generality we can assume $\chi(G_1) = k$. So, G_1 is a proper induced subgraph of G which requires k colors, a contradiction. □

SE-Class graphs

We now address k-critical SE-Class graphs, these are important as per the structural result in Theorem 1.3.5. Recall, (Even-Hole, Diamond)-free graphs are in SE-Class.

Observation 5.1.4 *Let G be a k-critical SE-Class graph with $k \geq 4$, then G is a clique with k vertices.*

Proof. We claim that in a k-critical graph each vertex $v \in V$ has $deg(v) \geq k-1$. Assume this is false, then we can remove v with $deg(v) < k-1$ and color the graph $G[V - \{v\}]$ with at most $k-1$ colors. Now since v has at most $k-2$ neighbors, we can surely assign it a color in $\{1, ..., k-1\}$, consequently making G $(k-1)$-colorable. By the definition of SE-Class, G has a simplicial extreme v. Furthermore, v must be a simplicial vertex as $deg(v) \geq k-1 \geq 3$. Thus $G[N[v]]$ is a clique with k vertices. From Observation 5.1.2, every k-critical graph that contains a clique with k vertices is in itself a clique with k vertices. □

Corollary 5.1.5 *Let G be a SE-Class graph with $\chi(G) = k \geq 4$ then G contains a clique with k vertices.* □

Combining Observation 5.1.4 and Observation 5.1.1 leads us to the next proposition.

Proposition 5.1.6 *A k-critical SE-Class graph G is either:*

- *a clique with k vertices, or*
- *$k = 3$ and G is an odd-hole* □

(Even-Hole, Kite)-free Graphs

Let us recall, a (Even-Hole, Kite)-free graph G is either (Even-Hole, Diamond)-free, the join of a clique and a (Even-Hole, Diamond)-free graph or G has a clique cut-set. We now demonstrate k-critical (Even-Hole, Kite)-free may only have one of two structures.

Theorem 5.1.7 *Let G be a k-critical (Even-Hole, Kite)-free graph with $k \geq 3$. Then G is either:*

- *a clique with k vertices, or*

- *a join of a clique of with $k-3$ vertices and an odd hole.*

Proof. Let G be a k-critical (Even-Hole, Kite)-free graph with $k \geq 3$. It is easy to see that a 3-critical graph is a K_3, or an odd hole. So we may assume $k \geq 4$. By Lemma 5.1.3, G does not contain a clique cutset. So by Theorem 1.3.5, G is (Even-Hole, Diamond)-free, or is the join of a clique K_t and a (Even-Hole, Diamond)-free graph G'. If G is (Even-Hole, Diamond)-free, then by Theorem 1.3.4, G is in SE-Class, and so by Corollary 5.1.5, G is a clique with k vertices, and we are done. So we know G is the join of a clique K_t and a (Even-Hole, Diamond)-free graph G'. We may assume G' is not a clique, for otherwise G is a clique and we are done. Obviously, the graph G' is $(k-t)$-critical, it also belongs to SE-Class. By Proposition 5.1.6, G' is an odd hole, and we are done. □

5.2 Certifying Algorithm for k-colorability

A certifying algorithm is one that produces a certificate with each output that proves it has not been compromised by an implementation bug. For example, a certifying algorithm for checking if an input graph G is bipartite produces either a 2-coloring, or an induced odd hole proving G is not bipartite. Of course, one would want the certificate to be simpler, i.e. easier to verify, than the algorithm itself. The notation of certifying algorithm provides new insights into the design and analysis of algorithms. Most known algorithms are not certifying. In the majority of these cases, the algorithms produce a certificate for Yes-instance but none for No-instances.

A certifying algorithm for the k-colorability problem, for an input graph G, outputs either a k-coloring of G, or an induced subgraph H that is not k-colorable; H could be for example a $(k+1)$-critical graph.

5.2.1 (Even-Hole)-free SE-Class Graphs

In this section, we show there is a $O(n^2m)$ time certifying algorithm for k-colorability of (Even-Hole)-free SE-Class graphs. This algorithm is described in Algorithm 10. Recall, from Proposition 4.0.2, that (Even-Hole)-free SE-Class graphs are β-perfect. Thus, they can be colored in linear time. On line 1 of Algorithm 10 we perform such coloring. If $\chi(G) \leq k$, the k-coloring is a Yes-certificate. The else clause of Algorithm 10, starting on line 4, deals with finding a No-certificate that is a $\chi(G)$-critical induced subgraph of G. Let H be the largest clique of G. If $\chi(G) \geq 4$, then by Corollary 5.1.5, we have $|H| = \chi(G)$, and we return H on line 8. If $\chi(G) \leq 3$, and $|H| < \chi(G)$, then we know G contains an odd hole by Proposition 5.1.6. We find the odd hole on line 10. This algorithm can be implemented in $O(n^2m)$ time, as suggested by the proposition 5.2.3 below.

Algorithm 10 Certifying k-colorability of (Even-Hole)-free SE-Class graphs

Input: G: an (Even-Hole)-free SE-Class graph
k: an integer
Output: (i) a k-coloring of G, or
(ii) a $\chi(G)$-critical induced subgraph for $\chi(G) > k$

```
1:  Compute χ(G) by coloring G
2:  if χ(G) ≤ k then
3:     return coloring
4:  else
5:     Generate SEO σ of G
6:     Find maximum clique H of G using σ
7:     if |H| = χ(G) then
8:        return H
9:     else
10:       Search for an odd hole H of G
11:       return H
12:    end if
13: end if
```

First, we are going to show line 10 can be implemented in linear time. In general, recognizing an odd hole is not an easy task, but in our scenario it is performed under

the assumption that $\chi(G) = 3$ and G contains no triangle (Proposition 5.1.6). We also know G has no even holes, so the problem of finding an odd hole is reduced to merely identifying a cycle, which is a fairly simple task. Let us prove that.

Observation 5.2.1 *Let G be (Even-Hole, Triangle)-free graph, then every cycle of G contains an induced odd hole.*

Proof. Let H be a cycle in G. Clearly, if H is chordless we have found our odd hole. So we assume H has some chord vu. Note that removing the edge vu does not disconnect H. Now, by performing a BFS on $G' = (V, E - \{vu\})$ starting from v we can obtain a path P with endpoints v, u. P is the shortest path from v to u and therefore chordless (in accordance with BFS properties). But now, $P \cup vu$ (alternatively, $G[V(P)]$) is an odd hole. □

It is well known that DFS can recognize a cycle in any graph G (if exists) in time $O(n+m)$. So a DFS combined with Observation 5.2.1 gives us an efficient algorithm to find an odd hole.

Proposition 5.2.2 *An odd hole can be identified in a (Even-Hole, Triangle)-free graph G in $O(n+m)$ time.* □

We are now ready to prove the following:

Proposition 5.2.3 *Let G be (Even-Hole)-free SE-Class graph with $\chi(G) = k$. Then a k-critical induced subgraph H of G can be found in $O(n^2m)$ time.*

Proof. We may assume G is connected. As per Theorem 4.1.2 generating a SEO takes $O(n^2m)$ time (line 5). Looking for the maximum clique takes $O(n+m)$ time by Theorem 4.1.5 (line 6). Finding an odd hole is linear-time from Proposition 5.2.2 (line 10). Hence, the overall complexity is $O(n^2m)$ time. □

5.2.2 (Even-Hole, Kite)-free Graphs

In this section we design Algorithm 11, a certifying algorithm for k-colorability of (Even-Hole, Kite)-free graphs. Let G be a (Even-Hole, Kite)-free graph. From Theorem 1.3.6 we can compute $\chi(G)$ in linear time. So, we are concerned only with finding a $\chi(G)$-critical subgraph. We distinguish among two cases: when G is in SE-Class; and when G is not in SE-Class.

Algorithm 11 Certifying k-colorability of (Even-Hole, Kite)-free graphs

Input: G: an (Even-Hole, Kite)-free graph
k: an integer
Output: (i) a k-coloring of G, or
(ii) a $\chi(G)$-critical induced subgraph for $\chi(G) > k$

1: Compute $\chi(G)$ by coloring G
2: **if** $\chi(G) \leq k$ **then**
3: return coloring
4: **else if** G in SE-Class **then**
5: return output of Algorithm 10(G, k)
6: **else**
7: Decompose G using Tarjan's decomposition
8: **for** $A \in atoms(G)$ **do**
9: Compute $\chi(A)$ by coloring A
10: **if** $\chi(A) = \chi(G)$ **then**
11: **if** A in SE-Class **then**
12: return output of Algorithm 10(A, k)
13: **else**
14: U = the set of universal vertices in A
15: $A' = A - U$
16: H' = output of Algorithm 10$(A', k - |U|)$
17: return $H = U \oplus H'$
18: **end if**
19: **end if**
20: **end for**
21: **end if**

Recognizing (Even-Hole, Kite)-free SE-Class Graphs

On line 4 of Algoritm 11 we check if an (Even-Hole, Kite)-free graph G is in SE-Class. Let us describe an $O(nm)$ time algorithm to test that. We use Tarjan's decomposition

method which produces at most $n-1$ atoms and requires $O(nm)$ time [91]. Obviously, each atom has at most n vertices and m edges, so we can safely use these as upper bounds.

Observation 5.2.4 *Let G be a SE-Class (Even-Hole, Kite)-free graph with no clique cutset. If G contains a universal vertex v then G is a clique.*

Proof. We know that if G is chordal, then it is a clique. Suppose G is not chordal and let H be a hole of G. Clearly $v \notin V(H)$. Now, $G[H \cup \{v\}]$ is a wheel, a contradiction to Corollary 4.2.6. □

Observation 5.2.5 *Let G be an (Even-Hole, Kite)-free graph. Then G is in SE-Class if and only if for every atom $A \in atoms(G)$:*

- *A has no universal vertex, or*
- *A is a clique*

Proof. We prove the "if" part. From Theorem 1.3.5, if A has no universal vertex, then A is diamond free. Also, recall cliques have no diamonds. So the "if" part follows from Theorem 4.2.7. We will prove the "only if" part by contradiction. Let G be an (Even-Hole, Kite)-free SE-Class graph. Note, all atoms of G are in SE-Class by Theorem 4.2.7. Suppose there exists $A \in atoms(G)$ such that A is not a clique and A has a universal vertex v, then it contradicts Observation 5.2.4. □

Lemma 5.2.6 *Let G be an (Even-Hole, Kite)-free graph, it is possible to check if G is in SE-Class in $O(nm)$ time.*

Proof. Let G be an (Even-Hole, Kite)-free graph. From Observation 5.2.5, in order to check that G is in SE-Class, we only need to make sure any atom $A \in atoms(G)$ which has a universal vertex is a clique. This can be performed in time $O(n + m)$,

per atom (by checking the degrees of the vertices of A). Since we have at most $n-1$ atoms, the overall time complexity is $O(nm)$ time. □

By Proposition 5.2.3, we can recognize an (Even-Hole, Kite)-free SE-Class graph in $O(n^2m)$ time. Note that on line 5 we invoke Algorithm 10 to perform this task. At this stage, we know Algorithm 10 will produce a $\chi(G)$-critical graph (because G is not k-colorable).

Now, let us consider a (Even-Hole, Kite)-free graph G which is not in SE-Class.

Identifing a $\chi(G)$-critical subgraph in Atoms

Let H be a $\chi(G)$-critical induced subgraph of G. From Lemma 5.1.3 we know H has no clique cutsets, meaning H is a prime subgraph of G and is therefore contained in some mp-subgraph of G. Our approach is to first decompose G using minimal clique separators (line 7) and then search for H in the atoms of the decomposition (recall, $\textit{mp-subgraphs}(G) = \textit{atoms}(G)$ for this decomposition by Theorem 2.3.6).

Obviously, every $A \in \textit{atoms}(G)$ is an (Even-Hole, Kite)-free graph. So, for each A, we can compute $\chi(A)$ in linear time (Theorem 1.3.6). Clearly, $\chi(A) \leq \chi(G)$. If $\chi(A) < \chi(G)$, then A does not contain a $\chi(G)$-critical induced subgraph. So suppose $\chi(A) = \chi(G)$. Let us explain how to find a $\chi(G)$-critical graph in A in time $O(n^2m)$.

Lemma 5.2.7 *Let G be an (Even-Hole, Kite)-free graph which is not in SE-Class. Let $\chi(G) = k$ and $A \in \textit{atoms}(G)$. If A contain a k-critical induced subgraph H, then H can be found in time $O(n^2m)$.*

Proof. Let $A \in \textit{atoms}(G)$ be an atom which contains a k-critical induced subgraph H. Suppose A is in SE-Class. Then from Proposition 5.2.3 we can find such H in $O(n^2m)$ time and we are done. So, we may assume A is not in SE-Class. From Theorem 1.3.5 we know $A = U \oplus A'$ (the join of a clique U and a diamond free graph A'), or A is in SE-Class. Note, A' is in SE-Class. Observe, $\chi(A) = k = |U| + \chi(A') \Rightarrow \chi(A') = k - |U|$.

Let H' be a $(k - |U|)$-critical induced subgraph of A'. We have $H = U \oplus H'$. We can obtain A' from A in linear time (by removing all universal vertices of A). From Proposition 5.2.3, we can find H' in A' in $O(n^2m)$ time. □

Note that on lines 12 and 16 we know Algorithm 10 outputs a $\chi(G)$-critical graph and a $(\chi(G) - |U|)$-critical graph, respectively (A is not k-colorable and A' is not $(k - |U|)$-colorable).

Certifying Algorithm: Complexity Analysis

We conclude the analysis of Algorithm 11 with the following proposition.

Proposition 5.2.8 *Let G be (Even-Hole, Kite)-free graph with $\chi(G) = k$. Then a k-critical induced subgraph H of G can be found in $O(n^3m)$ time.*

Proof. Let G be an (Even-Hole, Kite)-free graph. We may assume G is connected. If G is in SE-Class, we are done by Proposition 5.2.3. For G not in SE-Class, let us consider $atoms(G)$ produced by Tarjan's decomposition. Now, for each $A \in atoms(G)$ we check if it contains a $\chi(G)$-critical graph in linear time (lines 9-10). By Lemma 5.2.7, for an atom A with $\chi(A) = \chi(G)$ we can extract a $\chi(G)$-critical induced subgraph in time $O(n^2m)$ (lines 11-18). Finally, by looking through all the atoms we arrive at a time-complexity of $O(n^3m)$. □

Chapter 6

Conclusions and Open Problems

This final chapter presents problems left unsettled at the time of completing this thesis. These are questions arisen during the research process, some have extremely close proximity to discussions conducted herein while others seek to ambitiously generalize the attained results.

6.1 Clique Cutset Decomposition

We have provided a thorough analysis of the clique cutset decomposition, however we believe our results, specifically regarding cutsets interaction can be pushed further.

- Let C_1 and C_2 be mutually-exclusive clique cutsets of a decomposable graph G. Proposition 3.1.8 guarantees that if $C' = C_1 \cap C_2$ is not a clique cutset of G, then the atoms of a decomposition in which we consecutively choose C_1, C_2 are not altered by the order the cutsets are choose. We would like know whether a similar statement could be made for a pair of clique cutsets "farther down" a decomposition path. In addition, it would be nice to describe the opposite condition, i.e. under which circumstances two clique cutsets "cancel each other out"?

- As to bounding the number of atoms, Theorem 3.3.17 gives a strong bound but it remains to be seen what application would be able to take advantage of it, as it requires to identify a large prime subgraph of G. Is it possible to identify the largest prime subgraph in linear time?

- Theorem 3.4.9 gave an upper bound for the number of atoms of a maximum clique cutset decomposition with respect to $f(G, C)$ (as in Definition 3.4.7). Could $f(G, C)$ be also used to give a lower bound for such a decomposition?

- Corollary 3.4.10 bounded the number of atoms of any clique cutset decomposition with respect to any maximum clique cutset of G. We note that a maximal clique cutset decomposition is not unique. Yet, we are still curious whether a statement of the form: all maximal clique cutset decompositions are bounded by $|C|(|G|-|C|)$ where C is a maximal clique cutset of G, is correct. We should note, that if this is in fact true, it will only provide a better bound than the one of Corollary 3.4.10 when $|C'| < |C| < \frac{|G|}{2}$ where C is a maximum clique cutset and C' is some maximal clique cutset.

6.2 SE-Class

In Chapter 4 we presented SE-Class. We were able to point out specific subclasses of SE-Class but the large characterization problem remains open. One way to tackle this, which we have only briefly mentioned, might be a deeper investigation into the relationship between SE-Class and β-perfect graphs. We do know that any (Even-Hole)-free G which belongs to SE-Class is β-perfect, but what can be said about the opposite direction? In other words, which β-perfect graphs are in SE-Class?

We demonstrated how SEO can be used for coloring and finding a largest clique but a utilization towards the maximum stable set problem is yet unknown. We have also attempted to find a subclass of SE-Class for which any LexBFS ordering is a SEO.

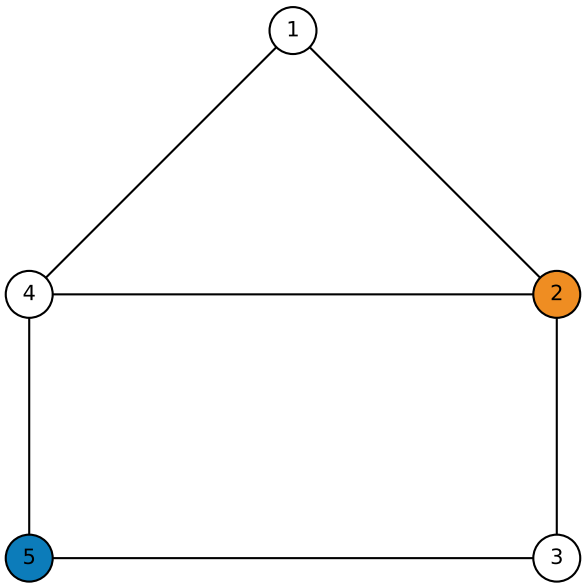


Figure 6.1: Three different paths with endpoints x (blue), y (orange) and length $|P_1| = |P_2| = 2, |P_3| = 3$ can be easily identified in a *house*. The original definition suggests an obstructive structure exists, however every LexBFS ordering for this graph is SEO (the vertex labels depict one such ordering).

We proved that (Even-Hole, Claw, Diamond)-free graphs have this property in Section 4.3 but this is not a complete characterization (take the even hole for example). From the study on those graphs we have learned there exists some structure obstructing this requirement, which can be roughly characterized as three paths P_1, P_2, P_3 with the same endpoints u, v and lengths that differ by at most one. It is easy to verify that if a LexBFS ordering is not SEO, then such structure exists. In fact the spears we have encountered during the proof of Theorem 4.3.3 are a special case of this structure. However, this general description is not strong enough to sustain an *if and only if* proof, for instance the *house* is a graph which contains such a structure but for whom all LexBFS orderings are SEO (Fig. 6.1). We therefore believe the obstruction would be better defined by a set of structures with more precise constraints over P_1, P_2, P_3. These are yet to be determined.

Bibliography

[1] L. Addario-Berry, M. Chudnovsky, F. Havet, B. Reed, and P. Seymour. "Bisimplicial vertices in even-hole-free graphs". In: *Journal of Combinatorial Theory, Series B* 98.6 (2008), pp. 1119–1164.

[2] C. Berge. *Hypergraphs: Combinatorics of Finite Sets*. North-Holland, 1989.

[3] C. Berge. "Motivations and history of some of my conjectures". In: *Discrete Mathematics* 165-166 (1997), pp. 61–70.

[4] C. Berge. *The Theory of Graphs*. Dover Publications, 2001.

[5] A. Berry, J.-P. Bordat, and O. Cogis. "Generating All the Minimal Separators of a Graph". In: *Graph-Theoretic Concepts in Computer Science*. Springer Berlin Heidelberg, 1999, pp. 167–172.

[6] A. Berry, R. Pogorelcnik, and G. Simonet. "An Introduction to Clique Minimal Separator Decomposition". In: *Algorithms* 3.2 (2010), pp. 197–215.

[7] A. Bjrklund, T. Husfeldt, and M. Koivisto. "Set Partitioning via Inclusion-Exclusion". In: *SIAM Journal on Computing* 39.2 (2009), pp. 546–563.

[8] V. Boncompagni, I. Penev, and K. Vušković. "Clique cutsets beyond chordal graphs". In: *Electronic Notes in Discrete Mathematics* 62 (2017), pp. 81–86.

[9] A. Brandstdt and C. T. Hoàng. "On clique separators, nearly chordal graphs, and the Maximum Weight Stable Set Problem". In: *Theoretical Computer Science* 389.1-2 (2007), pp. 295–306.

[10] A. Brandstdt, V. B. Le, and J. P. Spinrad. *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, 1999.

[11] A. Brandstdt, F. F. Dragan, V. B. Le, and T. Szymczak. "On stable cutsets in graphs". In: *Discrete Applied Mathematics* 105.1-3 (2000), pp. 39–50.

[12] A. Bretscher, D. Corneil, M. Habib, and C. Paul. "A Simple Linear Time LexBFS Cograph Recognition Algorithm". In: *SIAM Journal on Discrete Mathematics* 22.4 (2008), pp. 1277–1296.

[13] K. Cameron, S. Chaplick, and C. T. Hoàng. "On the structure of (pan, even hole)-free graphs". In: *Journal of Graph Theory* 87.1 (2017), pp. 108–129.

[14] H. C. Chang and H. I. Lu. "A faster algorithm to recognize even-hole-free graphs". In: *Journal of Combinatorial Theory, Series B* 113 (2015), pp. 141–161.

[15] P. Charbit, M. Habib, N. Trotignon, and K. Vušković. "Detecting 2-joins faster". In: *Journal of Discrete Algorithms* 17 (2012), pp. 60–66.

[16] M. Chudnovsky. "Berge trigraphs". In: *Journal of Graph Theory* 53.1 (2006), pp. 1–55.

[17] M. Chudnovsky and R. Kapadia. "Detecting a Theta or a Prism". In: *SIAM Journal on Discrete Mathematics* 22.3 (2008), pp. 1164–1186.

[18] M. Chudnovsky, G. Cornuéjols, X. Liu, P. Seymour, and K. Vušković. "Recognizing Berge Graphs". In: *Combinatorica* 25.2 (2005), pp. 143–186.

[19] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. "The strong perfect graph theorem". In: *Annals of Mathematics* 164.1 (2006), pp. 51–229.

[20] M. Chudnovsky, N. Trotignon, T. Trunck, and K. Vušković. "Coloring perfect graphs with no balanced skew-partitions". In: *Journal of Combinatorial Theory, Series B* 115 (2015), pp. 26–65.

[21] M. Chudnovsky, A. Lagoutte, P. Seymour, and S. Spirkl. "Colouring perfect graphs with bounded clique number". In: *Journal of Combinatorial Theory, Series B* 122 (2017), pp. 757–775.

[22] V. Chvátal. "Topics on Perfect Graphs". In: ed. by C. Berge and V. Chvátal. North-Holland Math, 1984. Chap. Perfectly ordered graphs, pp. 63–65.

[23] V. Chvátal. "Star-cutsets and perfect graphs". In: *Journal of Combinatorial Theory, Series B* 39.3 (1985), pp. 189–199.

[24] V. Chvátal and N. Sbihi. "Bull-free Berge graphs are perfect". In: *Graphs and Combinatorics* 3.1 (1987), pp. 127–139.

[25] V. Chvátal, C. T. Hoàng, N. V. Mahadev, and D. D. Werra. "Four classes of perfectly orderable graphs". In: *Journal of Graph Theory* 11.4 (1987), pp. 481–495.

[26] M. Conforti, G. Cornuéjols, and K. Vušković. "Decomposition of odd-hole-free graphs by double star cutsets and 2-joins". In: *Discrete Applied Mathematics* 141.1-3 (2004), pp. 41–91.

[27] M. Conforti, G. Cornuéjols, and K. Vušković. "Square-free perfect graphs". In: *Journal of Combinatorial Theory, Series B* 90.2 (2004), pp. 257–307.

[28] M. Conforti and M. R. Rao. "Testing balancedness and perfection of linear matrices". In: *Mathematical Programming* 61.1-3 (1993), pp. 1–18.

[29] M. Conforti, G. Cornuéjols, A. Kapoor, and K. Vušković. "Even and odd holes in cap-free graphs". In: *Journal of Graph Theory* 30.4 (1999), pp. 289–308.

[30] D. G. Corneil. "Lexicographic Breadth First Search – A Survey". In: *Graph-Theoretic Concepts in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 1–19.

[31] D. G. Corneil. *Lexicographic Breadth First Search A Survey*. Ed. by J. Hromkovič, M. Nagl, and B. Westfechtel. Vol. 3353. Springer Berlin Heidelberg, 2005.

[32] D. G. Corneil and J. Fonlupt. "Stable Set Bonding in Perfect Graphs and Parity Graphs". In: *Journal of Combinatorial Theory, Series B* 59.1 (1993), pp. 1–14.

[33] D. G. Corneil, E. Khler, and J.-M. Lanlignel. "On end-vertices of Lexicographic Breadth First Searches". In: *Discrete Applied Mathematics* 158.5 (2010), pp. 434–443.

[34] D. G. Corneil, S. Olariu, and L. Stewart. "The LBFS Structure and Recognition of Interval Graphs". In: *SIAM Journal on Discrete Mathematics* 23.4 (2010), pp. 1905–1953.

[35] G. Cornuéjols and W. H. Cunningham. "Compositions for perfect graphs". In: *Discrete Mathematics* 55.3 (1985), pp. 245–254.

[36] A. Cournier and M. Habib. "A new linear algorithm for Modular Decomposition". In: *Trees in Algebra and Programming 94*. Springer-Verlag, pp. 68–84.

[37] W. H. Cunningham. "Decomposition of Directed Graphs". In: *SIAM Journal on Algebraic Discrete Methods* 3.2 (1982), pp. 214–228.

[38] W. H. Cunningham and J. Edmonds. "A Combinatorial Decomposition Theory". In: *Canadian Journal of Mathematics* 32.03 (1980), pp. 734–765.

[39] R. Diestel. "Simplicial decompositions of graphs—Some uniqueness results". In: *Journal of Combinatorial Theory, Series B* 42.2 (1987), pp. 133–145.

[40] E. Diot, S. Tavenas, and N. Trotignon. "Detecting wheels". In: *Applicable Analysis and Discrete Mathematics* 8.1 (2014), pp. 111–122.

[41] G. A. Dirac. "On rigid circuit graphs". In: *Abhandlungen aus dem Mathematischen Seminar der Universitt Hamburg* 25.1-2 (1961), pp. 71–76.

[42] F. F. Dragan, F. Nicolai, and A. Brandstdt. "LexBFS-orderings and powers of graphs". In: *Graph-Theoretic Concepts in Computer Science*. Springer Berlin Heidelberg, 1997, pp. 166–180.

[43] J. D. Eblen, C. A. Phillips, G. L. Rogers, and M. A. Langston. "The maximum clique enumeration problem: algorithms, applications, and implementations". In: *BMC Bioinformatics* 13.Suppl 10 (2012), S5.

[44] A. Eisenblätter, M. Grötschel, and A. M. Koster. *Frequency Planning and Ramifications of Coloring*. eng. Tech. rep. 00-47. Takustr. 7, 14195 Berlin: ZIB, 2000.

[45] M. Farber. "On diameters and radii of bridged graphs". In: *Discrete Mathematics* 73.3 (1989), pp. 249–260.

[46] C. M. H. de Figueiredo and K. Vušković. "A class of β-perfect graphs". In: *Discrete Mathematics* 216.1-3 (2000), pp. 169–193.

[47] S. Fortunato. "Community detection in graphs". In: *Physics Reports* 486.3-5 (2010), pp. 75–174.

[48] D. J. Fraser, A. M. Hamel, and C. T. Hoàng. "On the Structure of (Even Hole, Kite)-Free Graphs". In: *Graphs and Combinatorics* 34.5 (2018), pp. 989–999.

[49] M. R. Garey, D. S. Johnson, and L. Stockmeyer. "Some simplified NP-complete graph problems". In: *Theoretical Computer Science* 1.3 (1976), pp. 237–267.

[50] F. Gavril. "Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques, and Maximum Independent Set of a Chordal Graph". In: *SIAM Journal on Computing* 1.2 (1972), pp. 180–187.

[51] F. Gavril. "Algorithms on clique separable graphs". In: *Discrete Mathematics* 19.2 (1977), pp. 159–165.

[52] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. Amsterdam, The Netherlands, The Netherlands: North-Holland Publishing Co., 2004.

[53] M. Grötschel, L. Lovász, and A. Schrijver. "The ellipsoid method and its consequences in combinatorial optimization". In: *Combinatorica* 1.2 (1981), pp. 169–197.

[54] M. Habib and C. Paul. "A survey of the algorithmic aspects of modular decomposition". In: *Computer Science Review* 4.1 (2010), pp. 41–59.

[55] M. Habib, R. McConnell, C. Paul, and L. Viennot. "Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing". In: *Theoretical Computer Science* 234.1-2 (2000), pp. 59–84.

[56] C. T. Hoàng. "Some properties of minimal imperfect graphs". In: *Discrete Mathematics* 160.1-3 (1996), pp. 165–175.

[57] B. Jamison and S. Olariu. "On the semi-perfect elimination". In: *Advances in Applied Mathematics* 9.3 (1988), pp. 364–376.

[58] T. Jian. "An $O(2^{0.304n})$ Algorithm for Solving Maximum Independent Set Problem". In: *IEEE Transactions on Computers* C-35.9 (1986), pp. 847–851.

[59] R. M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations*. Springer US, 1972, pp. 85–103.

[60] J. Keijsper and M. Tewes. "Conditions for β-perfectness". In: *Discussiones Mathematicae Graph Theory* 22.1 (2002), p. 123.

[61] W. S. Kennedy and B. Reed. "Fast Skew Partition Recognition". In: *Computational Geometry and Graph Theory*. Springer Berlin Heidelberg, 2008, pp. 101–107.

[62] T. Kloks and D. Kratsch. "Listing all Minimal Separators of a Graph". In: *SIAM Journal on Computing* 27.3 (1998), pp. 605–613.

[63] T. Kloks, H. Mller, and K. Vušković. "Even-hole-free graphs that do not contain diamonds: A structure theorem and its consequences". In: *Journal of Combinatorial Theory, Series B* 99.5 (2009), pp. 733–800.

[64] D. Král', J. Kratochvíl, Z. Tuza, and G. J. Woeginger. "Complexity of Coloring Graphs without Forbidden Induced Subgraphs". In: *Graph-Theoretic Concepts in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 254–262.

[65] H. G. Leimer. "Optimal decomposition by clique separators". In: *Discrete Mathematics* 113.1-3 (1993), pp. 99–123.

[66] C. Lekkeikerker and J. Boland. "Representation of a finite graph by a set of intervals on the real line". eng. In: *Fundamenta Mathematicae* 51.1 (1962), pp. 45–64.

[67] L. Lovász. "Normal hypergraphs and the perfect graph conjecture". In: *Discrete Mathematics* 2.3 (1972), pp. 253–267.

[68] L. Lovász. "On the Shannon capacity of a graph". In: *IEEE Transactions on Information Theory* 25.1 (1979), pp. 1–7.

[69] S. Markossian, G. Gasparian, and B. Reed. "β-Perfect Graphs". In: *Journal of Combinatorial Theory, Series B* 67.1 (1996), pp. 1–11.

[70] D. Marx. "Graph Coloring problems and their applications in scheduling". In: *Periodica Polytechnica Electrical Engineering (Archives)* 48.1-2 (2004), pp. 11–16.

[71] D. W. Matula and L. L. Beck. "Smallest-last ordering and clustering and graph coloring algorithms". In: *Journal of the ACM* 30.3 (1983), pp. 417–427.

[72] R. M. McConnell and J. P. Spinrad. "Modular decomposition and transitive orientation". In: *Discrete Mathematics* 201.1-3 (1999), pp. 189–241.

[73] H. Meyniel. "On the perfect graph conjecture". In: *Discrete Mathematics* 16.4 (1976), pp. 339–342.

[74] H. Meyniel. "A New Property of Critical Imperfect Graphs and some Consequences". In: *European Journal of Combinatorics* 8.3 (1987), pp. 313–316.

[75] M. Middendorf and F. Pfeiffer. "On the complexity of recognizing perfectly orderable graphs". In: *Discrete Mathematics* 80.3 (1990), pp. 327–333.

[76] R. H. Möhring. "Algorithmic aspects of the substitution decomposition in optimization over relations, set systems and Boolean functions". In: *Annals of Operations Research* 4.1 (1985), pp. 195–225.

[77] T. Ohtsuki. "A Fast Algorithm for Finding an Optimal Ordering for Vertex Elimination on a Graph". In: *SIAM Journal on Computing* 5.1 (1976), pp. 133–145.

[78] S. Olariu. "No antitwins in minimal imperfect graphs". In: *Journal of Combinatorial Theory, Series B* 45.2 (1988), pp. 255–257.

[79] S. Olariu. "Paw-free graphs". In: *Information Processing Letters* 28.1 (1988), pp. 53–54.

[80] S. Olariu. "The Strong Perfect Graph Conjecture for pan-free graphs". In: *Journal of Combinatorial Theory, Series B* 47.2 (1989), pp. 187–191.

[81] K. Parthasarathy and G. Ravindra. "The strong perfect-graph conjecture is true for $K_{1,3}$-free graphs". In: *Journal of Combinatorial Theory, Series B* 21.3 (1976), pp. 212–223.

[82] J. M. Robson. "Algorithms for maximum independent sets". In: *Journal of Algorithms* 7.3 (1986), pp. 425–440.

[83] J. M. Robson. *Finding a maximum independent set in time* $O(2^{n/4})$. Tech. rep. Technical Report 1251-01, LaBRI, Université Bordeaux I, 2001.

[84] D. J. Rose. "Triangulated graphs and the elimination process". In: *Journal of Mathematical Analysis and Applications* 32.3 (1970), pp. 597–609.

[85] D. J. Rose, R. E. Tarjan, and G. S. Lueker. "Algorithmic Aspects of Vertex Elimination on Graphs". In: *SIAM Journal on Computing* 5.2 (1976), pp. 266–283.

[86] D. Seinsche. "On a property of the class of n-colorable graphs". In: *Journal of Combinatorial Theory, Series B* 16.2 (1974), pp. 191–193.

[87] M. V. da Silva and K. Vušković. "Decomposition of even-hole-free graphs with star cutsets and 2-joins". In: *Journal of Combinatorial Theory, Series B* 103.1 (2013), pp. 144–183.

[88] G. Szekeres and H. S. Wilf. "An inequality for the chromatic number of a graph". In: *Journal of Combinatorial Theory* 4.1 (1968), pp. 1–3.

[89] R. E. Tarjan. "Depth-First Search and Linear Graph Algorithms". In: *SIAM Journal on Computing* 1.2 (1972), pp. 146–160.

[90] R. E. Tarjan. "Graph Theory and Gaussian Elimination". In: *Sparse Matrix Computations*. Elsevier, 1976, pp. 3–22.

[91] R. E. Tarjan. "Decomposition by clique separators". In: *Discrete Mathematics* 55.2 (1985), pp. 221–232.

[92] R. E. Tarjan and A. E. Trojanowski. "Finding a Maximum Independent Set". In: *SIAM Journal on Computing* 6.3 (1977), pp. 537–546.

[93] R. E. Tarjan and M. Yannakakis. "Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs". In: *SIAM Journal on Computing* 13.3 (1984), pp. 566–579.

[94] N. Trotignon. "Perfect graphs". In: *Topics in Chromatic Graph Theory*. Ed. by L. W. Beineke and R. J. Wilson. Cambridge University Press, pp. 137–160.

[95] N. Trotignon. "Decomposing Berge graphs and detecting balanced skew partitions". In: *Journal of Combinatorial Theory, Series B* 98.1 (2008), pp. 173–225.

[96] N. Trotignon and K. Vušković. "Combinatorial optimization with 2-joins". In: *Journal of Combinatorial Theory, Series B* 102.1 (2012), pp. 153–185.

[97] A. Tucker. "Critical perfect graphs and perfect 3-chromatic graphs". In: *Journal of Combinatorial Theory, Series B* 23.1 (1977), pp. 143–149.

[98] A. Tucker. "Coloring graphs with stable cutsets". In: *Journal of Combinatorial Theory, Series B* 34.3 (1983), pp. 258–267.

[99] A. Tucker. "Coloring perfect $(K_4 - e)$-free graphs". In: *Journal of Combinatorial Theory, Series B* 42.3 (1987), pp. 313–318.

[100] K. Vušković. "Even-hole-free graphs: A survey". In: *Applicable Analysis and Discrete Mathematics* 4.2 (2010), pp. 219–240.

[101] S. H. Whitesides. "An algorithm for finding clique cut-sets". In: *Information Processing Letters* 12.1 (1981), pp. 31–32.

[102] S. H. Whitesides. "A Method for Solving Certain Graph Recognition and Optimization Problems, with Applications to Perfect Graphs". In: *Topics on Perfect Graphs*. Elsevier, 1984, pp. 281–297.

[103] R. J. Wilson. *Introduction to Graph Theory*. New York, NY, USA: John Wiley & Sons, Inc., 1986.

[104] M. Yannakakis. "Computing the Minimum Fill-In is NP-Complete". In: *SIAM Journal on Algebraic Discrete Methods* 2.1 (1981), pp. 77–79.