ALGORÍTMOS, PARÂMETROS E COMPLEXIDADE PARA PROBLEMAS DE PARTIÇÃO EM GRAFOS

GUILHERME DE CASTRO MENDES GOMES

ALGORÍTMOS, PARÂMETROS E COMPLEXIDADE PARA PROBLEMAS DE PARTIÇÃO EM GRAFOS

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Orientador: Vinícius Fernandes dos Santos
Coorientador: Carlos Vinícius Gomes Costa Lima

Belo Horizonte

Junho de 2019

GUILHERME DE CASTRO MENDES GOMES

ALGORITHMS, PARAMETERS AND COMPLEXITY FOR GRAPH PARTITIONING PROBLEMS

Thesis presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Advisor: Vinícius Fernandes dos Santos
Co-Advisor: Carlos Vinícius Gomes Costa Lima

Belo Horizonte

June 2019

Gomes, Guilherme de Castro Mendes

C667h Algorithms, Parameters and Complexity for Graph Partitioning Problems / Guilherme de Castro Mendes Gomes. — Belo Horizonte, 2019
xxiv, 123 f. : il. ; 29cm

Tese (doutorado) — Universidade Federal de Minas Gerais
Orientador: Vinícius Fernandes dos Santos

1. Equitable Coloring. 2. Clique Coloring. 3. Biclique Coloring. 4. Star Graph. 5. Vertex Partitioning. 6. Matching Cut. 7. Exact Algorithms. 8. Parameterized Complexity. I. Título.

CDU 123

[Folha de Aprovação]

Quando a secretaria do Curso fornecer esta folha,
ela deve ser digitalizada e armazenada no disco em formato gráfico.

Se você estiver usando o `pdflatex`,
armazene o arquivo preferencialmente em formato PNG
(o formato JPEG é pior neste caso).

Se você estiver usando o `latex` (não o `pdflatex`),
terá que converter o arquivo gráfico para o formato EPS.

Em seguida, acrescente a opção `approval={`*nome do arquivo*`}`
ao comando `\ppgccufmg`.

Se a imagem da folha de aprovação precisar ser ajustada, use:
`approval=[`*ajuste*`][`*escala*`]{`*nome do arquivo*`}`
onde *ajuste* é uma distância para deslocar a imagem para baixo
e *escala* é um fator de escala para a imagem. Por exemplo:
`approval=[-2cm][0.9]{`*nome do arquivo*`}`
desloca a imagem 2cm para cima e a escala em 90%.

To my family.

Acknowledgments

I'd like to deeply thank everyone that somehow was a part of this process. My family, Cláudia, Mauro and Paulinha, for their support, from the moment I started this doctorate until now. All my friends, who were always full of jokes (specially during the apparently endless mid-afternoon breaks), encouragement, ideas and discussions, making the daily research feel much lighter and enjoyable than it could otherwise have been. Jacqueline, for listening to my endless musings about graphs with a million properties that made exactly zero sense to anyone - me included - and being exactly who she is.

I'd really like to thank Vinícius for accepting me as his student, specially after a year and a half spent on a completely different subject and not much that could be salvaged. I'd also like to thank Carlos for being such a massive addition to the team, squashing bugs on the more delicate proofs, presenting outright great advice, speeding things up considerably. Last, but definitely not least, I'd like to thank Ignasi for supervising me during my internship at LIRMM, even though we hadn't known each other for long; it was a wonderful experience, both personally and prefessionally. I'd like to thank all of you for being great friends.

It was an honour and priviledge to work with and live alongside each of you. Nothing would be the same without you.

"There is no struggle too vast,
no odds too overwhelming,
for even should we fail–
should we fall – we will know
that we have lived."
(Steven Erikson)

Resumo

Problemas de partição em grafos modelam diferentes tarefas do mundo real, como aloção de recursos ou design de redes tolerantes a falhas. Geralmente, esse problemas são NP-difíceis, e projetar algoritmos cuja complexidade dependa apenas do tamanho do grafo de entrada levam a tempos de exacução impraticáveis. A complexidade parametrizada aborda esse desafio por meio do projeto de algoritmos que funcionam bem em apenas algumas instancias do problem. Nesta tese, cinco problemas em teoria dos grafos foram estudados do ponto de vista da complexidade computacional: coloração equilibrada, clique coloração, biclique coloração, d-corte, e reconhecimento de grafos estrela.

Coloração equilibrada foi investigada em termos de grafos cordais, grafos bloco e algumas subclasses. Foi provado que coloração equilibrada é W[1]-difcil para grafos bloco de diametro limitado e para a união disjunta de grafos split, quando parametrizado pelo número de cores e treewidth; e W[1]-difcil para grafos de intervalo livros de $K_{1,4}$ quando parametrizado por treewidth, número de cores e grau máximo, generalizando os resultados Fellows et al. (2014) por meio de reduções muito mais simples. Usando resultados alteriores de Werra (1985), uma dicotomia para a complexidade de coloração equilibrada de grafos cordais baseada na maior estrela induzida foi estabelecida. Finalmente, é demonstrado que o problema de coloração equilibrada é FPT quando parametrizada pelo treewidth do grafo complementar. É apresentado o primeiro algoritmo $\mathcal{O}^*(2^n)$ para biclique coloração, que faz uso de propriedades associadas ao hipergrafo biclique e do princípio da inclusão exclusão. Algoritmos parametrizados por diversidade de vizinhança são discutidos para os problemas de clique e biclique coloração, sendo esses os primeiros algoritmos parametrizados para esses problemas. Biclique coloração foi apenas recentemente introduzida na literatura, e muito do trabalho exploratório em diferentes classes de grafos ainda deve ser feito.

Foi definido e investigado o problema d-corte, uma generalização natural do problema de corte emparelhado. São generalizados e, em alguns casos, melhorados, vários resultados do estado-da-arte para corte emparelhado. Em particular, são apresentados

reduções de NPdificuldade para d-corte em grafos $(2d+2)$-regulares, um algoritmo polinomial para grafos de grau máximo $d+2$, e um algoritmo exato exponencial marginalmente mais eficiente que a estratégia ingênua por força bruta, cuja complexidade é $\mathcal{O}^*(2^n)$. Em seguida, são dados algoritmos FPTpara diversos parâmetros: número máximo de arestas cruzando o corte, treewidth, distância para cluster e distância para co-cluster. A principal contribuição é um kernel polinomial para d-corte quando parametrizado pela distância para cluster; ao mesmo tempo, descartamos a existência de um kernel polinomial quando parametrizado simultaneamente por treewidth, grau máximo e número máximo de arestas cruzando o corte.

Por fim, grafos estrela - grafos de interseção das estrelas maximais de um grafo - foram discutidos e definidos em termos de uma cobertura de arestas por cliques, com o intuito de que tal class possa ser uma ferramenta útil na investigação de grafos biclique. O problema natural de reconhecimento foi apresentado e, apesar de não termos uma prova concreta de seu pertencimento em NP, um algoritmo de verificação que faz uso da cobertura de arestas por cliques característica é apresentado. De interesse, vale destacar a conexão entre grafos estrela e o quadrado de grafos livres de triangulo. Essa relação é explorada para demonstrar a NP-completude do reconhecimento de grafos estrela de grafos com cintura exatamente 4. Próximos objetivos incluem a determinação de pertencimento (ou não) do problema, no caso geral, em NP. Instâncias particulares de grafos biclique podem render conexões semelhantes com outros problemas; tal exploração configura entre trabalhos futuros nesse tópico.

Abstract

Graph partitioning problems are used to model many different real world tasks, such as the allocation of resources or designing fault tolerant networks. Usually, however, they are NP-hard problems, and designing algorithms with complexity solely dependent on the size of the input graph leads to impractical running times. Parameterized complexity approaches this challenge by designing algorithms that work well for some instances of the problem. In this thesis, five graph theoretical problems were studied from the complexity point of view: equitable coloring, clique coloring, biclique coloring, d-cut, and star graph recognition.

Equitable coloring was investigated in terms of chordal graphs, block graphs and some of its subclasses. It is proved that Equitable Coloring is W[1]-hard for block graphs of bounded degree and for disjoint union of split graphs when parameterized by the number of colors and treewidth; and W[1]-hard for $K_{1,4}$-free interval graphs when parameterized by treewidth, number of colors and maximum degree, generalizing a result by Fellows et al. (2014) through a much simpler reduction. Using a previous result due to Dominique de Werra (1985), a dichotomy for the complexity of equitable coloring of chordal graphs based on the size of the largest induced star is established. Finally, it is shown that Equitable Coloring is FPT when parameterized by the treewidth of the complement graph. The first $\mathcal{O}^*(2^n)$ time exact algorithm for biclique coloring was presented, which makes use of properties of the associated biclique hypergraph and the powerful inclusion-exclusion principle. Algorithms parameterized by neighbourhood diversity were discussed for both clique and biclique coloring, being the first parameterized algorithms for these problems. Biclique coloring was only recently introduced in the literature, and much of the exploratory work on different graph classes remains to be done.

A natural generalization of the Matching Cut problem, called d-Cut is defined and investigated. Namely, an NP-hardness reduction for d-Cut on $(2d + 2)$-regular graphs is given, followed by a polynomial time algorithm for graphs of maximum degree at most $d+2$. The degree bound in the hardness result is unlikely to be improved,

as it would disprove a long-standing conjecture in the context of internal partitions. FPTalgorithms for several parameters are given: the maximum number of edges crossing the cut, treewidth, distance to cluster, and distance to co-cluster. In particular, the treewidth algorithm improves upon the running time of the best known algorithm for Matching Cut. Our main technical contribution is a polynomial kernel for d-Cut for every positive integer d, parameterized by the distance to a cluster graph. The existence of polynomial kernels when parameterizing simultaneously by the number of edges crossing the cut, the treewidth, and the maximum degree is also ruled out. An exact exponential algorithm slightly faster than the naive brute force approach running in time $\mathcal{O}^*(2^n)$ is provided. We also discuss two other generalizations of Matching Cut which appear to be considerably more challenging than d-Cut.

Finally, star graphs - intersection graph of maximal stars of a graph - were first discussed and defined in terms of a characteristic edge clique cover, in the hope that they could be a useful tool on the investigation of biclique graphs. A bound on the size of minimal pre-images by a quadratic function on the number of vertices of the star graph is presented, then a Krausz-type characterization for this graph class is described; the combination of these results yields membership of the recognition problem in NP. Some properties of star graphs are presented. In particular, it is shown that all graphs in this class are biconnected, that every edge belongs to at least one triangle, a characterization of the structures the pre-image must have in order to generate degree two vertices, and the diameter of the star graph is bounded by a function of the diameter of its pre-image. Finally, a monotonicity theorem is provided, which we apply to generate all star graphs on at most eight vertices and prove that the classes of star graphs and square graphs are not properly contained in each other.

List of Figures

List of Tables

Contents

Chapter 1

Introduction and Preliminaries

Graphs are a mathematical tool mainly used to model situations where objects have some sort of interaction with each other. As such, they naturally arise on a plethora of problems from the most varied domains, ranging from geographical data to computer science, physics, chemistry and biology. Computer science, in particular, is heavily reliant on graphs and their many properties, being a core component of many database, network and artificial intelligence algorithms. In the information age, which encompasses the later 20th and early 21st centuries, the development of communication technologies has been a (if not the) focus point for human society. It is quite hard conceive a more fitting structure to the modelling of communication networks than a graph; the schematics of such a network are pretty much a drawing of a graph. In more recent years, the explosive popularity of online social networks has created a demand for extremely efficient and scalable implementations of graph structures and algorithms.

Beside their wide applicability range, graphs by themselves have been the subject of countless investigations. Much of the foundation of modern graph theory was laid by some of the greatest mathematicians of the nineteen hundreds, such as Bill Tutte, Claude Berge, and Paul Erdős, with many milestone results in structural theory. Their work transformed graph theory from a small topic in combinatorics into one of the underpinning fields of applied mathematics, with connections to older, more established, mathematical domains. Another growth spurt in the area is attributed to the rapid expansion of computer science and its demands for efficient algorithms. Various computational problems quickly became graph theoretical ones, leading to many profound insights, which in turn presented new venues of investigation and even more important questions in graph theory, creating an ongoing virtuous cycle of research.

Most of this thesis is devoted to the study of some graph theoretical problems

belonging to the broad class of partitioning problems. Members of this family seek a partition of the graph's vertices and/or edges such that each part of the partition satisfies some problem-specific properties. The focus of this work is on two branches of partitioning problems: colorings and cuts. In a coloring problem, the goal is to partition (i.e. color) the vertices (or edges) of a graph such that each set of the partition, individually, satisfies some condition. For the classical VERTEX COLORING problem, the goal is to partition the graph's vertex set such that, inside each member of the partition there are no two adjacent vertices. By further desiring that the size of each partition member be as close as possible to each other, the EQUITABLE COLORING problem, which appears to be much harder to solve even for graph classes where classical vertex coloring is efficiently solvable, is generated. One may also impose the constraint that no maximal induced subgraph be entirely contained in a single set of the partition. For example, it may be required that no maximal clique or biclique (complete bipartite graph) of the given graph may be monochromatic generating the problems known as CLIQUE COLORING and BICLIQUE COLORING, respectively.

On the other hand, there are cut problems. While coloring problems are concerned with each set of the partition, cut problems usually define properties among different sets of the partition, usually involving the disconnection of a subset of vertices. Certainly, the most well known cut problem is MINIMUM CUT, where the goal is to disconnect a given pair of vertices through the removal of the smallest possible subset of edges. This problem has been a component of numerous optimization algorithms, usually as a subroutine of more sophisticated heuristics, cutting plane, or pricing techniques. A lesser known relative of MINIMUM CUT is the MATCHING CUT problem; in this case, a bipartition of the vertex set of a graph such that each vertex has at most one neighbor across the cut is sought. Much work was done in this problem in recent years, building upon results of the 1980s, specially in terms of parameterized complexity. Many possible generalizations come to mind simply by looking at the definition. For instance, one could ask for a multipartition of the vertex set so that between each pair of sets there is a matching cut, or maybe a bipartition is still desired, but now a vertex may have more neighbors across the cut. Another cut problem with degree constraints and quite similar to the latter is known as INTERNAL PARTITION, where the goal is to find a bipartition of the vertices so that no vertex has more than half of its neighbors across the cut.

A secondary object of study are graph classes defined as intersection graphs. While it is known that every graph is the intersection graph of the subgraphs of some graph, constraints on the intersecting subgraph family impose all sorts of properties to the resulting graph. For example, the literature is rife with works on clique graphs

– the intersection graph class of maximal cliques of some graph, with results ranging from characterizations to other structural aspects; while biclique graphs are a far more recently studied class. The intersection graphs of these maximal structures are usually hard to characterize and provide few algorithmically useful insights on the topology of the underlying graph. Nevertheless, their understanding was crucial to the development of a consistent theory that is used to describe important classes, such as chordal graphs (intersection graphs of subtrees of a tree), cographs (intersection graphs of paths of a grid) and line graphs (intersection graphs of the edges of a graph).

In short, the study presented in this thesis, as many algorithmic graph theory works, is done from the structural and complexity point of view. While the reported results are mainly algorithms, hardness proofs, and kernelization techniques, most of these are heavily reliant on structural aspects, either by supporting themselves on previous results of the literature or by being structural themselves. Specifically, four partitioning problems were investigated: equitable coloring, clique coloring, biclique coloring, and a generalization of matching cut. For intersection graphs of maximal structures, motivated by the difficulty in working with biclique graphs, star graphs (intersection graph of maximal stars of a graph) are introduced and investigated. The following is a summary of the topics and results discussed in this thesis.

- The remainder of this chapter defines most of the notation used throughout this work. It also revisits some of the main concepts employed throughout this thesis.

- Chapter 2 tackles coloring problems. For equitable coloring, some W[1]-hardness results are provided: for block graphs of bounded diameter when parameterized by treewidth and maximum number of colors, for $K_{1,4}$-free interval graphs when parameterized by treewidth, maximum number of colors and maximum degree, and for disjoint union of complete multipartite graphs when parameterized by treewidth and maximum number of colors. Some algorithms for equitable coloring are also described; in particular, it is shown that the problem admits an XP algorithm for chordal graphs when a parameterized by the maximum number of colors, a constructive polynomial time algorithm to equitably color claw-free chordal graphs, and an FPT algorithm parameterized by the treewidth of the complement graph. Also in this chapter, both clique and biclique colorings are discussed. The first exact exponential time algorithm for biclique coloring, which builds upon ideas used for clique coloring, is presented. Then, kernelization algorithms for clique and biclique coloring when parameterized by neighborhood diversity are given; using results on covering problems, an FPT algorithm under the same parameterization is obtained for clique coloring, which has optimal running

time, up to the base of the exponent, unless the Exponential Time Hypothesis fails. For biclique coloring, an FPT algorithm is given, but when parameterized by maximum number of colors and neighborhood diversity.

- Chapter 3 discusses generalizations of the matching cut problem. Most of the chapter is devoted to the study of the d-cut problem. Among the presented results are included an NP-hardness proof for $(2d+2)$-regular graphs – which has an important connection to a conjecture on the context of internal partitions – as well as a polynomial time algorithm for graphs of maximum degree $d+2$. FPT algorithms for several parameters are then given; namely: the maximum number of edges crossing the cut, treewidth, distance to cluster, and distance to co-cluster. The algorithm parameterized by treewidth improves upon the running time of the best known algorithm for MATCHING CUT. Afterwards, building on techniques employed for MATCHING CUT, a polynomial kernel for d-CUT for every positive integer d, parameterized by the distance to a cluster graph is shown. The existence of polynomial kernels when parameterizing simultaneously by the number of edges crossing the cut, the treewidth, and the maximum degree is ruled out. Also, an exact exponential algorithm slightly faster than the naive brute force approach is described. We conclude the chapter with some remarks on two other generalizations of MATCHING CUT, with some results in another version, which we called ℓ-NESTED MATCHING CUT, and a brief discussion on a much harder problem, namely p-WAY MATCHING CUT.

- Chapter 4 deals with star graphs, the intersection graph of the maximal stars of a graph, and with star-critical graphs which are minimal with respect to the star graph they generate. The chapter begins with a bound on the number of vertices of star-critical graphs by a quadratic function of the size of its set of maximal stars. Afterwards, a Krausz-type characterization is given; both results are combined to show that the recognition problem belongs to NP. Then, a series of properties of star graphs are proved. In particular, it is shown that they are biconnected, that every edge belongs to at least one triangle, the structures that the pre-image must have in order to generate degree two vertices are characterized, and bound the diameter of the star graph with respect to the diameter of its pre-image is given. Finally, a monotonicity theorem is provided, which is used to generate all star graphs on no more than eight vertices and prove that the class of star graphs and square graphs are not properly contained in each other.

The following table summarizes the submissions and collaborators of each chapter.

Chapter	Title	Venue	Status	Collaborators
2	Parameterized Complexity of Equitable Coloring	Discrete Mathematics & Theoretical Computer Science	Accepted	Carlos V. Gomes & Vinícius dos Santos
3	Finding cuts of bounded degree: complexity, FPT and exact algorithms, and kernelization	International Symposium on Parameterized and Exact Computation	Under Review	Ignasi Sau
4	Intersection graph of maximal stars	Discrete Applied Mathematics	Under Review	Carlos V. Gomes & Marina Groshaus & Vinícius dos Santos

Table 1: Submissions and Collaborators.

1.1 Basic Definitions

We denote by $[n] = \{1, \ldots, n\}$. A *(multi)family* is a (multi)set of sets. The *power set* 2^S of a set S is the family of all subsets of S. A *k-partition* of S into k sets is denoted by $S \sim \{S_1, \ldots, S_k\}$ such that $S_i \cap S_j = \emptyset$ and $\bigcup_{i \leq k} S_i = S$. A *k-(multi)cover* of S is a (multi)family $\{S_1, \ldots, S_k\}$ of subsets of S such that $\bigcup_{i \in [k]} S_i = S$. A (multi)family $\mathcal{F}$ satisfies the *Helly condition* or *Helly property* if and only if, for every pairwise intersecting subfamily $\mathcal{F}'$ of $\mathcal{F}$, $\bigcap_{F \in \mathcal{F}'} F \neq \emptyset$.

A *simple graph* of *order* (or *size*) n is an ordered pair $G = (V(G), E(G))$, where $V(G)$ is its *vertex set* of cardinality n and its *edge set*, $E(G)$, is a family of subsets of $V(G)$, each of cardinality two. A graph is *trivial* if $|V(G)| = 1$. Instead of $\{u, v\} \in E(G)$, we identify an edge by uv, simply due to convenience. Moreover, when there is no ambiguity, we denote $V(G)$ by V, $E(G)$ by E, $|V|$ as n and $|E|$ as m.

We say that two vertices $u, v \in V$ are *adjacent* or *neighbours* if $uv \in E$. A graph $G' = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$. If $E' = \{uv \in E \mid u, v \in V'\}$ we say that V' *induces* G', $G' = G[V']$ and that G' is the *induced subgraph* of G by V. For simplicity, we denote by $G - v$ the graph $G[V(G) \setminus \{v\}]$ and, similarly, for $S \subseteq V(G)$, $G \setminus S$ is equivalent to $G[V(G) \setminus S]$.

The *open neighbourhood*, or just *neighbourhood* of a vertex v in G is given by $N_G(v) = \{u \mid uv \in E(G)\}$, its *closed neighbourhood* by $N_G[v] = N_G(v) \cup \{v\}$ and its *degree* by $\deg_G(v) = |N_G(v)|$. A vertex is *simplicial* if its neighbours are pairwise adjacent. For a set $S \subseteq V$, we denote its open and closed neighbourhood as $N_G(S) = \bigcup_{v \in S} N_G(v) \setminus S$, $N_G[S] = N_G(S) \cup S$ and $\deg_G(S) = |N_G(S)|$, respectively. The *complement* $\overline{G}$ of G is defined as $V(\overline{G}) = V(G)$ and $E(\overline{G}) = \{uv | uv \notin E\}$. Given a graph G, we denote its *maximum degree* by $\Delta(G)$ and *minimum degree* by $\delta(G)$.

Two vertices u, v are *false twins* if $N_G(u) = N_G(v)$ and *true twins* if $N_G[u] =$

$N_G[v]$. u, v are of the same *type* if they are either true or false twins. Being of the same type is an equivalence relation [Ganian, 2012], and the number of different types on a graph G is called its *neighbourhood diversity*, $\mathsf{nd}(G)$.

Two graphs G and H are *isomorphic* if and only if there is a bijection $f : V(G) \mapsto V(H)$ such that $uv \in E(G) \Leftrightarrow f(u)f(v) \in E(H)$. We denote isomorphism by $G \simeq H$. A graph G is said to be *free* of a graph H, or H-free, if there is no induced subgraph G' of G such that G' and H are isomorphic.

The *path* of length k, or P_k, is a graph with k vertices $v_1 \dots v_k$ such that $v_i v_j \in E(P_k)$ if and only if $j = i + 1$. Moreover, we say that v_1, v_k are the *extremities*, or *endvertices*, of P_k and all other v_i are its *inner* vertices. The length of a path is the number of edges contained in it, that is, P_k has length $k - 1$. An *induced path* of G is a subgraph G' of G that is isomorphic to a path. A *cycle* with $k \geq 3$ vertices is a path with k vertices plus the edge $v_k v_1$; analogously, the length of a cycle is defined as the number of edges it contains. An *induced cycle* of G is an induced subgraph G' of G that is isomorphic to a cycle. A *chord* in a cycle C of length at least 4 is an edge between two non-consecutive vertices of C. The *girth* of G, denoted by $\mathsf{girth}(G)$, is the length of the smallest induced cycle of G. A *hole* is a chordless cycle of length at least 4; it is an even-hole if it has an even number of vertices, or an odd-hole, otherwise. An *anti-hole* is the complement of a hole. G is *acyclic* if and only if there is no induced cycle in G. A *matching* is a set of edges such no two share a common endpoint. A maximum matching is said to be *perfect* if every vertex of the graph is contained in one edge of the matching.

A graph G is *connected* if and only if there is an induced path between every pair $u, v \in V$. A *connected component*, or simply a *component*, of G is a maximal connected induced subgraph of G. Given a graph G, the *distance* $\mathsf{dist}_G(u, v)$ between two vertices u, v in G is the minimum number of edges in any path between them. If u, v are in different components, we say that $\mathsf{dist}_G(u, v) = \infty$. The *diameter* of a connected graph G is defined as the length of the longest shortest path between any pair of vertices $u, v \in V(G)$. The *k-th power* G^k of a graph G is the graph where $V(G^k) = V(G)$ and $E(G^k) = \{uv \mid \mathsf{dist}_G(u, v) \leq k\}$. When $k = 2$, G^2 is also called the *square* of graph G and G is called the *square root* of G^2. The class of all graphs that admit a square root is called *square graphs*. When the graph in question is clear, we will omit the G subscript.

An *articulation point*, *cut point* or *cut vertex* of a connected graph G is a vertex v such that $G - v$ is disconnected. A *bridge* is an edge of G whose removal increases the number of connected components of G. G is *biconnected* if G is connected and does not have a cut vertex. A *cutset* of a connected graph G is a set $S \subset V$ such that

$G \setminus S$ is disconnected. In particular, a cut vertex is a cutset of size one.

The *complete* graph K_n of order n is a graph where every pair of vertices is adjacent. A *clique* of G with n vertices is an induced subgraph of G isomorphic to K_n. An *independent set* of G with n vertices is an induced subgraph of G isomorphic to $\overline{K_n}$. We denote by $\omega(G)$ and $\alpha(G)$ the size of the maximum induced clique and maximum independent set of a given G. A *vertex cover* of G is a subset $U \subseteq V(G)$ such that, for every edge $uv \in E(G)$, at least one of u, v belongs to U. The size of a minimum vertex cover of a graph is denote by $\tau(G)$, and can be computed in $\mathcal{O}\big(1.2738^{\tau(G)} + \tau(G)|V(G)|\big)$ time.

A graph is a *cluster graph* if all of its connected components are cliques. Analogously, a graph is a *co-cluster graph* if its complement is a cluster graph. The *distance to cluster* (resp. *co-cluster*) of a graph G, denoted by $\mathsf{dc}(G)$ (resp. $\mathsf{d\overline{c}}(G)$), is the size of the smallest subset of vertices U of G such that $G - U$ is a (co-)cluster graph. These parameters can be computed in $\mathcal{O}\big(1.92^{\mathsf{dc}(G)}n^2\big)$ time and $\mathcal{O}\big(1.92^{\mathsf{d\overline{c}}(G)}n^2\big)$ time, respectively [Boral et al., 2016]. It is quite easy, however, to obtain a 3-approximation for them in polynomial time, it suffices to note that a graph is a cluster graph if and only if it is P_3-free, so while there is some P_3 in the graph, it is sufficient to remove all three vertices. The above values are examples of *structural graph parameters*. Determining certain parameters of a generic graph G is efficient (such as $\Delta(G)$ and $\delta(G)$); however, others (such as $\omega(G)$ and $\alpha(G)$) are widely believed to be hard to ascertain.

A graph G is *bipartite* if $V(G) \sim \{X, Y\}$ such that both X and Y are independent sets. Such property implies that a graph is bipartite if and only if it is C_{2k+1}-free, for any $k \geq 1$. A *biclique* K_{n_1,n_2} is a bipartite graph with $|X| = n_1$, $|Y| = n_2$ and $uv \in E(G)$ for every pair $u \in X$ and $v \in Y$. A *star* is a biclique with $|X| = 1$ and $|Y| \geq 1$. Clearly, we can also define *induced bicliques* and *induced stars* much like induced cliques. A graph is *multipartite* if $V(G) \sim \{X_1, \ldots, X_p\}$ and X_i is an independent set for all i; it is a *complete multipartite* graph if $uv \in E(G)$ whenever $u \in X_i$, $v \in X_j$ and $i \neq j$.

A *hypergraph* $\mathcal{H} = (V, \mathcal{E})$ is a natural generalization of a graph. That is, $V(\mathcal{H})$ is its vertex set and $\mathcal{E} \subseteq 2^V$ its *hyperedge* set [Berge, 1984]. A graph G is said to be a *host* of $\mathcal{H}$ if $V(G) = V(\mathcal{H})$, every hyperedge of $\mathcal{H}$ induces a connected subgraph of G and every edge of G is contained in at least one hyperedge of $\mathcal{H}$. A hypergraph is *k-uniform* if all of its hyperedges have the same size.

A *transversal* of a hypergraph $\mathcal{H}$ is a set $X \subseteq V(\mathcal{H})$ such that, for every hyperedge $\varepsilon \in \mathcal{E}(\mathcal{H})$, $X \cap \varepsilon \neq \emptyset$. If X is not a transversal we say that it is an *oblique*.

The *clique hypergraph* $\mathcal{H}_{\mathcal{C}}(G)$ of a graph G is the hypergraph on the same vertex set of G and with hyperedge set equal to the family of maximal cliques of G. Similarly, the *biclique hypergraph* $\mathcal{H}_{\mathcal{B}}(G)$ of a graph G is the hypergraph on the same vertex set

of G and with hyperedge set equal to the family of maximal bicliques of G.

A *tree decomposition* of a graph G is defined as the pair $\mathbb{T} = (T, \mathcal{B} = \{B_j \mid j \in V(T)\})$, where T is a tree and $\mathcal{B} \subseteq 2^{V(G)}$ is a family satisfying $\bigcup_{B_j \in \mathcal{B}} B_j = V(G)$ [Robertson and Seymour, 1986]; for every edge $uv \in E(G)$ there is some B_j such that $\{u, v\} \subseteq B_j$; for every $i, j, q \in V(T)$, if q is in the path between i and j in T, then $B_i \cap B_j \subseteq B_q$. Each $B_j \in \mathcal{B}$ is called a *bag* of the tree decomposition. The *width* of a tree decomposition is defined as the size of a largest bag minus one. The *treewidth* $\mathsf{tw}(G)$ of a graph G is the smallest width among all valid tree decompositions of G [Downey and Fellows, 2013]. If $\mathbb{T}$ is a rooted tree, by G_x we will denote the subgraph of G induced by the vertices contained in any bag that belongs to the subtree of $\mathbb{T}$ rooted at bag x. An algorithmically useful property of tree decompositions is the existence of a so called *nice tree decompositions* of width $\mathsf{tw}(G)$.

Nice tree decomposition *A tree decomposition $\mathbb{T}$ of G is said to be* nice *if it is a tree rooted at, say, the empty bag $r(T)$ and each of its bags is from one of the following four types:*

1. Leaf node*: a leaf x of $\mathbb{T}$ with $B_x = \emptyset$.*
2. Introduce node*: an inner bag x of $\mathbb{T}$ with one child y such that $B_x \setminus B_y = \{u\}$.*
3. Forget node*: an inner bag x of $\mathbb{T}$ with one child y such that $B_y \setminus B_x = \{u\}$.*
4. Join node*: an inner bag x of $\mathbb{T}$ with two children y, z such that $B_x = B_y = B_z$.*

1.2 Parameterized Complexity

We discuss problems in different complexity classes; in particular, we work with the usual classes P, NP,and the *polynomial hierarchy* [Stockmeyer, 1976]. We say that an algorithm is *efficient* if its running time is bounded by a polynomial on the size of the input and that a problem belonging to NP-hard is most likely intractable. As such, our complexity results will be given either by efficient algorithms or polynomial reductions from NP-hard problems.

Besides the classic complexity classes, we also deal with the field of *parameterized complexity* (or *multivariate complexity*). In this area, algorithms are designed and analyzed not only with respect to the size of the input object, but also with other *parameters* of the input. Parameters come in all sorts of flavors. Many decision problems usually have some integer quantity representing a constraint of the problem. For instance, VERTEX COVER – one of the classical examples of success of parameterized

complexity asks for a set of size at most k of vertices covering all the edges of the graph. Other parameters are less problem specific and relate to the structure of the graph, such as diameter or maximum degree. The most prominent of these examples, however, is the graph parameter treewidth, which played a pivotal role in the theory of graph minors. Other previously discussed structural parameter is neighborhood diversity, distance to cluster and distance to co cluster.

A problem is said to be *fixed-parameter tractable* (or FPT) when *parameterized by* k if there is an algorithm with running time $f(k)n^{\mathcal{O}(1)}$, where n is the size of the input object. We denote complexities of this form by $\mathcal{O}^*(f(k))$. In fact, we shall use $\mathcal{O}^*(\cdot)$ to omit polynomial factors of the running time; that is, an algorithm with complexity $2^{f(n)}\text{poly}(n)$ is said to execute in $\mathcal{O}^*\big(2^{f(n)}\big)$. In a slight abuse of notation, k is simultaneously the parameter we are working with and the value of such parameter. An instance of a parameterized algorithm is, therefore, the pair (x, k), with x the input object and k as previously defined. The class of all problems that admit an FPT algorithm is the class FPT. If an algorithm has running time $\mathcal{O}\big(n^{f(k)}\big)$, for some computable function on k, we say it is an XP(slicewise polynomial) algorithm, and the corresponding problem it solves is in XP.

Much like classical univariate theory, some problems do not appear to admit an FPTalgorithm for certain parameterizations. In particular, its widely believed that finding a clique of size k in a graph, parameterized by k, is not in FPT. In an analogue to the classical case, hardness results are usually given by what are called *parameterized reductions*.

Parameterized Reduction *A parameterized reduction from problem* Π *to problem* Π' *is a transformation from an instance* (x, k) *of* Π *to an instance* (x', k') *of* Π' *such that:*

1. *There is a solution to* (x, k) *if and only if there is a solution to* (x', k')*;*

2. $k' \leq g(k)$ *for some computable function* g*;*

3. *The transformation's running time is* $\mathcal{O}^*(f(k))$.

Note that the constraints imposed by parameterized reductions are quite similar to those imposed by polynomial reductions. We ask that k' does not depend on $|x|$ - which doesn't always happen with polynomial reductions - but, at the same time, allow FPT time for the transformation, instead of the more restrictive polynomial time. These differences imply that polynomial reductions and parameterized reductions are

incomparable, with some rare cases where the transformation is both polynomial and parameterized.

Unlike the theory of NP-completeness, where most hard problems are equivalent to each other under polynomial reductions, in parameterized complexity problems seem to be distributed along a hierarchy of difficulty. Before handling the classes themselves, we must first define the problems of parameterized complexity that play the same role as SATISFABILITY for the classical theory.

The *depth* of a circuit is the length (in terms of number of gates) of the longest path from any one variable to the output. The *weft* of a circuit is the the maximum number of gates with more than 2 input variables in any path from any one variable to the circuit's output. The circuits with weft t and depth d, denoted by $\text{WCS}_{t,d}$, will be the fundamental problems of the t-th level of our hierarchy.

WEIGHTED CIRCUIT SATISFABILITY OF WEFT t AND DEPTH d ($\text{WCS}_{t,d}$)
Instance: A Boolean circuit C with n variables, weft t and depth d.
Parameter: A positive integer k.
Question: Is C satisfiable with exactly k variables set to TRUE?

W-hierarchy *For $t \geq 1$, a parameterized problem Π is in $\mathsf{W}[t]$ if there is a parameterized reduction from $\text{WSC}_{t,d}$ to it, for some $d \geq 1$. Moreover,*

$$\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subset \mathsf{XP}$$

1.2.1 Kernelization

One of the broadest class of techniques to be found in the realm of computing is perhaps that of *pre-processing*. Every real system, in one way or another, employs routines that try to prune the search space or reduce the input instance as much as possible before doing any heavy lifting. Such is the case with most optimization suites, such as CPLEX and Gurobi, where dozens upon dozens of pre-processing methods are readily available and in many cases successfully eliminate large chunks of the input before trying to solve the integer program directly. Furthermore, in many cases, simple heuristics or algorithms with terrible worst case running times perform surprisingly well, and, in many cases, there was no theoretically sound approach to explain this phenomenon. This lack of work on the subject is explained by the fact that, if an instance of a NP-hard problem can be reduced in polynomial time to one of bounded size, then P = NP [Fomin et al., 2019]. With the advent of parameterized algorithms,

however, the situation is changing drastically. Using this framework it has become possible to derive upper on lower bounds on the sizes of the instances obtained after a set of pre-processing rules have been applied. We define the notions of kernels and kernelization below.

Kernelization *A kernelization algorithm is an algorithm that takes as input an instance* (x, k) *of a parameterized problem* Π *and its output is an equivalent instance* (x', k') *of* Π *such that* $|x'| \leq f(k)$ *and* $k' \leq g(k)$*, for some pair of computable functions* f, g*; instance* (x', k') *is called the kernel of* (x, k)*.*

It is not hard to see that a parameterized problem is in FPT if and only if it admits a kernel. Not all kernels are equal, and a natural desire is for the best (i.e. smallest) possible kernel. The size of the kernel is measured by the dependency of the kernel on the parameter – that is, a kernel that satisfies $|x'| \leq 4k$ is much better than a kernel with $|x'| \leq k^2$. If this dependency is linear, we say that we have a *linear kernel*, if $f(k)$ is a quadratic function, than the kernel is *quadratic*, and so on. Let k be the natural parameter for the following problems. Some famous examples of problems and their kernels include: Vertex Cover, which admits a kernel of size $2k$; Max 3-Satisfability, which has a kernel on $6k$ variables and $2k$ clauses; Independent Set on planar graphs, with a kernel of size $4(k-1)$; meanwhile, for Dominating Set on graphs of girth at least five, there is a cubic kernel, but no known subcubic one.

The bound on the instance size, however, can be exponential. For instance, Matching Cut parameterized by the number of edges crossing the cut does not have a polynomial kernel [Komusiewicz et al., 2018]. For some time, there were no techniques to prove that a parameterized problem does *not* admit a polynomial kernel; this changed, however with the seminal work of Bodlaender et al. [2009], where the composition and distillation techniques were first discussed, being was further deepened by Hermelin and Wu [2012] and Bodlaender et al. [2014], where weak and cross-compositions were described. All of these techniques, however, make use of some well established hypothesis about classical complexity classes. For instance, distillation is based on assumption that the polynomial hierarchy does not collapse to the third level; weak-composition and cross-composition rely on the hypothesis that NP $\subseteq$ coNP/poly. Despite appearing strong assumptions, if either of these hypotheses fail the implications would reverberate through much of theoretical computer science, and not only parameterized complexity.

For further reading and other more insightful discussions on the subjects of parameterized complexity and kernelization, we point to [Downey and Fellows, 2013;

Cygan et al., 2015a; Fomin et al., 2019] from where most of the given definitions come from.

1.3 Explicit Running Time Lower Bounds

Both the theory of NP-completeness and W[1]-hardness give us evidence that no polynomial or FPT algorithm may exist for a myriad of problems. However, simply assuming that $\mathsf{P} \neq \mathsf{NP}$ or that $\mathsf{FPT} \neq \mathsf{W}[1]$ seems to not be enough to prove statements about asymptotic lower bounds on the running time of an algorithm. All is not lost, but we do need to make some additional complexity assumptions.

In their groundbreaking work, Impagliazzo and Paturi [2001] give many key insights and tools which have been broadly used across the field of algorithms and parameterized complexity to prove that long known algorithms are probably optimal. Specifically, they prove what is known as the *Sparsification Lemma*, described below. A logical formula ϕ on n variables an m clauses is in *Conjunctive Normal Form* (CNF) if $\phi = \bigwedge_{i=1}^{m} C_i$ and every C_i is a disjunction of a subset of the $2n$ possible literals. A formula is said to be in r-CNF if the size of each clause is no larger than r.

Sparsification Lemma *For every $\epsilon > 0$ and positive integer r, there is a constant $C = \mathcal{O}\big((\frac{n}{\epsilon})^{3r}\big)$ so that any r-CNF formula F with n variables, can be expressed as $F = \bigvee_{i=1}^{t} Y_i$, where $t \leq 2^{\epsilon n}$ and each Y_i is an r-CNF formula with every variables appearing in at most C clauses. Moreover, this disjunction can be computed by an algorithm running in time $2^{\epsilon n} n^{\mathcal{O}(1)}$.*

Essentially, the Sparsification Lemma implies that, when performing a polynomial reduction r-SATISFABILITY, for *fixed* r, it suffices to assume the input instance on n variables has $\mathcal{O}(n)$ *clauses*. Impagliazzo and Paturi then conjecture a cornerstone of lower bound asymptotic analysis, the *Exponential Time Hypothesis*, commonly referred to as ETH, and its strong version, known as SETH.

Exponential Time Hypothesis *There is a real number s such that* 3-SATISFABILITY *cannot be solved in $2^{sn}(n+m)^{\mathcal{O}(1)}$ time.*

Strong Exponential Time Hypothesis SATISFABILITY *cannot be solved in $(2-\epsilon)^n(n+m)^{\mathcal{O}(1)}$ time, for any $\epsilon > 0$.*

It is not hard to see that if ETH holds, then $\mathsf{P} \neq \mathsf{NP}$. From the moment they were first claimed, both hypothesis have been successfully applied across the literature. Lokshtanov et al. [2013] survey some of these results. For instance, unless the

Exponential Time Hypothesis is false, there is no algorithm running in $2^{o(n)}$ time for Vertex 3-Coloring, Dominating Set, Independent Set, Vertex Cover, nor Hamiltonian Path; Hamiltonian Cycle in planar graphs cannot be solved in $2^{o(\sqrt{n})}n^{\mathcal{O}(1)}$ time. Let k denote the natural parameter of each of the following problems. In terms of FPT algorithms, the existence of $2^{o(k)}n^{\mathcal{O}(1)}$ was ruled out for Vertex Cover, Feedback Vertex Set, and Longest Path, while no $2^{o(\sqrt{k})}n^{\mathcal{O}(1)}$ time algorithm exists for Vertex Cover on planar graphs. ETH can also be used to give algorithmic lower bound to problems not in FPT. Lokshtanov et al. [2013] neither Dominating Set, Clique, Independent Set, nor their multicolored versions can be solved in $f(k)n^{o(k)}$.

While most of the complexity theory community believes ETH to be true, the same is not true for the Strong Exponential time Hypothesis [Pătraşcu and Williams, 2010]. The implications for SETH, however, as the name suggests, are quite powerful. While ETH is generally used to prove assertions on the exponent of the running times of many algorithms, SETH allows for a *much* finer-grained analysis, at the cost of much more complex reductions and arguments, specially because the hypothesis of the Sparsification Lemma *are not respected by* Satisfability. Lokshtanov et al. [2018] give a series of reductions for many problems parameterized by treewidth. They show that the best known algorithms parameterized by treewidth for Independent Set, Dominating Set, Max Cut, Odd Cycle Transversal, Vertex q-Coloring (for any $q \geq 3$), Partition Into Triangles cannot be improved, unless SETH is false. Recently, Abboud et al. [2019] proved what may surely be considered a breakthrough result: by using a hypothesis on the running time of Satsifability (SETH), they proved that the pseudo-polynomial dynamic programming algorithm given by Bellman [1957] for Subset Sum is optimal.

1.4 Graph Classes

Most problems in graph theory can be tackled with an arbitrary input, that is, there is no particular property that we can exploit; this can happen if the considered application is too broad or little is known about its domain. However, it might be possible to guarantee certain characteristics for the given graph, either due constraints of the application [Pereira and Palsberg, 2005] or due to theoretical interest. Regardless, such guarantees might be strong enough to provide an efficient algorithm to an otherwise NP-hard problem. When constraining our analysis to certain graphs, we refer to the family of all graphs that satisfy the same properties as a *graph class*. A subfamily of a

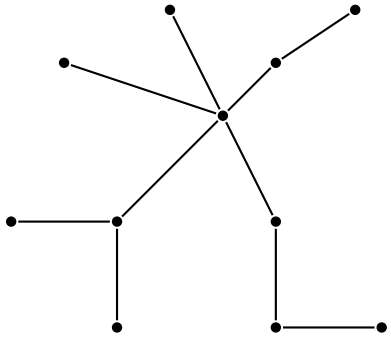

Figure 1: A tree.

class that satisfies additional properties is referred to as a *subclass*. For (much) more on graph classes, Brandstädt et al. [1999] give an extensive survey of much of the work done on the field until the late 1990s.

In this section, we review some of the most studied classes and some of their properties that will aid us in the design of our algorithms.

A graph is a *tree* T if it is a connected acyclic graph or, equivalently, the connected graph such that, between every pair of vertices u and v, there exists a unique path. The vertices of degree one of a tree are called its *leaves*, and all others are *inner* nodes. A *subtree* T' of a tree T is a connected subgraph which, clearly, must also be a tree. A *rooted tree* T_v is a tree with a special vertex v, called its *root*. Rooted trees offer a straight forward ordering of the vertices of a tree and a nice way to decompose problems into smaller instances and combine their solutions. A *rooted subtree* T_u of T_v is the subgraph of T_v induced by u and all vertices of T_v whose path to v passes through u. The vertices in $T_u \setminus \{u\}$ are called the *descendants* of u and its neighbours are its *children*.

A *forest* is a graph where every connected component is a tree. Many problems which are usually quite hard for general graphs, or even some classes, usually have a straightforward answer for forests, either using a greedy strategy or a slightly more sofisticated dynamic programming idea.

Chordal graphs have many nice properties that enable the computation of different graph parameters in polynomial time [Golumbic, 2004]. A *perfect elimination ordering* of a graph G is an ordering $v_1, \ldots, v_n$ of its vertices such that for the graph $G[\{v_i, \ldots, v_n\}]$, v_i is a simplicial vertex. As the name implies, chordal graphs are exactly the graphs where every cycle of size at least 4 has at least one chord.

Chordal Graph *For a graph G, the following properties are equivalent:*

(i) G is chordal;

(ii) It is C_k-free, for any $k \geq 4$;

(iii) Every minimal cutset is a clique;

(iv) It is the intersection graph of subtrees of a tree;

(v) There is a perfect elimination ordering of its vertices.

Figure 2: A chordal graph and its clique tree.

For a chordal graph G, its *clique tree* is a tree $\mathcal{T}(G)$ such that: its vertex set, each of which is called a *bag*, is the set of maximal cliques of G, and, for every vertex v of G, the set of bags which contains v induces a subtree of $\mathcal{T}(G)$. It can be shown that such a tree satisfies property *(iv)*. For more on clique trees and other chordal graph properties please refer to [Blair and Peyton, 1993].

Not surprisingly, many subclasses of chordal graphs have also been studied, since even forests are chordal graphs. A *block graph* is a chordal graph where every minimal cutset is a single vertex. An *interval graph* is the intersection graph of a set of intervals over the real numbers. A *split graph* is a graph whose vertex set can be partitioned into a clique and an independent set.

Cographs are the graphs G such that either G or its complement is disconnected. At first glance, such property may not seem very helpful to the algorithm designer, but it is equivalent to a very nice recursive definition, first given in [Corneil et al., 1981]. Given two graphs G and H, we define their *disjoint union* as the graph $G \cup H$ with $V(G \cup H) = V(G) \cup V(H)$ and $E(G \cup H) = E(G) \cup E(H)$, and their *join* as the graph $G \otimes H$ with vertex set is $V(G \otimes H) = V(G) \cup V(H)$ and edge set $E(G \otimes H) = E(G) \cup E(H) \cup \{uv \mid u \in V(G), v \in V(H)\}$.

Cograph *For a graph G, the following properties are equivalent:*

(i) G is a cograph;

(ii) G is P_4-free;

(iii) G can be constructed from isolated vertices by successively applying disjoint union and join operations.

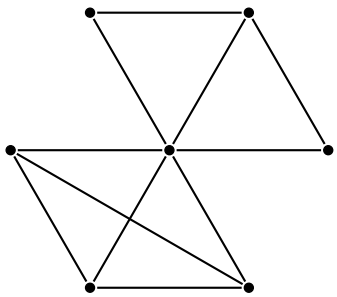

Figure 3: A cograph.

Another important class on graph theory, and one with a very long history of research, is the class of *regular graphs*. A graph G is regular if all of vertices of G have the same degree, and is k-regular if $\deg(v) = k$ for all $v \in V(G)$. Despite its simplicity, regular graphs appear in many different scenarios, such as in the EΔCC conjecture on EQUITABLE COLORING, a conjecture for INTERNAL PARTITIONS, but even more in terms of algebraic graph theory [Godsil and Royle, 2013], a field dedicated to the analysis of many graph parameters through algebraic methods, such as spectral decompositions, graph polynomials, and interlacing. In particular, by using the eigenvalues of the adjacency matrix of a graph, or its Laplacian matrix, it is possible to derive bounds for a large collection of parameters, such as independence number and chromatic number, usually in polynomial time. Regular graphs, in particular, benefit greatly from this approach, with stronger results for this class when compared to other classes.

Chapter 2

Equitable, Clique, and Biclique Coloring

In a coloring problem, the goal is to partition (i.e. color) the vertices (or edges) of a graph such that each set (color class) of the partition, individually, satisfies some condition. For the classical Vertex Coloring problem, the goal is to partition the graph's vertex set such that, inside each member of the partition there are no two adjacent vertices. Multiple additional constraints or properties may be added to the desired partition. By further imposing that the size of each partition member be as close as possible to each other, the Equitable Coloring problem, which appears to be much harder to solve even for graph classes where classical vertex coloring is efficiently solvable, is generated. Another possible modification to Vertex Coloring generates the B-Coloring problem [Campos et al., 2013], where a coloring of the vertices such that each color class has at least one vertex with one neighbor in each of the other classes is sought. Much like Equitable Coloring appears to be considerably harder than Vertex Coloring, List Coloring also exhibits a similar behavior; in this problem each vertex has a list of admissible colors, and the goal is to color the graph respecting these restrictions. List Assignment, however, takes things to a whole different level. It asks if for a given graph, for every possible choice of list with exactly k colors to each vertex of the graph, it is possible to find a list-coloring. In fact, this coloring version is not even NP-complete being Π_2^p-complete even for bipartite graphs [Gutner, 1996]. One may also impose the constraint that no maximal induced subgraph be entirely contained in a single set of the partition. For example, it may be required that no maximal clique, biclique (complete bipartite graph), or star of the given graph may be monochromatic generating the problems known as Clique Coloring, Biclique Coloring, and Star Coloring, respectively.

In this chapter, we present results concerning the EQUITABLE, CLIQUE and BICLIQUE COLORING problems. We first formalize of many of the concepts we use in our proofs, as well as present some related work on each of the problems and a brief discussion on VERTEX COLORING. We then proceed in earnest to our results. For EQUITABLE COLORING, our first results are W[1]-hardness proofs for some subclasses of chordal graphs; namely, for block graphs of bounded diameter when parameterized by treewidth and maximum number of colors, for $K_{1,4}$-free interval graphs when parameterized by treewidth, maximum number of colors and maximum degree, and for disjoint union of split graphs (which are also complete multipartite) when parameterized by treewidth and maximum number of colors. We close the subject of EQUITABLE COLORING with some algorithms. We show that the problem admits an XP algorithm for chordal graphs when a parameterized by the maximum number of colors, a constructive polynomial time algorithm to equitably color claw-free chordal graphs, and an FPT algorithm parameterized by the treewidth of the complement graph. We then turn to CLIQUE COLORING and BICLIQUE COLORING. The first exact exponential time algorithm for biclique coloring, which builds upon ideas used for clique coloring, is presented. Afterwards, we give kernelization algorithms for both problems when parameterized by neighborhood diversity; using results on covering problems, an FPT algorithm under the same parameterization is obtained for CLIQUE COLORING, which has optimal running time, up to the base of the exponent, unless the Exponential Time Hypothesis fails. For BICLIQUE COLORING, an FPT algorithm is given, but when parameterized by maximum number of colors and neighborhood diversity.

2.1 Definitions and Related Work

A *k-coloring* of a graph G is a k-partition $\varphi = \{\varphi_1, \ldots, \varphi_k\}$ of $V(G)$. Each φ_i is a *color class* and $v \in V(G)$ is *colored* with color i if and only if $v \in \varphi_i$. In a slight abuse of notation, we use $\varphi(v)$ to denote the color of v and, for $X \subseteq V(G)$, $\varphi(X) = \bigcup_{v \in X} \{\varphi(v)\}$.

2.1.1 Proper Coloring

A *proper k-coloring* of G is a k-coloring such that each φ_i is an independent set. In the literature, proper coloring is usually referenced to as *Vertex Coloring*, a convention we also adopt. If G has a proper k-coloring we say that G is *k-colorable*. The smallest integer k such that G is k-colorable is called the *chromatic number* $\chi(G)$ of G. The natural decision problem associated with vertex coloring simply asks whether or not a given graph is k-colorable.

Vertex Coloring
Instance: A graph G and a positive integer k.
Question: Is G k-colorable?

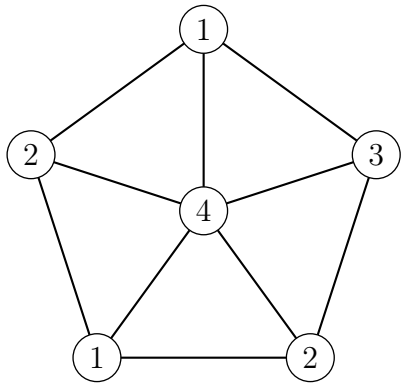

Figure 4: An optimal proper coloring.

Determining if a given instance of Vertex Coloring is a YES instance is a classic problem in both graph theory and algorithmic complexity, being a known NP-complete problem. Some particular cases of Vertex Coloring are still NP-complete. For instance, even if we fix $k = 3$ or restrict the input to K_3-free graphs the problem does not get any easier.

It is worth to point out the subtle difference between the parameter k being part of the input or being *fixed*. Informally, when k is fixed, we are willing to pay exponential time only on k to solve our problem, whereas when k is part of the input, we are not. Note that when we fix k and find an $f(k)n^{\mathcal{O}(1)}$ time algorithm, we show that the problem is in FPT when parameterized by k. The fact that 3-coloring is NP-complete is evidence that Vertex Coloring parameterized by the number of colors is not in FPT, otherwise we would have an $f(3)n^{\mathcal{O}(1)}$ algorithm, which would imply that P = NP.

For an unconstrained input, Vertex Coloring is hard to approximate to a factor of $n^{1-\epsilon}$, for any $\epsilon > 0$, unless some complexity hypothesis fail (see [Feige and Kilian, 1996] for more on the topic). On a brighter note, a celebrated theorem due to Brooks in [Brooks, 1941] gives a nice upper bound for general graphs, and gives a natural direction for research on tighter upper bounds on graph classes.

Theorem (Brooks' Theorem)**.** *For every connected graph G which is neither complete nor an odd-cycle $\chi(G) \leq \Delta(G)$.*

These results motivated much of the research about Vertex Coloring. There are polynomial time algorithms for a myriad of different classes, including chordal, bipartite and cographs. More generally, there are known polynomial time algorithms

for *perfect graphs* [Grötschel et al., 1984], which is a superclass of the aforementioned ones. G is perfect if for every induced subgraph G' of G, $\chi(G') = \omega(G')$.

More particular cases for VERTEX COLORING have also been analyzed. For instance, Karthick et al. [2017] present some results for graph classes that have two connected five-vertex forbidden induced subgraphs. There are some surveys on the subject, as such we point to [Golovach et al., 2017] and [Paulusma, 2016] for more on the classic VERTEX COLORING problem, since it is not the focus of this thesis.

2.1.2 Equitable Coloring

A k-coloring of an n vertex graph is said to be *equitable* if for every color class φ_i, $\lfloor \frac{n}{k} \rfloor \leq |\varphi_i| \leq \lceil \frac{n}{k} \rceil$ or, equivalently, if for, any two color class φ_i and φ_j, $||\varphi_i| - |\varphi_j|| \leq 1$. If G admits a proper equitable k-coloring, we say that G is *equitably k-colorable*. Unlike other coloring variants previously discussed, an equitably k-colorable graph is not necessarily equitably $(k+1)$-colorable.

As such, two different parameters are defined: the smallest integer k such that G is equitably k-colorable is called the *equitable chromatic number* $\chi_=(G)$; the smallest integer k' such that G is equitably k-colorable for every $k \geq k'$ is the *equitable chromatic threshold* $\chi^*_=(G)$ of G.

As with the previous coloring problems, we define the EQUITABLE COLORING decision problem.

EQUITABLE COLORING
Instance: A graph G and a positive integer k.
Question: Is G equitably k-colorable?

Figure 5: A proper non-equitable coloring (left) and an equitable coloring (right).

Equitable Coloring was first discussed by [Meyer, 1973], with an intended application for municipal garbage collection, and later in processor task scheduling [Baker and Coffman, 1996] and server load balancing [Smith et al., 2004].

Much of the work done over Equitable Coloring aims to prove an analogue of Brooks' theorem, known as the *Equitable coloring conjecture* (ECC). In terms of the equitable chromatic threshold, however, we have the *Hajnal-Szemerédi theorem* [Hajnal and Szemerédi, 1970].

Conjecture (ECC)**.** *For every connected graph G which is neither a complete graph nor an odd-hole, $\chi_{=}(G) \leq \Delta(G)$.*

Theorem (Hajnal-Szmerédi Theorem)**.** *Any graph G is equitably k-colorable if $k \geq \Delta(G) + 1$. Equivalently, $\chi^{*}_{=}(G) \leq \Delta(G) + 1$.*

Chen et al. [1994] suggest that a stronger result than the Hajnal-Szmerédi theorem may be achievable, presenting some classes where the *Equitable Δ-coloring conjecture* (EΔCC) holds. Moreover, they prove that if EΔCC holds for every regular graph, then it holds for every graph.

Conjecture (EΔCC)**.** *For every connected graph G which is not a complete graph, an odd-hole nor $K_{2n+1,2n+1}$, for any $n \geq 1$, $\chi^{*}_{=}(G) \leq \Delta(G)$ holds.*

Quite a lot of effort was put into finding classes where EΔCC holds, even with the knowledge that only proofs for regular graphs are required. A result given by de Werra [1985], combined with Brooks' Theorem, implies that every claw-free graph is equitably k-colorable for every $k \geq \chi(G)$. A very extensive survey on the subject was conducted by Lih [2013], where many of the results of the past 50 years were assembled. Among the many reported results, the EΔCC is known to hold for: bipartite graphs (with the obvious exceptions, where the ECC holds), trees, split graphs, planar graphs, *outerplanar graphs* (planar graphs with a drawing such that no vertex is within a polygon formed by other vertices), *low degeneracy graphs* (graphs such that every subgraph has a vertex with a small degree), *Kneser graphs* (complement of the intersection graph of $F \subset 2^{[n]}$, with every set of F containing exactly k elements), interval graphs, random graphs and some forms of graph products. For the exact results please refer to the survey.

Almost all complexity results for Equitable Coloring arise from a related problem, known as Bounded Coloring, an observation given by Bodlaender and Fomin [2004]. A k-coloring is said to be *l-bounded* if for every color class φ_i, $|\varphi_i| \leq l$. G is *l-bounded k-colorable* if it admits an l-bounded k-coloring.

BOUNDED COLORING
Instance: A graph G and two positive integers l and k.
Question: Is G l-bounded k-colorable?

Observation. *A Graph G with n vertices is l-bounded k-colorable if and only if $G' = G \cup \overline{K_{lk-n}}$ is equitably k-colorable.*

In terms of computational complexity, however, neither problem was nearly as explored as VERTEX COLORING. Among the complexity results for BOUNDED COLORING and, consequently, EQUITABLE COLORING, we have polynomial time solvability for split graphs [Chen et al., 1996], complement of interval graphs [Bodlaender and Jansen, 1995], forests [Baker and Coffman, 1996], trees [Jarvis and Zhou, 2001] and complements of bipartite graphs [Bodlaender and Jansen, 1995].

For cographs, we have a polynomial time algorithm when k is fixed, otherwise the problem is NP-complete [Bodlaender and Jansen, 1995], a situation similar to that of bipartite and interval graphs [Bodlaender and Jansen, 1995]. A consequence of the hardness result for cographs is that EQUITABLE COLORING is NP-complete for graphs of bounded cliquewidth.

On complements of *comparability graphs* (graphs representing a valid partial ordering) however, even if we fix l, BOUNDED COLORING is still NP-complete [Lonc, 1992]. Fellows et al. [2011] show that, when parameterized by treewidth, EQUITABLE COLORING is W[1]-hard. Also in terms of treewidth, Bodlaender and Fomin [2004] give a polynomial time algorithm for graphs of bounded treewidth. Note that for all of the mentioned classes, VERTEX COLORING is polynomially solvable.

A summary of the known complexities is available in Table 2.

2.1.3 Clique Coloring

A *k-clique-coloring* of G is a k-coloring of G such that no maximal clique of G is entirely contained in a single color class. We say that G is *k-clique-colorable* if G admits a k-clique-coloring. The smallest integer k such that G is k-clique-colorable is called the *clique chromatic number* $\chi_{\mathcal{C}}(G)$ of G. Much like VERTEX COLORING, there is a natural decision problem associated with this coloring variant, which we refer to as CLIQUE COLORING.

CLIQUE COLORING
Instance: A graph G and a positive integer k.
Question: Is G k-clique-colorable?

Class	fixed k	input k
Trees	P	P
Forests	P	P
Bipartite	NP-complete	NP-complete
Co-bipartite	P	P
Cographs	P	NP-complete
Bounded Cliquewidth	NP-complete	NP-complete
Bounded Treewidth	P	P
Chordal	P*	NP-complete
Block	P*	NP-complete*
Split	P	P
Interval	P	NP-complete
Co-interval	P	P
General case	NP-complete	NP-complete

Table 2: Complexity results for Equitable Coloring. Entries marked with a * are results established in this work.

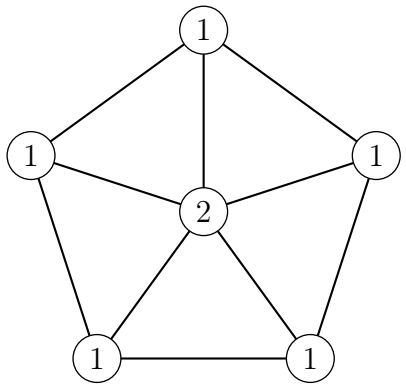

Figure 6: An optimal clique coloring.

Research on the topic is much more recent than what was done for Vertex Coloring, with the first papers appearing in the early 1990s [Duffus et al., 1991] and interest on the subject rising in the early 2000s. Even when k is fixed, Clique Coloring is known to be Σ_2^{P}-complete, as shown by Marx [2011], with an $\mathcal{O}^*(2^n)$ algorithm being proposed by Cochefert and Kratsch [2014].

As with Vertex Coloring, Clique Coloring has been studied when restricting the input graph to certain graph classes. Macêdo Filho et al. [2016] investigate 2-clique-coloring in terms of *weakly chordal graphs* (graphs free of any hole or anti-hole with more than 4 vertices), giving a series of results for the general case (Σ_2^{P}-complete) and showing that, for some nested subclasses, there are NP-complete and P instances of the problem. When dealing with *unichord-free* graphs (graphs that contain no induced cycle with a unique chord), the problem is solvable in polynomial time [Macêdo Filho

et al., 2012].

Circular-arc graphs (intersection graphs of a set of arcs of a circle) are always 3-clique-colorable, with a polynomial time algorithm to determine if the input is 2-clique-colorable [Cerioli and Korenchendler, 2009]. When the given graph is *odd-hole-free*, it is Σ_2^{P}-complete to decide whether it is 2-clique-colorable or not [Défossez, 2009]. Klein and Morgana [2012] give a series of bounds on graphs that, in some sense, contain few P_4's, showing that most them are either 2 or 3-clique-colorable.

For *planar graphs* (graphs that can be drawn on a plane with no crossing edges), Mohar and Skrekovski [1999] show that they are 3-clique-colorable, and Kratochvíl and Tuza [2002] present a polynomial time algorithm to decide whether a planar graph is 2-clique-colorable or not.

Some of these classes are subclasses of perfect graphs, and a conjecture suggests that every perfect graph is 3-clique-colorable [Bacsó et al., 2004]. Also in terms of perfect graphs, it is NP-complete to decide whether a perfect graph is 2-clique-colorable [Kratochvíl and Tuza, 2002]. Défossez [2009] also give the observation that every *strongly perfect graph* [Berge and Duchet, 1984] is 2-clique-colorable, a superclass of both chordal graphs and cographs. For a summary of the mentioned results, please refer to Table 3.

Class	χ_C	Complexity
Cograph	$= 2$	P
Chordal	$= 2$	P
Weakly Chordal	$\leq 3^*$	Σ_2^{P}-complete†
Unichord-free	≤ 3	P
Circular-arc	≤ 3	P†
Odd-hole-free	$\leq 3^*$	Σ_2^{P}-complete†
Few P_4's	≤ 2 or ≤ 3	P
Planar	≤ 3	P†
Perfect	$\leq 3^*$	NP-complete†
Strongly Perfect	$= 2$	P
General case	Unbounded	Σ_2^{P}-complete

Table 3: Complexity and bounds for CLIQUE COLORING. Entries marked with a $*$ are conjectures. † indicates results for 2-clique-colorability.

2.1.4 Biclique Coloring

A *k-biclique-coloring* of G is a k-coloring of G such that no maximal biclique of G is entirely contained in a single color class. We say that G is *k-biclique-colorable* if G

admits a k-biclique-coloring. The smallest integer k such that G is k-biclique-colorable is called the *biclique chromatic number* $\chi_{\mathcal{B}}(G)$ of G. Much like Clique Coloring, there is a natural decision problem associated with this coloring variant, which we refer to as Biclique Coloring.

Biclique Coloring
Instance: A graph G and a positive integer k.
Question: Is G k-biclique-colorable?

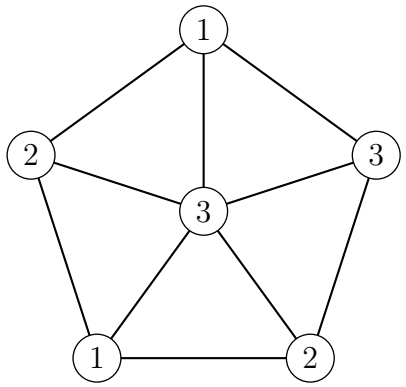

Figure 7: An optimal biclique coloring.

Biclique Coloring is an even more recent research topic than Clique Coloring, with the first results being a Σ_2^{P}-completeness proof due to Groshaus et al. [2014] and the confirmation that verifying a solution to the problem is a coNP-complete task [Macêdo Filho et al., 2015].

In terms of complexity results, very little is known about Biclique Coloring. For unichord-free graphs, Macêdo Filho et al. [2012] give a polynomial time algorithm to compute $\chi_{\mathcal{B}}$ and show that the biclique chromatic number of unichord-free graphs is either equal to or one greater than the size of the largest true twin class. Macêdo Filho et al. [2015] present a polynomial time algorithm for powers of cycles and powers of paths. Finally, Groshaus et al. [2014] give complexity results for H-free graphs, for every H on three vertices, being polynomial for $H \in \{K_3, P_3, \overline{P_3}\}$ and NP-complete for $\overline{K_3}$-free graphs; moreover they show that the problem is NP-complete for diamond (C_4 plus one chord) free graphs and split graphs, and polynomial for *threshold graphs* ($\{2K_2, P_4, C_4\}$-free). We summarize the presented results in Table 4.

Both Clique Coloring and Biclique Coloring are, actually, colorings of the hypergraphs arising from an underlying graph (a coloring of its vertices such that no hyperedge is monochromatic), which is also an NP-complete task. However, in classical hypergraph coloring problems, the hyperedge family is part of the input of the problem and, as such, naively verifying a solution is polynomial on the size of the input.

Class	$\chi_{\mathcal{B}}$	Complexity
Split	Unbounded	NP-complete
Threshold	Unbounded	P
Diamond-free	Unbounded	NP-complete
C_n^r	≤ 3	P
P_n^r	$= 2$	P
Unichord-free	Bounded	P
General case	Unbounded	Σ_2^{P}-complete

Table 4: Complexity and bounds for BICLIQUE COLORING.

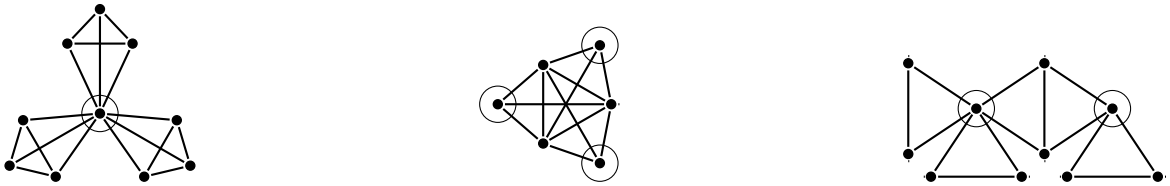

Figure 8: A $(2,4)$-flower, a $(2,4)$-antiflower, and a $(2,2)$-trem.

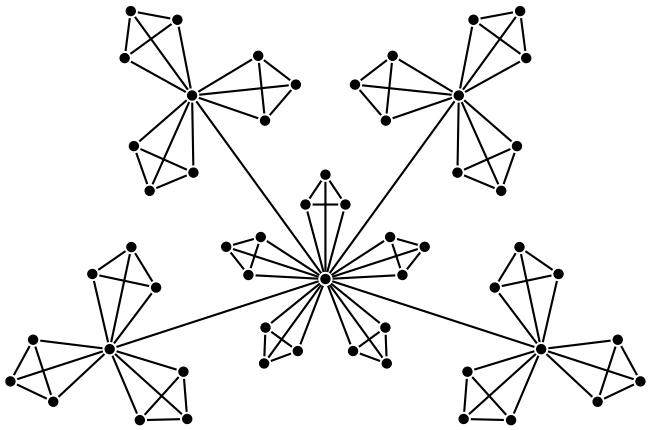

Figure 9: EQUITABLE COLORING instance built on Theorem 4 corresponding to the BIN PACKING instance $A = \{2,2,2,2\}$, $k = 3$ and $B = 4$.

2.2 Hardness of Equitable Coloring for subclasses of Chordal Graphs

All of our reductions involve the BIN PACKING problem, which is NP-hard in the strong sense [Garey and Johnson, 1979] and W[1]-hard when parameterized by the number of bins [Jansen et al., 2013]. In the general case, the problem is defined as: given a set of positive integers $A = \{a_1, \ldots, a_n\}$, called *items*, and two integers k and B, can we partition A into k *bins* such that the sum of the elements of each bin is at most B? We shall use a version of BIN PACKING where each bin sums *exactly* to B. This second version is equivalent to the first, even from the parameterized point of view; it suffices

to add $kB - \sum_{j\in[n]} a_j$ unitary items to A. For simplicity, by Bin Packing we shall refer to the second version, which we formalize as follows.

BIN-PACKING
Instance: A set of n items A and a bin capacity B.
Parameter: The number of bins k.
Question: Is there a k-partition φ of A such that, $\forall i \in [k]$, $\sum_{a_j\in\varphi_i} a_j = B$?

The idea for the following reductions is to build one gadget for each item a_j of the given Bin Packing instance, perform their disjoint union, and equitably k-color the resulting graph. The color given to the circled vertices in Figure 8 control the bin to which the corresponding item belongs to. Each reduction uses only one of the three gadget types. Since every gadget is a chordal graph, their treewidth is precisely the size of the largest clique minus one, that is, k, which is also the number of desired colors for the built instance of equitable coloring.

2.2.1 Disjoint union of Split Graphs

Definition 1. An (a, k)-antiflower is the graph $F_-(a, k) = K_{k-1} \oplus \left(\bigcup_{i\in[a+1]} K_1\right)$, that is, it is the graph obtained after performing the disjoint union of $a + 1$ K_1's followed by the join with K_{k-1}.

Theorem 2. equitable coloring *of the disjoint union of split graphs parameterized by the number of colors is* W[1]-hard.

Proof. Let $\langle A, k, B\rangle$ be an instance of Bin Packing and G a graph such that $G = \bigcup_{j\in[n]} F_-(a_j, k)$. Note that $|V(G)| = \sum_{j\in[n]} |F_-(a_j, k)| = \sum_{j\in[n]} k + a_j = nk + kB$. Therefore, in any equitable k-coloring of G, each color class has $n + B$ vertices. Define $F_j = F_-(a_j, k)$ and let C_j be the corresponding K_{k-1}. We show that there is an equitable k-coloring ψ of G if and only if $\varphi = \langle A, k, B\rangle$ is a YES instance of Bin Packing.

Let φ be a solution to Bin Packing. For each $a_j \in A$, we do $\psi(C_j) = [k] \setminus \{i\}$ if $a_j \in \varphi_i$. We color each vertex of the independent set of F_j with i and note that all remaining possible proper colorings of the gadget use each color the same number of times. Thus, $|\psi_i| = \sum_{j|a_j\in\varphi_i}(a_j + 1) + \sum_{j|a_j\notin\varphi_i} 1 = \sum_{j|a_j\in\varphi_i}(a_j + 1) + \sum_{j\in[n]} 1 - \sum_{j|a_j\in\varphi_i} 1 = n + B$.

Now, let ψ be an equitable k-coloring of G. Note that $|\psi_i| = n + B$ and that the independent set of an antiflower is monochromatic. For each $j \in [n]$, $a_j \in \varphi_i$ if

$i \notin \psi(C_j)$. That is, $n + B = |\psi_i| = \sum_{j|i\notin C_j}(a_j + 1) + \sum_{j|i\in C_j} 1 = \sum_{j|i\notin C_j}(a_j + 1) + \sum_{j\in[n]} 1 - \sum_{j|i\notin C_j} 1 = \sum_{j|i\notin C_j} a_j + n$, from which we conclude that $\sum_{j|i\notin C_j} a_j = B$. □

2.2.2 Block Graphs

We now proceed to the parameterized complexity of block graphs. Conceptually, the proof follows a similar argumentation as the one developed in Theorem 2; in fact, we are able to show that even restricting the problem to graphs of diameter at least four is not enough to develop an FPT algorithm, unless FPT = W[1].

Definition 3. An (a, k)-flower is the graph $F(a, k) = K_1 \oplus \left(\bigcup_{i\in[a+1]} K_{k-1}\right)$, that is, it is obtained from the union of $a + 1$ cliques of size $k - 1$ followed by a join with K_1.

Theorem 4. EQUITABLE COLORING *of block graphs of diameter at least four parameterized by the number of colors and treewidth is* W[1]-hard.

Proof. Let $\langle A, k, B\rangle$ be an instance of BIN PACKING, $\forall k \in [n]$, $F_j = F(a_j, k + 1)$, $F_0 = F(B, k + 1)$ and, for $j \in \{0\} \cup [n]$, let y_j be the universal vertex of F_j. Define a graph G such that $V(G) = V\left(\bigcup_{j\in\{0\}\cup[n]} V(F_j)\right)$ and $E(G) = \{y_0y_j \mid j \in [n]\} \cup E\left(\bigcup_{j\in\{0\}\cup[n]} E(F_j)\right)$. Looking at Figure 9, it is easy to see that any minimum path between a non-universal vertex of F_a and a non-universal vertex of F_b, $a \neq b \neq 0$ has length four. We show that $\langle A, k, B\rangle$ is an YES instance if and only if G is equitably $(k + 1)$-colorable.

$$\begin{aligned} |V(G)| = |V(F_0)| + \sum_{j\in[n]} |V(F_j)| &= k(B + 1) + 1 + \sum_{j\in[n]} (1 + k(a_j + 1)) \\ = kB + k + n + k^2B + kn + 1 &= (k + 1)(kB + n + 1) \end{aligned}$$

Given a k-partition φ of A that solves our instance of BIN PACKING, we construct a coloring ψ of G such that $\psi(y_j) = i$ if $a_j \in \varphi_i$ and $\psi(y_0) = k + 1$. Using a similar argument to the previous theorem, after coloring each y_j, the remaining vertices of G are automatically colored. For ψ_{k+1}, note that $|\psi_{k+1}| = 1+\sum_{j\in[n]}(a_j+1) = kB+n+1 = \frac{|V(G)|}{k+1}$. It remains to prove that every other color class ψ_i also has $\frac{|V(G)|}{k+1}$ vertices.

$$\begin{aligned} |\psi_i| &= B+1+\sum_{j|y_j\notin\psi_i}(a_j+1)+\sum_{j|y_j\in\psi_i}1 & &= B+1+\sum_{j\in[n]}(a_j+1)-\sum_{j|y_j\in\psi_i}a_j \\ &= B+1+kB+n-B & &= kB+n+1 \end{aligned}$$

For the converse we take an equitable $(k+1)$-coloring of G and suppose, without loss of generality, that $\psi(y_0)=k+1$ and, consequently, for every other y_i, $\psi(y_i)\neq k+1$. To build our k-partition φ of A, we say that $a_j\in\varphi_i$ if $\psi(y_j)=i$. The following equalities show that $\sum_{a_j\in\varphi_i}a_j=B$ for every i, completing the proof.

$$\begin{aligned} |\psi_i| &= B+1+\sum_{j|y_j\in\psi_i}1+\sum_{j|y_j\notin\psi_i}(a_j+1) & &= B+1+\sum_{j\in[n]}(a_j+1)-\sum_{j|y_j\in\psi_i}a_j \\ kB+n+1 &= B+1+kB+n-\sum_{j|y_j\in\psi_i}a_j & &\Rightarrow B=\sum_{j|y_j\in\psi_i}a_j \end{aligned}$$

□

2.2.3 Interval Graphs without some induced stars

Definition 5. Let $\mathcal{Q}=\{Q_1,Q'_1,\ldots,Q_a,Q'_a\}$ be a family of cliques such that $Q_i\simeq Q'_i\simeq K_{k-1}$ and $Y=\{y_1,\ldots,y_a\}$ be a set of vertices. An (a,k)-trem is the graph $H(a,k)$ where $V(H(a,k))=\mathcal{Q}\cup Y$ and $E(H(a,k))=E\left(\bigcup_{i\in[a]}(Q_i\cup Q'_i)\oplus y_i\right)\cup E\left(\bigcup_{i\in[a-1]}y_i\oplus Q_{i+1}\right)$.

Theorem 6. Equitable Coloring *of* $K_{1,4}$ *free interval graphs parameterized by treewidth, maximum number of colors and maximum degree is* W[1]-hard.

Proof. Once again, let $\langle A,k,B\rangle$ be an instance of Bin Packing, define $\forall j\in[n]$, $H_j=H(a_j,k)$ and let Y_j be the set of cut-vertices of H_j. The graph G is defined as $G=\bigcup_{j\in[n]}V(H_j)$. By the definition of an (a,k)-trem, we note that the vertices with largest degree are the ones contained in $Y_j\setminus\{y_a\}$, which have degree equal to $3(k-1)$. We show that $\langle A,k,B\rangle$ is an YES instance if and only if G is equitably k-colorable, but first note that $|V(G)|=\sum_{j\in[n]}|V(H_j)|=\sum_{j\in[n]}a_j+2a_j(k-1)=kB+2(k-1)kB=k(2kB-B)$.

Given a k-partition φ of A that solves our instance of Bin Packing, we construct a coloring ψ of G such that, for each $y\in Y_j$, $\psi(y)=i$ if and only if $a_j\in\varphi_i$. Using a

similar argument to the other theorems, after coloring each Y_j, the remaining vertices of G are automatically colored, and we have $|\psi_i| = \sum_{j\in\varphi_i} a_j + \sum_{j\notin\varphi_i} 2a_j = B + 2(k-1)B = 2kB - B$.

For the converse we take an equitable k-coloring of G and observe that, for every $j \in [n]$, $|\psi(Y_j)| = 1$. As such, to build our k-partition φ of A, we say that $a_j \in \varphi_i$ if and only if $\psi(Y_j) = \{i\}$. Thus, since $|\psi| = 2kB - B$, we have that $2kB - B = \sum_{j\in\varphi_i} a_j + \sum_{j\notin\varphi_i} 2a_j = \sum_{j\in[n]} a_j - \sum_{j\in\varphi_i} a_j = 2kB - \sum_{j\in\varphi_i} a_j$, from which we conclude that $B = \sum_{j\in\varphi_i} a_j$. □

2.3 Exact Algorithms for Equitable Coloring

2.3.1 Chordal Graphs

In this section we will make heavy use of the clique tree $\mathcal{T}(G) = (\mathcal{Q}, F)$ of our chordal graph G, which we denote by $\mathcal{T}$ for simplicity. We also assume that $\mathcal{Q} = \{Q_1, \ldots, Q_r\}$, $|V(G)| = n$, that $\mathcal{T}$ is rooted at Q_1 and that T_i is the subtree of $\mathcal{T}$ rooted at bag Q_i.

Our dynamic programming algorithm explores the separability structure inherent to chordal graphs, embodied by the clique tree, to combine every coloring of a subtree that may yield an equitable coloring of the whole graph. To do so, we must keep track of which bag, say Q_i, we are currently exploring and which colors have been used at the separator between Q_i and $\mathcal{T} \setminus Q_i$.

A *k-color counter*, or simply a *counter* for an n vertex graph is an element $X \in \left(\left[\left\lceil\frac{n}{k}\right\rceil\right] \cup \{0\}\right)^k$, that is, the k-th Cartesian power of $\left[\left\lceil\frac{n}{k}\right\rceil\right] \cup \{0\}$. A counter X is *equitable* if for every $x_i, x_j \in X$, $|x_i - x_j| \leq 1$. For simplicity, denote $S(n,k) = \left[\left\lceil\frac{n}{k}\right\rceil\right] \cup \{0\}$. The *sum* of two counters X, Y is defined as $X + Y = (x_1 + y_1, \ldots, x_k + y_k)$. We also define the sum of two families $\mathcal{X}, \mathcal{Y}$ of counters as $\mathcal{Z} = \mathcal{X} + \mathcal{Y} = \left\{X + Y \in S(n,k)^k \mid X \in \mathcal{X}, Y \in \mathcal{Y}\right\}$, that is, the sum of all pairs of elements from each family that belong to $S(n,k)^k$.

Observation 7. *If $\mathcal{X}, \mathcal{Y} \subseteq S(n,k)^k$ and $\mathcal{Z} = \mathcal{X} + \mathcal{Y}$, $|\mathcal{Z}| \leq |S(n,k)|^k$.*

Theorem 8. *There is an $\mathcal{O}\left(n^{2k+2}\right)$ time algorithm for* EQUITABLE COLORING *on chordal graphs where k is the maximum number of colors.*

Proof. Let G be the input chordal graph of order n and $\mathcal{T}$ its clique tree, rooted at bag Q_1. Moreover, we assume that $k \geq \omega(G)$, otherwise the answer is trivially NO.

For a bag Q, denote by $p(Q)$ the parent clique of Q on $\mathcal{T}$, $I(Q) = Q \cap p(Q)$ and by $U(Q) = Q \setminus p(Q)$ the set of vertices that first appeared on the path between Q and the root Q_1 (if $Q = Q_1$, $U(Q) = Q$).

Given Q and a list of available colors L, define $\Pi(Q, L)$ as the set of all colorings of $U(Q)$ using only the colors of L, $\beta(Q, L, \pi)$ the list of colors used by L and π to color $I(Q)$ and $Y(Q, \pi)$ as the counter where $y_i = 1$ if and only if some vertex of $U(Q)$ was colored with color i in π.

We define our dynamic programming state $f(Q_i, L)$ as all colorings of T_i conditioned to the fact that $I(Q_i)$ was colored with L. Intuitively, we will try every possible coloring π of $U(Q)$ and combine the solutions of each bag adjacent to Q_i given the colors used by π and L. Finally, there will be an equitable k-coloring of G if there is an equitable counter in $f(Q_1, \emptyset)$.

$$f(Q_i, L) = \bigcup_{\pi \in \Pi(Q_i, [k] \setminus L)} \left(Y(Q_i, \pi) + \sum_{Q_j \in N_{T_i}(Q_i)} f(Q_j, \beta(Q_j, L, \pi)) \right)$$

To prove the correctness of our algorithm, we will use induction on the size of $\mathcal{T}$. For the base case, where $|V(\mathcal{T})| = 1$, trivially, any proper coloring of G will be equitable.

For the general case, suppose that Q_i has at least one child, say Q_j. Inductively, for any list of colors R, $f(Q_j, R)$ holds every proper coloring of $T_j \setminus I(Q_j)$, given that $I(Q_j)$ was colored with R. In particular, for every $\pi \in \Pi(Q_i, [k] \setminus L)$ we have this guarantee. Note that, for each pair of children Q_j, Q_l of Q_i, their solutions are completely independent because $Q_j \cap Q_l \subseteq Q_i$ (clique tree property) and Q_i is entirely colored by π and L. This implies that no vertex was counted more than once on each counter, since their color is chosen exactly once for each possible π. Therefore, since the problems are independent for each child of Q_i, $\mathcal{Z}(\pi) = \sum_{Q_j \in N_{T_i}(Q_i)} f(Q_j, \beta(Q_j, L, \pi))$ combines every possible coloring of the children of Q_i and $\mathcal{Z}(\pi) + Y(Q_i, \pi)$ is the family of all colorings of T_i, given that Q_i was colored with π and L. Finally, $\bigcup_{\pi \in \Pi(Q_i, [k] \setminus L)} Y(Q, \pi) + Z(\pi)$ tries every possible coloring π of Q_i, guaranteeing that $f(Q_i, L)$ has every possible coloring of T_i, given that $I(Q_i)$ used L. Since $f(Q_i, L)$ has every coloring of $\mathcal{T}_\rangle$ given L, G will be equitably k-colorable if and only if there is an equitable counter at $f(Q_1, \emptyset)$.

In terms of complexity analysis, we first note that each sum of two counter families $\mathcal{X}, \mathcal{Y}$ takes $\mathcal{O}(|\mathcal{X}||\mathcal{Y}|)$. However, due to Observation 7, the size of $\mathcal{X} + \mathcal{Y}$ is at most $|S(N, k)|^k$ and, therefore, $\sum_{Q_j \in N_{T_i}(Q_i)} f(Q_j, \beta(Q_j, L, \pi))$ takes at most $\mathcal{O}\big(n|S(n,k)|^{2k}\big) = \mathcal{O}\Big(n \left(\lceil \frac{n}{k} \rceil + 1\right)^{2k}\Big)$ time; moreover, the addition of $Y(Q_i, \pi)$ to $\mathcal{Z}(\pi)$, by the same argument, is $\mathcal{O}\big(|S(n,k)|^k\big)$.

For the outermost union, we have $\mathcal{O}(k!)$ possible colorings π for $U(Q_i)$, which implies that computing each $f(Q_i, L)$ takes $\mathcal{O}\Big(k!n \left(\lceil \frac{n}{k} \rceil + 1\right)^{2k}\Big)$ time. Since we have $r \leq n$ bags and $k!$ possible lists, there are $\mathcal{O}(nk!)$ states, therefore the total complexity

of our dynamic programming algorithm is $\mathcal{O}\Big(k!^2 n^2 \left(\lceil \frac{n}{k} \rceil + 1\right)^{2k}\Big) = \mathcal{O}\big(n^{2k+2}\big)$. □

Corollary 9. EQUITABLE COLORING *of chordal graphs parameterized by the number of colors is in* XP.

As shown by Theorem 6, EQUITABLE COLORING of $K_{1,4}$-free interval graphs is W[1]-hardwhen parameterized by treewidth, maximum number of colors and maximum degree. The construction of the hard instance, however, has a massive amount of copies of the claw ($K_{1,3}$); determining the complexity of the problem for the class of claw-free chordal graphs is, therefore, a direct question. Before answering it, we require a bit more of notation. Given a partial k-coloring φ of G, let $G[\varphi]$ denote the subgraph of G induced by the vertices colored with φ, define φ_- as the set of colors used $\lfloor |V(G[\varphi])|/k \rfloor$ times in φ and φ_+ the remaining colors. If k divides $|V(G[\varphi])|$, we say that $\varphi_+ = \emptyset$. Our goal is to color G one maximal clique (say Q) at a time and keep the invariant that the new vertices introduced by Q can be colored with a subset of the elements of L_-. To do so, we rely on the fact that, for claw-free graphs, the maximal connected components of the subgraph induced by any two colors form either cycles, which cannot happen since G is chordal, or paths. By carefully choosing which colors to look at, we find odd length paths that can be greedily recolored to restore our invariant.

Lemma 10. *There is an $\mathcal{O}(n^2)$-time algorithm to equitably k-color a claw-free chordal graph or determine that no such coloring exists.*

Proof. We proceed by induction on the number n of vertices of G, and show that G is equitably k-colorable if and only if its maximum clique has size at most k. The case $n = 1$ is trivial. For general n, take one of the leaves of the clique tree of G, say Q, a simplicial vertex $v \in Q$ and define $G' = G \setminus \{v\}$. By the inductive hypothesis, there is an equitable k-coloring of G' if and only if $k \geq \omega(G')$. If $k < \omega(G')$ or $k < |Q|$, G can't be properly colored.

Now, since $k \geq \omega(G) \geq |Q|$, take an equitable k-coloring φ' of G' and define $Q' = Q \setminus \{v\}$. If $|\varphi'_- \setminus \varphi'(Q')| \geq 1$, we can extend φ' to φ using one of the colors of $\varphi'_- \setminus \varphi'(Q')$ to greedily color v. Otherwise, note that $\varphi'_+ \setminus \varphi'(Q') \neq \emptyset$ because $k \geq \omega(G')$. Now, take some color $c \in \varphi'_- \cap \varphi'(Q')$, $d \in \varphi'_+ \setminus \varphi'(Q')$; by our previous observation, we know that $G'[\varphi_c \cup \varphi_d]$ has $C = \{C_1, \ldots, C_l\}$ connected components, which in turn are paths. Now, take $C_i \in C$ such that C_i has odd length and both endvertices are colored with d; said component must exist since $d \in \varphi'_+$ and $c \in \varphi'_-$. Moreover, $C_i \cap Q' = \emptyset$, we can swap the colors of each vertex of C_i and then color v with d; neither operation makes an edge monochromatic.

As to the complexity of the algorithm, at each step we may need to select c and d – which takes $\mathcal{O}(k)$ time – construct C, find C_i and perform its color swap, all of which take $\mathcal{O}(n)$ time. Since we need to color n vertices and $k \leq n$, our total complexity is $\mathcal{O}(n^2)$. □

The above algorithm was not the first to solve Equitable Coloring for claw-free graphs; this was accomplished by de Werra [1985] which implies that, for any claw-free graph G, $\chi_=(G) = \chi^*_=(G) = \chi(G)$.

Theorem 11 (de Werra [1985])**.** *If G is claw-free and k-colorable, then G is equitably k-colorable.*

However, [de Werra, 1985] is not easily accessible, as it is not in any online repository. Moreover, the given algorithm has no clear time complexity and, as far as we were able to understand the proof, its running time would be $\mathcal{O}(k^2 n)$, which, for $k = f(n)$, is worse than the algorithm we present in Lemma 10.

2.3.2 Clique Partitioning

Since Equitable Coloring is W[1]-hard when simultaneously parameterized by many parameters, we are led to investigate a related problem. Much like Equitable Coloring is the problem of partitioning G in k' independent sets of size $\lceil n/k \rceil$ and $k - k'$ independent sets of size $\lfloor n/k \rfloor$, one can also attempt to partition $\overline{G}$ in cliques of size $\lceil n/k \rceil$ or $\lfloor n/k \rfloor$. A more general version of this problem is formalized as follows:

clique partitioning
Instance: A graph G and two positive integers k and r.
Question: Can G be partitioned in k cliques of size r and $\frac{n-rk}{r-1}$ cliques of size $r-1$?

We note that both maximum matching (when $k \geq n/2$) and triangle packing (when $k < n/2$) are particular instances of clique partitioning, the latter being FPT when parameterized by k [Fellows et al., 2005]. As such, we will only be concerned when $r \geq 3$. To the best of our efforts, we were unable to provide an FPT algorithm for clique partitioning when parameterized by k and r, even if we fix $r = 3$. However, the situation is different when parameterized by the treewidth of G, and we obtain an algorithm running in $2^{\mathsf{tw}(G)} n^{\mathcal{O}(1)}$ time for the corresponding counting problem, #clique partitioning.

The key ideas for our bottom-up dynamic programming algorithm are quite straightforward. First, cliques are formed only when building the tables for forget

nodes. Second, for join nodes, we can safely consider only the combination of two partial solutions that have empty intersection on the covered vertices (i.e. that have already been assigned to some clique). Finally, both join and forget nodes can be computed using fast subset convolution [Björklund et al., 2007]. For each node $x \in \mathbb{T}$, our algorithm builds the table $f_x(S, k')$, where each entry is indexed by a subset $S \subseteq B_x$ that indicates which vertices of B_x have already been covered, an integer k' recording how many cliques of size r have been used, and stores how many partitions exist in G_x such that only $B_x \setminus S$ is yet uncovered. If an entry is inconsistent (e.g. $S \nsubseteq B_x$), we say that $f(S, K') = 0$.

Theorem 12. *There is an algorithm that, given a nice tree decomposition of an n-vertex graph G of width tw, computes the number of partitions of G in k cliques of size r and $\frac{n-rk}{r-1}$ cliques of size $r-1$ in time $\mathcal{O}^*(2^{\mathsf{tw}})$ time.*

Proof. Leaf node: Take a leaf node $x \in \mathbb{T}$ with $B_x = \emptyset$. Since the only one way of covering an empty graph is with zero cliques, we compute f_x with:

$$f_x(S, k') = \begin{cases} 1, \text{ if } k' = 0 \text{ and } S = \emptyset; \\ 0, \text{ otherwise.} \end{cases}$$

Introduce node: Let x be a an introduce node, y its child and $v \in B_x \setminus B_y$. Due to our strategy, introduce nodes are trivial to solve; it suffices to define $f_x(S, k') = f_y(S, k')$. If $v \in S$, we simply define $f_x(S, k') = 0$.

Forget node: For a forget node x with child y and forgotten vertex v, we formulate the computation of $f_x(S, k')$ as the subset convolution of two functions as follows:

$$f_x(S, k') = f_y(S \cup \{v\}, k') + \sum_{A \subseteq S} f_y(S \setminus A, k' - 1) g_r(A, v) + \sum_{A \subseteq S} f_y(S \setminus A, k') g_{r-1}(A, v)$$

$$g_l(A, v) = \begin{cases} 1, \text{ if } A \text{ is a clique of size } l \text{ contained in } N[v] \text{ and } v \in A; \\ 0, \text{otherwise.} \end{cases}$$

The above computes, for every $S \subseteq B_x$ and every clique A (that contains v) of size r or $r-1$ contained in $N[v] \cap B_y \cap S$, if $S \setminus A$ and some k'' is a valid entry of f_y, or if v had been previously covered by another clique (first term of the sum). Directly computing the last two terms of the equation, for each pair (S, k'), yields a total running time of the order of $\sum_{|S|=0}^{\mathsf{tw}} \binom{\mathsf{tw}}{|S|} 2^{|S|} = (1+2)^{\mathsf{tw}} = 3^{\mathsf{tw}}$ for each forget

node. However, using the fast subset convolution technique described by Björklund et al. [2007], we can compute the above equation in time $\mathcal{O}^*\left(2^{|B_x|}\right) = \mathcal{O}^*(2^{\mathsf{tw}})$.

Correctness follows directly from the hypothesis that f_y is correctly computed and that, for every $A \subseteq B_x$, $g_r(A, v)g_{r-1}(A, v) = 0$. For the running time, we can pre-compute both g_r and g_{r-1} in $\mathcal{O}(2^{\mathsf{tw}}r^2)$, so their values can be queried in $\mathcal{O}(1)$ time. As such, each forget node takes $\mathcal{O}(2^{\mathsf{tw}}\mathsf{tw}^3k)$ time, since we can compute the subset convolutions of $f_y * g_r$ and $f_y * g_{r-1}$ in $\mathcal{O}(2^{\mathsf{tw}}\mathsf{tw}^3)$ time each. The additional factor of k comes from the second coordinate of the table index.

Join node: Take a join node x with children y and z. Since we want to partition our vertices, the cliques we use in G_y and G_z must be completely disjoint and, consequently, the vertices of B_x covered in B_y and B_z must also be disjoint. As such, we can compute f_x through the equation:

$$f_x(S, k') = \sum_{k_y + k_z = k'} \sum_{A \subseteq S} f_y(A, k_y) f_z(S \setminus A, k_z)$$

Note that we must sum over the integer solutions of the equation $k_y + k_z = k'$ since we do not know how the cliques of size r are distributed in G_x. To do that, we compute the subset convolution $f_y(\cdot, k_y) * f_z(\cdot, k_z)$. The time complexity of $\mathcal{O}(2^{\mathsf{tw}}\mathsf{tw}^3k^2)$ follows directly from the complexity of the fast subset convolution algorithm, the range of the outermost sum and the range of the second parameter of the table index.

For the root x, we have $f_x(\emptyset, k) \neq 0$ if and only if $G_x = G$ can be partitioned in k cliques of size r and the remaining vertices in cliques of size $r - 1$. Since our tree decomposition has $\mathcal{O}(n\mathsf{tw})$ nodes, our algorithm runs in time $\mathcal{O}(2^{\mathsf{tw}}\mathsf{tw}^4k^2n)$.

To recover a solution given the tables f_x, start at the root node with $S = \emptyset$, $k' = k$ and let $\mathcal{Q} = \emptyset$ be the cliques in the solution. We shall recursively extend $\mathcal{Q}$ in a top-down manner, keeping track of the current node x, the set of vertices S and the number k' of K_r's used to cover G_x. Our goal is to keep the invariant that $f_x(S, k') \neq 0$.

Introduce node: Due to the hypothesis that $f_x(S, k') \neq 0$ and the way that f_x is computed, it follows that $f_y(S, k') \neq 0$.

Forget node: Since the current entry is non-zero, there must be some $A \subseteq S$ such that exactly one of the products $f_y(S \setminus A, k' - 1)g_r(A, v)$, $f_y(S \setminus A, k')g_{r-1}(A, v)$ is non-zero and, in fact, any such A suffices. To find this subset, we can iterate through 2^S in $\mathcal{O}(2^{\mathsf{tw}})$ time and test both products to see if any of them is non-zero. Note that the chosen $A \cup \{v\}$ will be a clique of size either r or $r - 1$, and thus, we can set $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{A \cup \{v\}\}$.

Join node: The reasoning for join nodes is similar to forget nodes, however, we only need to determine which states to look at in the child nodes. That is, for each

integer solution to $k_y+k_z = k'$ and for each $A \subseteq S$, we check if both $f_y(A, k_y)f_z(S\backslash A, k_z)$ is non-zero; in the affirmative, we compute the solution for both children with the respective entries. Any such triple (A, k_y, k_z) that satisfies the condition suffices.

Clearly, retrieving the solution takes $\mathcal{O}(2^{\mathsf{tw}}k)$ time per node, yielding a running time of $\mathcal{O}^*(2^{\mathsf{tw}})$. □

Corollary 13. *Equitable coloring is* FPT *when parameterized by the treewidth of the complement graph.*

2.4 Clique and Biclique Coloring

Both CLIQUE COLORING and BICLIQUE COLORING are relaxations of the classical VERTEX COLORING problem, in the sense that monochromatic edges are allowed. However, this freedom comes at the cost of validating a solution, which becomes a coNP-complete task in both cases. One may think of VERTEX COLORING as the task of covering a graph's vertices using a given number of independent sets. That is, there cannot be a color class with an edge inside it. For CLIQUE COLORING and BICLIQUE COLORING, the idea is quite similar. We want to forbid not edges, but maximal clique or bicliques, respectively, inside our color classes. All of the following results establish families of sets that may safely be used to cover the given graph and describe how to compute them.

Much of the following discussion will deal with the clique and biclique hypergraphs $\mathcal{H}_{\mathcal{C}}(G)$ and $\mathcal{H}_{\mathcal{B}}(G)$. As such, we denote by $\mathbb{T}_C(G)$ ($\mathbb{T}_B(G)$) the *family of all transversals* of the clique (biclique) hypergraph of G and by $\mathbb{T}^*_C(G)$ ($\mathbb{T}^*_B(G)$) the *family of complements of transversals*. Also, denote by $\mathbb{O}_C(G)$ ($\mathbb{O}_B(G)$) the *family of all obliques* of the clique (biclique) hypergraph of G. Finally, $\mathcal{C}(G)$ ($\mathcal{B}(G)$) is the family of *maximal cliques* (bicliques) of G.

In this chapter, we present algorithms that make heavy use of the algorithm described by Björklund et al. [2009], which applies the inclusion-exclusion principle to solve a variety of problems in $2^n n^{\mathcal{O}(1)}$ time, including VERTEX COLORING. Our main results are an $\mathcal{O}^*(2^n)$ algorithm for BICLIQUE COLORING, an FPT algorithm for CLIQUE COLORING parameterized by neighbourhood diversity and an FPT algorithm for BICLIQUE COLORING parameterized by the number of colors and neighbourhood diversity. To achieve them, we will rely on the following problems and results of the literature.

Lemma 14 (Cochefert and Kratsch [2014])**.** *For any family $\mathcal{F}$, its down closure $\mathcal{F}_\downarrow = \{X \subseteq V \mid \exists Y \in \mathcal{F},\ X \subseteq Y\}$ can be enumerated in $O^*(|\mathcal{F}_\downarrow|)$ time.*

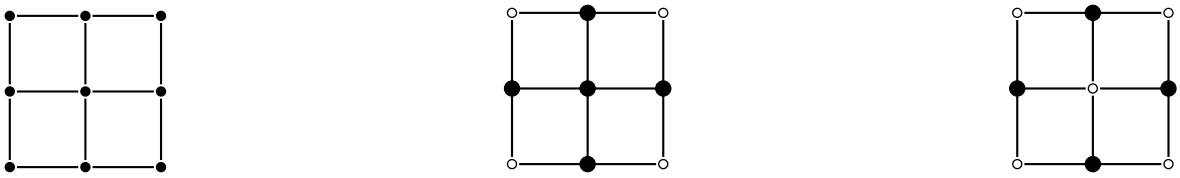

Figure 10: From left to right: a graph, one of its maximal bicliques, and a transversal.

Lemma 15 (Cochefert and Kratsch [2014])**.** *A k-partition $\varphi = \{\varphi_1, \ldots, \varphi_k\}$ is a k-clique-coloring of G if and only if for every i, $\overline{\varphi_i} \in \mathbb{T}_C(G)$.*

EXACT COVER
Instance: A set $A = \{a_1, \ldots, a_n\}$, a covering family $\mathcal{F} \subseteq 2^A$ and an integer k.
Question: Is it possible to k-partition A into φ such that $\varphi \subseteq \mathcal{F}$?

Theorem 16 (Björklund et al. [2009])**.** *There is a $\mathcal{O}^*(2^n)$ time algorithm to solve* EXACT COVER.

Theorem 17 (Cochefert and Kratsch [2014])**.** *There is an $\mathcal{O}^*(2^n)$ time algorithm for* Clique Coloring.

SET MULTICOVER
Instance: A set $A = \{a_1, \ldots, a_n\}$, a covering family $\mathcal{F} \subseteq 2^A$, an integer k and a coverage demand $c : A \mapsto \mathbb{N}$.
Question: Is it possible to k-cover A with $\varphi \subseteq \mathcal{F}$ and $\forall a_j, |\{i \mid a_j \in \varphi_i\}| \geq c(a_j)$?

Theorem 18 (Hua et al. [2009])**.** *Set multicover can be solved in $O^*((b+2)^n)$, with b the maximum coverage requirement.*

2.5 Exact Algorithm for Biclique Coloring

Drawing inspiration from Cochefert and Kratsch [2014], we first show the relationships between hypergraph structures and colorings, and use these to build an $\mathcal{O}^*(2^n)$-time algorithm for Biclique Coloring by stating it as an exact cover instance. Naturally, the covering family must be carefully chosen such that any solution to the covering problem produces a valid coloring. We first formalize the observation that, given a color i, every maximal biclique of G must have one color other than i.

Lemma 19. *A k-partition $\varphi = \{\varphi_1, \ldots, \varphi_k\}$ is a k-biclique-coloring of G if and only if for every i, $\overline{\varphi_i} \in \mathbb{T}_B(G)$.*

Proof. Suppose that there exists some φ_i such that $\overline{\varphi_i} \notin \mathbb{T}_B(G)$. This implies that there exists some $B \in \mathcal{B}(G)$ such that $B \cap \overline{\varphi_i} = \emptyset$ and that $B \subseteq \varphi_i$; that is, $|\varphi(B)| = 1$, which is a contradiction, since φ is a k-biclique-coloring.

For the converse, let φ be a k-partition of G with $\overline{\varphi_i} \in \mathbb{T}_B(G)$, but suppose that φ_k is not a k-biclique-coloring. That is, there exists some maximal biclique $B \in \mathcal{B}(G)$ such that $B \subseteq \varphi_i$ for some i. This implies that $B \cap \overline{\varphi_i} = \emptyset$, and, therefore, $\overline{\varphi_i} \notin \mathbb{T}_B(G)$, which contradicts the hypothesis. □

Simply testing for each $X \in 2^{V(G)}$ if $X \in \mathbb{T}^*_B(G)$ is a costly task. A naive algorithm would check, for each $B \in \mathcal{B}(G)$, if $\overline{X} \cap B \neq \emptyset$. With $|\mathcal{B}(G)| \in \mathcal{O}\big(n3^{\frac{n}{3}}\big)$ (see Gaspers [2010] for the proof), such algorithm would take $\mathcal{O}\big(n2^n 3^{\frac{n}{3}}\big)$-time. The next Lemma, along with Lemma 14, considerably reduces the complexity of enumerating $\mathbb{T}_B(G)$. We will enumerate $\mathbb{O}_B(G)$ by generating its maximal elements and then use the fact that $\mathbb{O}_B(G)$ is closed under the subset operation.

Lemma 20. *The maximal obliques of $\mathcal{H}_{\mathcal{B}}(G)$ are exactly the complements of the maximal bicliques of G.*

Proof. Let $X \in \mathbb{O}_B$ be a maximal oblique. By definition, there exists some $B \in \mathcal{B}(G)$ such that $X \cap B = \emptyset$, which implies that $X \subseteq \overline{B}$. Note that, if $X \subset \overline{B}$, there is some $v \in \overline{B} \setminus X$, which implies that $(X \cup \{v\}) \cap B = \emptyset$ and that X is not a maximal oblique. Let $B \in \mathcal{B}(G)$. By definition, $\overline{B} \in \mathbb{O}_B$ and must be maximal because $\{B, \overline{B}\}$ is a partition of $V(G)$. □

Corollary 21. *Given a graph $G = (V, E)$ and a subset $X \subseteq V(G)$, there exists an $O(n(n - |X|))$-time algorithm to determine if X is a maximal oblique.*

Theorem 22. *There is an $\mathcal{O}^*(2^n)$-time algorithm for* BICLIQUE COLORING.

Proof. Our goal is to make use of Theorem 16 to solve an instance of EXACT COVER, with $A = V(G)$, $\mathcal{F} = \mathbb{T}^*_B(G)$ and k the partition size. Lemma 19 guarantees that there is an answer to our instance of BICLIQUE COLORING if and only if there is an answer to the corresponding EXACT COVER one. To compute $\mathbb{T}^*_B(G)$, for each $X \in 2^{V(G)}$, we use Lemma 20 and Corollary 21 to say whether or not X is a maximal oblique of $\mathcal{H}_{\mathcal{B}}(G)$. Next, we compute $\mathbb{O}_B(G)$ from its maximal elements using Lemma 14, and use the fact that $\mathbb{T}_B(G) = 2^{V(G)} \setminus \mathbb{O}_B(G)$ and complement each transversal to obtain $\mathbb{T}^*_B(G)$. Clearly, this procedure takes $\mathcal{O}^*(2^n)$-time to construct $\mathbb{T}^*_B(G)$ and an additional $\mathcal{O}^*(2^n)$-time by Theorem 16. □

2.6 Algorithms Parameterized by Neighborhood Diversity

As previously discussed, a type is a maximal set of vertices that are either true or false twins to each other. Suppose that we are already given a partition $\{D_1, \ldots, D_{\mathsf{nd}(G)}\}$ of $V(G)$ in types. If D_i is composed of true twins, we say that it is a *true twin class* T_i and, by definition, $G[D_i]$ is a clique. Similarly, if D_i is composed of false twins, it is a *false twin class* F_i and $G[D_i]$ is an independent set. When $|D_i| = 1$, we treat the class differently depending on the problem. For the entirety of this section, we assume that $d = \mathsf{nd}(G)$.

2.6.1 Biclique Coloring

For Biclique Coloring, if there is some D_i with a single vertex we shall treat it as a true twin class.

Observation 23. *Given G and a true twin class T_i of G, any k-biclique-coloring φ of G has $|\varphi(T_i)| = |T_i|$.*

Lemma 24. *Given G and a false twin class $F \subset V(G)$, any k-biclique-coloring φ' of G can be changed into a k-biclique-coloring φ of G such that $|\varphi(F)| \leq 2$.*

Proof. If $|\varphi'(F)| \leq 2$, $\varphi = \varphi'$. Otherwise, there exists $f_1, f_2, f_3 \in F$ with three different colors. Since every maximal biclique B of G that intercepts F has that $F \subset B$ and thus $|\varphi'(B)| \geq 3$. By making $\varphi(f_1) = \varphi'(f_1)$ and $\varphi(f_3) = \varphi(f_2) = \varphi'(f_2)$, we obtain $|\varphi(B)| \geq |\varphi(F)| \geq 2$. Repeating this process until $|\varphi(F)| = 2$ does not make any biclique monochromatic and completes the proof. □

The central idea of our parameterized algorithm is to build an induced subgraph H of G and, afterward, use the results established here and in Section 2.5 to show that the solution to a particular instance of Set Multicover derived from H can be transformed in a solution to Biclique Coloring of G.

Definition 25 (B-Projection and B-Lifting)**.** Let T_i and F_j be as previously discussed. We define the following projection rules: $\forall t_i^q \in T_i$, $\mathsf{Proj}_B(t_i^q) = \{t_i'\}$; for $f_j^1 \in F_j$, $\mathsf{Proj}_B(f_j^1) = \{f'^1_j\}$; $\forall f_j^r \in F_j \setminus \{f_j^1\}$, $\mathsf{Proj}_B(f_j^r) = \{f'^2_j\}$ and $\mathsf{Proj}_B(X) = \bigcup_{u \in X} \mathsf{Proj}_B(u)$.

Lifting rules are defined as $\mathsf{Lift}_B(t_i') = \{t_i\}$; $\mathsf{Lift}_B(f'^1_j) = \{f_j^1\}$; $\mathsf{Lift}_B(f'^2_j) = F_j \setminus \{f_j^1\}$ and $\mathsf{Lift}_B(Y) = \bigcup_{u \in Y} \mathsf{Lift}_B(u)$. Note that $\mathsf{Proj}_B(\mathsf{Lift}_B(X)) = X$, $\forall X$.

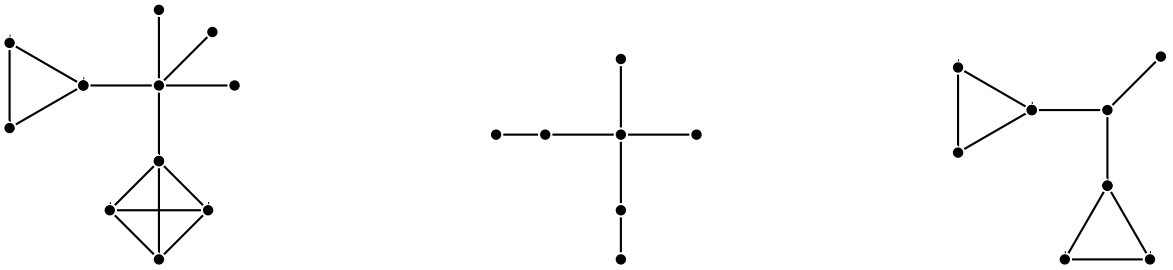

Figure 11: A graph, its B-projected and C-projected graphs

Definition 26 (B-Projected Graph)**.** The B-projected graph H of G satisfies $V(H) = \mathsf{Proj}_B(V(G))$ and $v'_iv'_j \in E(H)$ if and only if there exist $v_i \in \mathsf{Lift}_B(v'_i)$ and $v_j \in \mathsf{Lift}_B(v'_j)$ such that $v_iv_j \in E(G)$. H is an induced subgraph of G.

For the remainder of this section, G will be the input graph to BICLIQUE COLORING and H the B-Projected graph of G. Our SET MULTICOVER instance consists of $V(H)$ as the ground set, $\mathbb{T}^*_B(H)$ as the covering family, the size k of the cover the same as the coloring of G and $c(t'_i) = |T_i|$ for every true twin class T_i and $c(f'^1_j) = c(f'^2_j) = 1$ for each false twin class F_j. The next observation follows directly from the fact that $\mathbb{T}^*_B(H)$ is closed under the subset operation, while the subsequent results allow us to move freely between $\mathbb{T}^*_B(G)$ and $\mathbb{T}^*_B(H)$.

Observation 27. *If there is a minimum k-multicover ψ of $V(H)$ by $\mathbb{T}^*_B(H)$, then there exists a minimum k-multicover $\psi' = \{\psi_1, \ldots, \psi_k\}$ such that $|\{j \mid u \in \psi_j\}| = c(u)$ for every $u \in V(H)$.*

Lemma 28. *If $B' \in \mathcal{B}(H)$ then $B = \mathsf{Lift}_B(B') \in \mathcal{B}(G)$. Conversely, if $B \in \mathcal{B}(G)$ and B is not contained in any true twin class, then $B' = \mathsf{Proj}_B(B) \in \mathcal{B}(H)$.*

Proof. Note that B is a biclique by the definition of Lift_B and the fact that B' is a biclique. By the contrapositive, suppose that $B \notin \mathcal{B}(G)$ and that $u \in V(G)$ is such that $B \cup \{u\}$ is a (not necessarily maximal) biclique of G. Note that either: (i) if $u \in F_j$ then $F_j \nsubseteq B$ and $\mathsf{Proj}_B(u) \notin B'$, because $u \notin \mathsf{Lift}_B(f'^1_j)$ or $u \notin \mathsf{Lift}_B(f'^2_j)$; or (ii) if $u \in T_i$ then $T_i \cap B = \emptyset$, which implies that $\mathsf{Proj}_B(u) \notin B'$. Since $B \cup \{u\}$ is a biclique, $\mathsf{Proj}_B(u)$ is adjacent to only one partition of B'. The fact that $\mathsf{Proj}_B(u) \notin B'$ implies that $\mathsf{Proj}_B(B \cup \{u\}) = \mathsf{Proj}_B(B) \cup \mathsf{Proj}_B(u) = B' \cup \mathsf{Proj}_B(u)$ is a biclique of H and B' is not maximal.

Conversely, by the definition of Proj_B, $B' = (X, Y)$ must be a biclique of H. By the contrapositive, there is $u' \in V(H)$ such that $B' \cup \{u'\}$ is a (not necessarily maximal) biclique of H, and let $u \in \mathsf{Lift}_B(u')$. By the definition of Lift_B, it follows that u can only be adjacent to one of partition of B, say $\mathsf{Lift}_B(X)$. Thus, $u' \in Y$ and, for each

$v \in \mathsf{Lift}_B(Y)$, $uv \notin E(G)$, otherwise there would be $v' \in \mathsf{Proj}_B(v)$ with $u'v' \in E(H)$. Hence, $B \cup \{u\}$ is a biclique of G and B is not maximal. □

Theorem 29. *$X \subseteq V(H)$ is in $\mathbb{T}^*_B(H)$ if and only if $\mathsf{Lift}_B(X) \in \mathbb{T}^*_B(G)$.*

Proof. Recall that $X \in \mathbb{T}^*_B(H)$ if and only if no maximal biclique of H is contained in X. It is clear that, for every $B' \in \mathcal{B}(H)$, $B' \nsubseteq X$ implies that $\mathsf{Lift}_B(B') \nsubseteq \mathsf{Lift}_B(X)$, since no two vertices of H are lifted to the same vertex of G, and $\mathsf{Lift}_B(B') \in \mathcal{B}(G)$ due to Lemma 28. Moreover, no biclique of G entirely contained in a true twin class can be a subset of $\mathsf{Lift}_B(X)$. As such, $\mathsf{Lift}_B(X)$ contains a maximal biclique B only if $\mathsf{Proj}_B(B) \subseteq X$ and $\mathsf{Proj}_B(B) \notin \mathcal{B}(H)$, which is impossible due to Lemma 28 and the assumption that B is maximal.

Taking the contrapositive, $X \notin \mathbb{T}^*_B(H)$ implies that there is some maximal biclique B' of H such that $B' \subseteq X$. This implies that $\mathsf{Lift}_B(B') \subseteq \mathsf{Lift}_B(X)$, and, since $\mathsf{Lift}_B(B')$ is a maximal biclique of G due to Lemma 28, it holds that $\mathsf{Lift}_B(X)$ is not a complement of transversal of G. □

Theorem 30. *ψ is a k-multicover of H if and only if G is k-biclique-colorable.*

Proof. Recall that a k-partition is a k-biclique-coloring if and only if all elements of the partition belong to $\mathbb{T}^*_B(G)$. By the construction of our set multicover instance, we have that, for each ψ_i, $\psi_i \in \mathbb{T}^*_B(H)$. By making $\varphi = \{\mathsf{Lift}_B(\psi_1), \ldots, \mathsf{Lift}_B(\psi_k)\}$, and recalling Observation 27, we have that each vertex $u \in V(H)$ is covered exactly $c(u)$ times; moreover, since true twins appear multiple times and types are equivalence relations, we can attribute to each t^q_i any of the $|T_i|$ colors available, as long as no two receive the same color. Therefore, φ, after properly allocating the true twin classes, is a k-partition of $V(G)$. Due to Theorem 29, every $\mathsf{Lift}_B(\psi_i)$ is a complement of transversal and therefore φ is a valid k-biclique-coloring of G.

For the converse, we first make use of Lemma 24 to guarantee that every false twin class is in at most two color classes. In particular, if two colors are required we force f^1_j to have the smallest color and $F_j \setminus \{f^1_j\}$ to have the other one. Afterwards, for every color class φ_i, we take $\psi_i = \mathsf{Proj}_B(\varphi_i)$. Note that, each color class has at most one element of each T_i. Also, for each F_j and any two distinct color classes φ_l, φ_r, $\mathsf{Proj}_B(\varphi_l) \cap \mathsf{Proj}_B(\varphi_r) \cap \mathsf{Proj}_B(F_j) = \emptyset$, since f^1_j has a different color from $F_j \setminus \{f^1_j\}$. These observations guarantee that $\mathsf{Lift}_B(\psi_i) = \varphi_i$ and, because of Theorem 29, $\psi_i \in \mathbb{T}^*_B(H)$. Finally, $\psi = \{\psi_1, \ldots, \psi_k\}$ will be a valid k-multicover of H because every vertex of $V(H)$ will be covered the required amount of times. □

Note that the size of the largest true twin class is exactly the largest coverage requirement b of our SET MULTICOVER instance. Moreover, since we need at least b colors to biclique color G, it holds that $b \leq k$.

Theorem 31. BICLIQUE COLORING *can be solved in* $\mathcal{O}^*\big((k+2)^{2d}\big)$.

Proof. Start by computing the type partition of G in $\mathcal{O}(n^3)$ time and building H in $O(n+m)$. Afterwards, solve the corresponding SET MULTICOVER instance in $\mathcal{O}^*\big((b+2)^{2d}\big)$ time using Theorem 18 and lift the multicover using the construction described in the proof of Theorem 30 in $\mathcal{O}(n)$. □

Another option would be not to contract true twin classes, keeping all such vertices in the projected graph, which would effectively yield a kernel linear on the product kd. The brute force approach would yield a running time of $\mathcal{O}^*\big(k^{kd}3^{kd/3}\big)$: we could verify if one of the k^{kd} possible colorings is a proper k-biclique-coloring by checking if none of the $\mathcal{O}\big(3^{kd/3}\big)$ maximal bicliques is monochromatic. We could refine our algorithm and use Theorem 22 to solve the problem in $\mathcal{O}^*\big(2^{kd}\big)$, which is no better than $(k+2)^{2d} \approx k^{2d} = 2^{2d\log k}$.

2.6.2 Clique Coloring

For CLIQUE COLORING, a type class with a single vertex is treated as a false twin class. Unlike BICLIQUE COLORING, both true and false twin classes are well behaved, one of the reasons we get a much better algorithm for this problem.

Lemma 32. *Given G and a false twin class $F \subset V(G)$, any k-clique-coloring φ' can be changed into a k-clique-coloring φ such that $|\varphi(F)| = 1$.*

Proof. If $|\varphi'(F)| = 1$, we are done. Otherwise, there exists $f_1, f_2 \in F$ such that $\varphi'(f_1) \neq \varphi'(f_2)$. For every maximal clique C_1 where $f_1 \in C_1$, define $C' = C \setminus \{f_1\}$ and note that $C_2 = C' \cup \{f_2\}$ is also a maximal clique. Since φ' is an coloring $|\varphi'(C') \cup \{\varphi'(f_1)\}| \geq 2$. Therefore, making $\varphi(f_2) = \varphi(f_1) = \varphi'(f_1)$ does not make $|\varphi(C_2)| = 1$. Repeating this until $|\varphi(F)| = 1$ does not make any clique that intercepts F monochromatic and completes the proof. □

Lemma 33. *Given G and a true twin class $T \subseteq V(G)$, any k-clique-coloring φ' can be changed into a k-clique-coloring φ such that $|\varphi(T)| \leq 2$.*

Proof. If $|\varphi'(T)| \leq 2$, we are done. Otherwise, there exists $t_1, t_2, t_3 \in T$ with different colors. Note that, for every maximal clique C that intercepts T, $C \subseteq T$. Therefore,

$|\varphi'(C)| \geq |\varphi'(T)| \geq 3$. By making $\varphi(t_1) = \varphi'(t_1)$ and $\varphi(t_3) = \varphi(t_2) = \varphi'(t_2)$ we have $|\varphi(C)| \geq |\varphi(T)| \geq 2$. Repeating this process until $|\varphi(T)| \leq 2$ does not make any clique that intercepts T monochromatic and the proof follows. □

Definition 34 (C-Projection and C-Lifting)**.** Let T_i be any true twin class and F_j be any false twin class. We define the following projection rules: for $t_i^1 \in T_i$, $\mathsf{Proj}_C(t_i^1) = \{t'^1_i\}$, $\forall t_i^q \in T_i \setminus \{t_i^1\}$, $\mathsf{Proj}_C(t_i^q) = \{t'^2_i\}$, $\forall f_j^r \in F_j$, $\mathsf{Proj}_C(f_j^r) = \{f_j'\}$ and $\mathsf{Proj}_C(X) = \bigcup_{u \in X} \mathsf{Proj}_C(u)$.

Lifting rules are defined as $\mathsf{Lift}_C(t'^1_i) = \{t_i^1\}$, $\mathsf{Lift}_C(t'^2_i) = T_i \setminus \{t_i^1\}$, $\mathsf{Lift}_C(f_j') = \{f_j^1\}$ and $\mathsf{Lift}_C(Y) = \bigcup_{u' \in Y} \mathsf{Lift}_C(u')$. Note that $\mathsf{Proj}_C(\mathsf{Lift}_C(X)) = X$, $\forall X$.

Definition 35 (C-Projected Graph)**.** The C-projected graph H of G satisfies $V(H) = \mathsf{Proj}_C(V(G))$ and $v_i'v_j' \in E(H)$ if and only if there exist $v_i \in \mathsf{Lift}_C(v_i')$ and $v_j \in \mathsf{Lift}_C(v_j')$ such that $v_iv_j \in E(G)$. H is an induced subgraph of G.

For the remainder of this section, G will be the input graph to Clique Coloring and H the C-Projected graph of G. We show, using Lemma 36 and Theorem 37, that Clique Coloring parameterized by neighborhood diversity has a linear kernel. Note that our results imply that $\chi_{\mathcal{C}}(G) \leq 2d$. A straightforward brute force approach would yield an $\mathcal{O}^*\big(4^d d^{2d} 3^{2d/3}\big)$-time algorithm: for each of the $k^{2d} \leq (2d)^{2d}$ possible colorings of H, we check if none of the $\mathcal{O}\big(3^{2d/3}\big)$ maximal cliques of H are monochromatic, returning YES if none are, otherwise NO. Instead, we use Theorem 17 and obtain an $\mathcal{O}^*\big(2^{2d}\big)$-time algorithm.

Lemma 36. *If $C' \in \mathcal{C}(H)$ then $\mathsf{Lift}_C(C') \in \mathcal{C}(G)$. Conversely, if $C \in \mathcal{C}(G)$ then $\mathsf{Proj}_C(C) \in \mathcal{C}(H)$.*

Proof. Note that $C = \mathsf{Lift}_C(C')$ is a clique due to the definition of Lift_C and the fact that C' is a clique. By the contrapositive, suppose that C is not a maximal clique. In this case, there is some vertex $u \in V(G)$ such that $C \cup \{u\}$ is a (not necessarily maximal) clique of G. Note that either: (i) if $u \in T_i$, $T_i \nsubseteq C$ and $u \notin \mathsf{Lift}_C(t'^1_i)$ or $u \notin \mathsf{Lift}_C(t'^2_i)$, thus $\mathsf{Proj}_C(u) \notin C'$; (ii) if $u \in F_j$, $F_j \cap C = \emptyset$ and $\mathsf{Proj}_C(u) \notin C'$. Since no two vertices of H are lifted to the same vertex of G and $\mathsf{Proj}_C(u) \notin C'$, it follows that $\mathsf{Proj}_C(C \cup \{u\}) = \mathsf{Proj}_C(C) \cup \mathsf{Proj}_C(u) = C' \cup \mathsf{Proj}_C(u)$ is a clique by the definition of Proj_C.

Clearly, $C' = \mathsf{Proj}_C(C)$ is a clique of H, due to the definition of Proj_C. Suppose, however, that $C' \notin \mathcal{C}(H)$, which implies that there is some $u' \in V(H)$ such that $C' \cup \{u'\}$ is a clique of H and let $u \in \mathsf{Lift}_C(u')$. By the definition of Lift_C, $C \subseteq N(u)$ if and only if $\mathsf{Proj}_C(C) \subseteq N(u')$, which implies that C' is not maximal only if C is not maximal. A contradiction that completes the proof. □

Theorem 37. *G is k-clique-colorable if and only if H is k-clique-colorable.*

Proof. Let φ_G be a k-clique-coloring of G that complies with Lemmas 32 and 33. Without loss of generality, for every T_i, we color t_i^1 with one color and $T_i \setminus \{t_i^1\}$ with the other, if it exists, otherwise color every vertex of T_i with the same color. We define the k-clique-coloring of H as $\varphi_H(u') = \varphi_G(u \in \mathsf{Lift}_C(u'))$, for every $u' \in V(H)$. Suppose now that there exists some $C' \in \mathcal{C}(H)$ such that $|\varphi_H(C')| = 1$. By Lemma 36, $\mathsf{Lift}_C(C')$ is a maximal clique of G and, since $|\varphi_G(\mathsf{Lift}_C(C'))| = 1$, it holds that $\mathsf{Lift}_C(C')$ is a monochromatic maximal clique of G and φ_G is not a valid k-clique-coloring, which contradicts the hypothesis.

Now, let φ_H be a k-clique-coloring of H, and define $\varphi_G(u) = \varphi_H(u' \in \mathsf{Proj}_C(u))$. By assuming that there exists some $C \in \mathcal{C}(G)$ such that $|\varphi_G(C)| = 1$ and using Lemma 36, it is clear that $|\varphi_H(\mathsf{Proj}_C(C))| = 1$ which is impossible, since φ_H is a valid k-clique-coloring of H. □

Theorem 38. *There is an $\mathcal{O}^*\left(2^{2d}\right)$-time algorithm for* CLIQUE COLORING.

Proof. Start by computing the optimal type partition of G in $\mathcal{O}(n^3)$-time and building H in $\mathcal{O}(n+m)$. Then color H in $\mathcal{O}^*\left(2^{2d}\right)$-time by Theorem 17 and lift the coloring using the construction described in Theorem 37 in $\mathcal{O}(n)$. □

2.6.3 A lower bound under ETH

We now proceed to show that the algorithm described in Theorem 38 is optimal, up to a constant in the exponent, under the assumption that ETH holds. Before proceeding, we recall the canonical problem associated with the Σ_2^{P} class.

2-QUANTIFIED SATISFABILITY (QSAT_2)
Instance: An $n_1 + n_2$ variable 3DNF formula $\varphi(\boldsymbol{x}, \boldsymbol{y})$, on $\boldsymbol{x}$ and $\boldsymbol{y}$.
Question: Is there $\boldsymbol{x} \in \{0,1\}^{n_1}$ such that for every $\boldsymbol{y} \in \{0,1\}^{n_2}$, $\varphi(\boldsymbol{x}, \boldsymbol{y}) = 1$?

Lemma 39. *There is no $\mathcal{O}^*\left(2^{o(n_1+n_2)}\right)$ algorithm for an instance of QSAT_2 on n_1+n_2 variables if ETH holds.*

Proof. By the counter-positive, suppose that there is an algorithm $\prod$ for QSAT_2 with complexity $\mathcal{O}^*\left(2^{o(n_1+n_2)}\right)$ and let $\langle \boldsymbol{x}, \boldsymbol{y}, \varphi(\boldsymbol{x}, \boldsymbol{y})\rangle$ be an instance of QSAT_2 as in the definition of QSAT_2. With $\prod$ in hand, we can solve $\neg\left(\exists \boldsymbol{x} \forall \boldsymbol{y} \varphi(\boldsymbol{x}, \boldsymbol{y})\right) \equiv \forall \boldsymbol{x} \exists \boldsymbol{y} \neg\varphi(\boldsymbol{x}, \boldsymbol{y})$ simply by negating the output of $\prod$. Note that, since $\varphi(\boldsymbol{x}, \boldsymbol{y})$ is in 3DNF, $\neg\varphi(\boldsymbol{x}, \boldsymbol{y})$ is in 3CNF. The case where $n_1 = 0$ is precisely 3SAT, and we have an algorithm that solves it in $\mathcal{O}^*\left(2^{o(n_2)}\right)$, implying that ETH is false. □

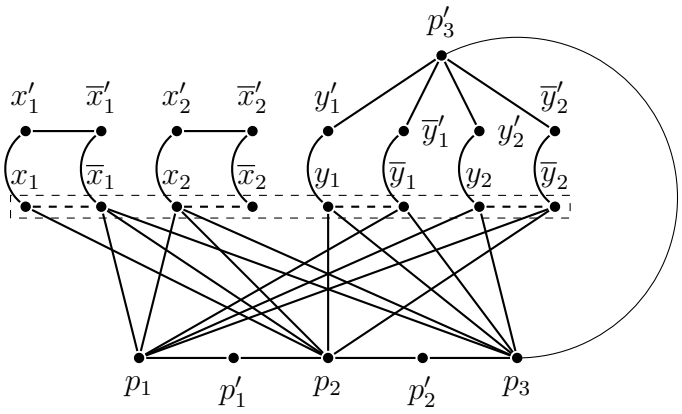

Figure 12: Construction for the formula $\varphi(\boldsymbol{x}, \boldsymbol{y}) = (\overline{x}_1 \wedge x_2 \wedge \overline{y}_1) \vee (x_2 \wedge y_1 \wedge \overline{y}_2) \vee (\overline{x}_1 \wedge x_2 \wedge y_2)$.

Theorem 40. *If ETH holds, there is no $\mathcal{O}^*\big(2^{o(d)}\big)$ time algorithm for* clique 2-coloring *parameterized by the neighborhood diversity d of the graph.*

Proof. Let $\Phi = \langle \boldsymbol{x}, \boldsymbol{y}, \varphi(\boldsymbol{x}, \boldsymbol{y}) \rangle$ be an instance of QSAT_2 as in the problem's definition. We construct the graph G for Clique bicoloring as follows: For each $x_i \in \boldsymbol{x}$, G has 4 vertices $x_i, x'_i, \overline{x}'_i, \overline{x}_i$ and the edges $x_i x'_i, x'_i \overline{x}'_i, \overline{x}'_i \overline{x}_i$. For each $y_j \in \boldsymbol{y}$, G also has 4 vertices $y_j, y'_j, \overline{y}'_j, \overline{y}_j$ but only the edges $y_j y'_j, \overline{y}_j \overline{y}'_j$. Vertices $x_i, \overline{x}_i, y_j, \overline{y}_j$ form a clique minus the edges between a literal and its negation. For each clause $p_l \in \varphi(\boldsymbol{x}, \boldsymbol{y})$, add two vertices p_l, p'_l to G and an edge between p_l and x_i ($\overline{x}_i$) if x_i ($\overline{x}_i$) is in clause p_l. If neither x_i nor $\overline{x}_i$ are in clause p_l, connect p_l to both x_i and $\overline{x}_i$. The same is done between p_l and each y_j. Vertex p'_m is adjacent to every y'_j and every $\overline{y}'_j$; furthermore, $p_1 p'_1 \dots p_m p'_m$ is an induced path of G. By Marx [2011], G is a YES instance if and only if Φ is also a YES instance. For an example of the constructed graph, please refer to Figure 12. We now show that $\mathsf{nd}(G)$ is linearly bounded by the size of Φ.

Define $\eta = \{x_1, \overline{x}_1, \dots, \overline{x}_{n_1}, y_1, \overline{y}_1, \dots, \overline{y}_{n_2}\}$ and $P = \bigcup_{l \leq m} \{p_l, p'_l\}$, $\eta' = \{x'_1, \overline{x}'_1, \dots, \overline{x}'_{n_1}, y'_1, \overline{y}'_1, \dots, \overline{y}'_{n_2}\}$, $P = \{p_l \mid l \leq m\}$ and $P' = \{p'_l \mid l \leq m\}$ For any $\{a, b\} \subseteq \eta \cup \eta'$, it is straightforward to verify that $N(a) \setminus N(b)$, $N[a] \setminus N[b]$, $N(b) \setminus N(a)$ and $N[b] \setminus N[a]$ are non-empty, which implies that a and b are neither false nor true twins. For any $a \in \eta'$ and any $b \in P \cup P'$, it is easy to see that a and b cannot be of the same type. If $a \in \eta$ and $b \in P$, since $\varphi(\boldsymbol{x}, \boldsymbol{y})$ is in 3DNF, there is at least one variable not adjacent to b which is adjacent to a, since η induces a clique minus a matching and, consequently, a and b are not of the same type. If $a \in \eta \cup P$ and $b \in P'$ or $\{a, b\} \subseteq P'$, it is trivial to verify that a and b are neither true nor false twins. For $\{a, b\} \subseteq P$, since no two clauses are equal, it follows that a and b are not of the same type.

As such, we conclude that each vertex of G is in a different type and, consequently,

it has $d = \mathsf{nd}(G) = 4(n_1 + n_2) + 2m$ which is $\mathcal{O}^*(n_1 + n_2 + m)$ and implies that there is no $\mathcal{O}^*\left(2^{o(d)}\right)$ time algorithm for CLIQUE 2-COLORING parameterized by neighborhood diversity unless ETH fails. □

2.7 Concluding Remarks

In this chapter, we investigated three partitioning problems that belong to the class of coloring problems. Namely, EQUITABLE COLORING, CLIQUE COLORING, BICLIQUE COLORING. For EQUITABLE COLORING, we developed novel parameterized reductions from BIN PACKING, which is W[1]-hard when parameterized by number of bins. These reductions showed that EQUITABLE COLORING is $\mathsf{W}[1] - \mathsf{hard}$ in three more cases: (i) if we restrict the problem to block graphs and parameterize by the number of colors, treewidth and diameter; (ii) on the disjoint union of split graphs, a case where the connected case is polynomial; (iii) EQUITABLE COLORING of $K_{1,r}$ interval graphs, for any $r \geq 4$, remains hard even if we parameterize by the number of colors, treewidth and maximum degree. This, along with a previous result by de Werra [1985], establishes a dichotomy based on the size of the largest induced star: for $K_{1,r}$-free graphs, the problem is solvable in polynomial time if $r \leq 2$, otherwise it is $\mathsf{W}[1] - \mathsf{hard}$. These results significantly improve the ones by Fellows et al. [2011] through much simpler proofs and in very restricted graph classes. Since the problem remains hard even for many natural parameterizations, we resorted to a more exotic one – the treewidth of the complement graph. By applying standard dynamic programming techniques on tree decompositions and the fast subset convolution machinery of Björklund et al. [2007], we obtain an FPT algorithm when parameterized by the treewidth of the complement graph. We also presented an XP algorithm parameterized by number of colors when the input graph is known to be chordal. Natural future research directions include the identification and study of other uncommon parameters that may aid in the design of other FPT algorithms. Revisiting CLIQUE PARTITIONING when parameterized by k and r is also of interest, since its a related problem to EQUITABLE COLORING and the complexity of its natural parameterization is yet unknown.

As to the other problems, we showed that, much like CLIQUE COLORING, BICLIQUE COLORING can be solved in $\mathcal{O}^*(2^n)$-time using the inclusion-exclusion principle. Also of interest is the nice behavior CLIQUE COLORING presents when parameterized by neighborhood diversity, which enabled us to apply very simple reduction rules and obtain an $\mathcal{O}^*\left(2^{2d}\right)$-time FPTalgorithm. Moreover, said algorithm has optimal running time, assuming that ETH holds. For BICLIQUE COLORING, however, we were

unable to provide an FPT algorithm when considering solely neighborhood diversity and had to include the size of the largest true twin class – which is a lower bound to the biclique chromatic number – to obtain a parameterized algorithm. As such, we are led to believe that Biclique Coloring parameterized by neighborhood diversity is not in FPT. Much of the exploratory work on different graph classes and parameters remains to be done for Biclique Coloring, and it may be an interesting venue for future work.

Chapter 3

Finding Cuts of Bounded Degree

Unlike coloring problems, cut problems enforce properties *between* sets. The most famous cut problem is, most likely, MINIMUM CUT (or MIN CUT), where the objective is to find a subset of edges of minimum size whose removal disconnects a specified pair of vertices. A celebrated theorem, known as the *Max-Flow Min-Cut theorem* [FORD and FULKERSON, 1962], a special case of strong linear programming duality [Bertsimas and Tsitsiklis, 1998], states that the capacity of a minimum s, t is precisely the value of the maximum flow between s and t; when all edges have unit capacity, we simply seek a cut of minimum cardinality. Other cases of cut problems in the literature include, for instance, MAXIMUM CUT, MAX CUT for short. As the name implies, the goal is to find an s, t cut of the graph with the maximum number of edges between them; however, unlike MIN CUT, MAX CUT is NP-hard, in fact, the value of the cannot be arbitrarily approximated by a polynomial time approximation scheme unless P = NP, i.e. it is APX-hard. Naturally, one can ask whether or not there is a set of edges whose removal causes the disconnection of some vertices, but *not others*. Some research has been dedicated to this topic under the name of MULTICUT-UNCUT, especially in terms of parameterized complexity [Marx et al., 2010; Marx, 2006], with powerful meta-theorems guaranteeing fixed parameter tractability for any class whose membership is hereditary and decidable. Aside from constraints on which vertices are separated or the size of the separation, some cutting problems impose restrictions on the "shape" of the edges crossing the cut. Such is the case of MATCHING CUT and the generalizations we discuss further on.

After some additional definitions and related work, we generalize several results for the d-CUT problem, which, to the best of our knowledge, we are the firsts to formally describe and investigate. First, by using a reduction inspired by Chvátal's [Chvátal, 1984], we show that for every $d \geq 1$, d-CUT is NP-hard even when restricted to $(2d+2)$-

regular graphs and that, if $\Delta(G) \leq d+2$, finding a d-cut can be done in polynomial time. The degree bound in the NP-hardness result is unlikely to be improved: if we had an NP-hardness result for d-CUT restricted to $(2d+1)$-regular graphs, this would disprove the conjecture about the existence of internal partitions on r-regular graphs [DeVos, 2009; Ban and Linial, 2016; Shafique and Dutton, 2002] for r odd, unless every problem in NP could be solved in *constant* time. Afterwards, we present a simple exact exponential algorithm that, for every $d \geq 1$, runs in time $\mathcal{O}^*(c_d^n)$ for some constant $c_d < 2$, hence improving over the trivial brute-force algorithm running in time $\mathcal{O}^*(2^n)$.

We then proceed to analyze the problem in terms of its parameterized complexity. Section 3.3 begins with a proof, using the treewidth reduction technique of Marx et al. [2010], that d-CUT is FPT parameterized by the maximum number of edges crossing the cut. Afterwards, we present a dynamic programming algorithm for d-CUT parameterized by treewidth running in time $\mathcal{O}^*\big(2^{\mathsf{tw}(G)+1}(d+1)^{2\mathsf{tw}(G)+2}\big)$; in particular, for $d=1$ this algorithm runs in time $\mathcal{O}^*\big(8^{\mathsf{tw}(G)}\big)$ and improves the one given by Aravind et al. [2017] for MATCHING CUT, running in time $\mathcal{O}^*\big(12^{\mathsf{tw}(G)}\big)$. By employing the cross-composition framework of Bodlaender et al. [2011], and using a reduction similar to the one given by Komusiewicz et al. [2018], we show that, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, there is no polynomial kernel for d-CUT parameterized simultaneously by the number of crossing edges, the maximum degree, and the treewidth of the input graph. We then present a polynomial kernel and an FPT algorithm when parameterizing by the distance to cluster. This polynomial kernel is our main technical contribution, and it is strongly inspired by the technique presented by Komusiewicz et al. [2018] for MATCHING CUT. Finally, we give an FPT algorithm parameterized by the distance to co-cluster, denoted by $\mathsf{d\overline{c}}(G)$. These results imply fixed-parameter tractability for d-CUT parameterized by $\tau(G)$. Finally, we conclude with an exact exponential algorithm for another generalization of MATCHING CUT and a brief discussion on a third related problem of interest.

3.1 Definitions and Related Work

A *cut* of a graph $G=(V,E)$ is a bipartition of its vertex set $V(G)$ into two non-empty sets, denoted by (A,B). The set of all edges with one endpoint in A and the other in B is the *edge cut*, or the set of *crossing edges*, of (A,B). A *matching cut* is a (possibly empty) edge cut that is a matching, that is, such that its edges are pairwise vertex-disjoint. Equivalently, (A,B) is a matching cut of G if and only if every vertex is

incident to at most one crossing edge of (A, B) [Graham, 1970; Chvátal, 1984], that is, it has at most one neighbor across the cut. The Matching Cut problem is, thus, the task of deciding whether a graph admits a matching cut. Figure 13 gives an example of a graph with a matching cut.

Matching Cut
Instance: A graph G.
Question: Does G have a matching cut?

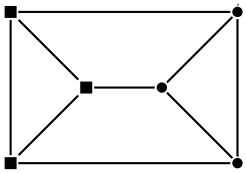

Figure 13: Example of a matching cut. Square vertices would be assigned to A, circles to B.

Motivated by an open question posed by Komusiewicz et al. [2018] during the presentation of their article, we investigate a natural generalization that arises from this alternative definition For a positive integer $d \geq 1$, a *d-cut* is a a cut (A, B) such that each vertex has at most d neighbors across the partition, that is, every vertex in A has at most d neighbors in B, and vice-versa. Note that a 1-cut is a matching cut. As expected, not every graph admits a d-cut, and the d-Cut problem is the problem of, for a fixed integer $d \geq 1$, deciding whether or not an input graph G has a d-cut.

d-Cut
Instance: A graph G.
Question: Does G admit a d-cut?

When $d = 1$, the problem is known as Matching Cut. Graphs with no matching cut first appeared in Graham's manuscript [Graham, 1970] under the name of *indecomposable graphs*, presenting some examples and properties of decomposable and indecomposable graphs, leaving their recognition as an open problem. In answer to Graham's question, Chvátal [1984] proved that the problem is NP-hard for graphs of maximum degree at least four and polynomially solvable for graphs of maximum degree at most three; in fact, as shown by Moshi [1989], every graph of maximum degree three and at least eight vertices has a matching cut.

Chvátal's results spurred a lot of research on the complexity of the problem [Komusiewicz et al., 2018; Aravind et al., 2017; Kratsch and Le, 2016; Le and Le, 2016; Bonsma, 2009; Patrignani and Pizzonia, 2001; Le and Randerath, 2003]. In particular,

Bonsma [2009] showed that MATCHING CUT remains NP-hard for planar graphs of maximum degree four and for planar graphs of girth five; Le and Randerath [2003] gave an NP-hardness reduction for bipartite graphs of maximum degree four; Le and Le [2016] proved that MATCHING CUT is NP-hard for graphs of diameter at least three, and presented a polynomial-time algorithm for graphs of diameter at most two. Beyond planar graphs, Bonsma [2009] also proves that the matching cut property is expressible in monadic second order logic and, by Courcelle's Theorem [Courcelle, 1990], it follows that MATCHING CUT is FPT when parameterized by the treewidth of the input graph; he concludes with a proof that the problem admits a polynomial-time algorithm for graphs of bounded cliquewidth.

Kratsch and Le [2016] noted that Chvátal's original reduction also shows that, unless the Exponential Time Hypothesis fails, there is no algorithm solving MATCHING CUT in time $2^{o(n)}$ on n-vertex input graphs. Also in [Kratsch and Le, 2016], the authors provide a first branching algorithm, running in time $\mathcal{O}^*\left(2^{n/2}\right)$, a single-exponential FPT algorithm when parameterized by the vertex cover number $\tau(G)$, and an algorithm generalizing the polynomial cases of line graphs [Moshi, 1989] and claw-free graphs [Bonsma, 2009]. Kratsch and Le [2016] also asked for the existence a single-exponential algorithm parameterized by treewidth. In response, Aravind et al. [2017] provided a $\mathcal{O}^*\left(12^{\mathsf{tw}(G)}\right)$ algorithm for MATCHING CUT using nice tree decompositions, along with FPT algorithms for other structural parameters, namely neighborhood diversity, twin-cover, and distance to split graph.

The natural parameter – the number of edges crossing the cut – has also been considered. Indeed, Marx et al. [2010] tackled the STABLE CUTSET problem, to which MATCHING CUT can be easily reduced via the line graph, and through a breakthrough technique showed that this problem is FPT when parameterized by the maximum size of the stable cutset. Recently, Komusiewicz et al. [2018] improved on the results of Kratsch and Le [2016], providing an exact exponential algorithm for MATCHING CUT running in time $\mathcal{O}^*(1.3803^n)$, as well as FPT algorithms parameterized by the distance to a cluster graph and the distance to a co-cluster graph, which improve the algorithm parameterized by the vertex cover number, since both parameters are easily seen to be smaller than the vertex cover number. For the distance to cluster parameter, they also presented a quadratic kernel; while for a combination of treewidth, maximum degree, and number of crossing edges, they showed that no polynomial kernel exists unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

A problem closely related to d-CUT is that of INTERNAL PARTITION, first studied by Thomassen [1983]. In this problem, we seek a bipartition of the vertices of an input graph such that every vertex has at least as many neighbors in its own part as in the

other part. Such a partition is called an *internal partition*. Usually, the problem is posed in a more general form: given functions $a, b : V(G) \to \mathbb{Z}_+$, we seek a bipartition (A, B) of $V(G)$ such that every $v \in A$ satisfies $\mathsf{deg}_A(v) \geq a(v)$ and every $u \in B$ satisfies $\mathsf{deg}_B(u) \geq b(u)$, where $\mathsf{deg}_A(v)$ denotes the number of neighbors of v in the set A. Such a partition is called an (a, b)-*internal partition*.

Originally, Thomassen asked in [Thomassen, 1983] whether for any pair of positive integers s, t, a graph G with $\delta(G) \geq s + t + 1$ has a vertex bipartition (A, B) with $\delta(G[A]) \geq s$ and $\delta(G[B]) \geq t$. Stiebitz [1996] answered that, in fact, for any graph G and any pair of functions $a, b : V(G) \to \mathbb{Z}_+$ satisfying $\mathsf{deg}(v) \geq a(v) + b(v) + 1$ for every $v \in V(G)$, G has an (a, b)-internal partition. Following Stiebitz's work, Kaneko [1998] showed that if G is triangle-free, then the pair a, b only needs to satisfy $\mathsf{deg}(v) \geq a(v) + b(v)$. More recently, Ma and Yang [2019] proved that, if G is $\{C_4, K_4, \text{diamond}\}$-free, then $\mathsf{deg}(v) \geq a(v) + b(v) - 1$ is enough. Furthermore, they also showed, for any pair a, b, a family of graphs such that $\mathsf{deg}(v) \geq a(v) + b(v) - 2$ for every $v \in V(G)$ that do not admit an (a, b)-internal partition.

It is conjectured that, for every positive integer r, there exists some constant n_r for which every r-regular graph with more than n_r vertices has an internal partition [DeVos, 2009; Ban and Linial, 2016] (the conjecture for r even appeared first in [Shafique and Dutton, 2002]). The cases $r \in \{3, 4\}$ have been settled by Shafique and Dutton [2002]; the case $r = 6$ has been verified by Ban and Linial [2016]. This latter result implies that every 6-regular graph of sufficiently large size has a 3-cut.

3.2 NP-hardness, polynomial cases, and exact exponential algorithm

In this section we focus on the classical complexity of the d-CUT problem, and on exact exponential algorithms. Namely, we provide the NP-hardness result in Section 3.2.1, the polynomial algorithm for graphs of bounded degree in Section 3.2.2, and a simple exact exponential algorithm in Section 3.2.3.

3.2.1 NP-hardness for regular graphs

Before stating our NP-hardness result, we need some definitions and observations.

Definition 41. A set of vertices $X \subseteq V(G)$ is said to be *monochromatic* if, for any d-cut (A, B) of G, $X \subseteq A$ or $X \subseteq B$.

Observation 42. *For fixed $d \geq 1$, the graph $K_{d+1,2d+1}$ is monochromatic. Moreover, any vertex with $d+1$ neighbors in a monochromatic set is monochromatic.*

Definition 43 (Spool)**.** For $n, d \geq 1$, a (d, n)-*spool* is the graph obtained from n copies of $K_{d+1,2d+2}$ such that, for every $1 \leq i \leq n$, one vertex of degree $d+1$ of the i-th copy is identified with one vertex of degree $d+1$ of the $(i+1 \mod n)$-th copy, so that the two chosen vertices in each copy are distinct. The *exterior* vertices of a copy are those of degree $d+1$ that are not used to interface with another copy. The *interior* vertices of a copy are those of degree $2d+2$ that do not interface with another copy.

An illustration of a $(2, 3)$-spool is shown in Figure 14.

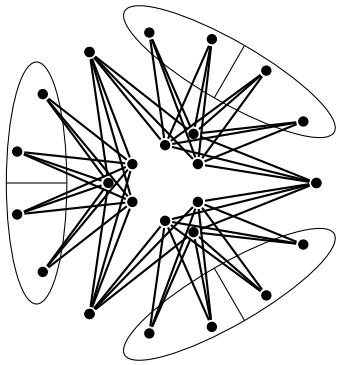

Figure 14: A $(2, 3)$-spool. Circled vertices are exterior vertices.

Observation 44. *For fixed $d \geq 1$, a (d, n)-spool is monochromatic.*

Proof. Let S be a (d, n)-spool. If $n = 1$, the observation follows by combining the two statements of Observation 42. Now let $X, Y \subsetneq S$ be two copies of $K_{d+1,2d+2}$ that share exactly one vertex v. By Observation 42, $X' = X \setminus \{v\}$ and $Y' = Y \setminus \{v\}$ are monochromatic. Since v has $d+1$ neighbors in X' and $d+1$ in Y', it follows that $X \cup Y$ is monochromatic. By repeating the same argument for every two copies of $K_{d+1,2d+2}$ that share exactly one vertex, the observation follows. □

Chvátal Chvátal [1984] proved that MATCHING CUT is NP-hard for graphs of maximum degree at least four. In the next theorem, whose proof is inspired by the reduction of Chvátal Chvátal [1984], we prove the NP-hardness of d-CUT for $(2d+2)$-regular graphs. In particular, for $d = 1$ it implies the NP-hardness of MATCHING CUT for 4-regular graphs, which is stronger than Chvátal Chvátal [1984] hardness for graphs of maximum degree four.

Theorem 45. *For every integer $d \geq 1$, d-CUT is NP-hard even when restricted to $(2d+2)$-regular graphs.*

Proof. Our reduction is from the 3-UNIFORM HYPERGRAPH BICOLORING problem, which is NP-hard; see Lovász [1973].

3-UNIFORM HYPERGRAPH BICOLORING
Instance: A hypergraph $\mathcal{H}$ with exactly three vertices in each hyperedge.
Question: Can we 2-color $V(\mathcal{H})$ such that no hyperedge is monochromatic?

Throughout this proof, i is an index representing a color, j and k are redundancy indices used to increase the degree of some sets of vertices, and ℓ and r are indices used to refer to separations of sets of exterior vertices.

Given an instance $\mathcal{H}$ of 3-UNIFORM HYPERGRAPH BICOLORING, we proceed to construct a $(2d+2)$-regular instance G of d-CUT as follows. For each vertex $v \in V(\mathcal{H})$, add a $(d, 4\mathsf{deg}(v)+1)$-spool to G. Each set of exterior vertices receives an (arbitrarily chosen) unique label from the following types: $S(v^*)$ and $S(v,e,i,j)$, such that $i,j \in [2]$ and $e \in E(\mathcal{H})$ with $v \in e$. Separate each of the labeled sets into two parts of equal size (see Figure 14). For the first type, we denote the sets by $S(v^*,i)$, $i \in [2]$; for the second type, by $S_\ell(v,e,i,j)$, $\ell \in [2]$. For each set $S(v^*,i)$, we choose an arbitrary vertex and label it with $s(v^*,i)$. To conclude the construction of vertex gadgets, add every edge between $S_1(v,e,i,j)$ and $S_2(v,e,i,j)$, and form a perfect matching between $S(v^*,1)\setminus\{s(v^*,1)\}$ and $S(v^*,2)\setminus\{s(v^*,2)\}$. Note that all inner vertices of these spools have degree $2d+2$, every vertex labeled $s(v^*,i)$ has $d+1$ neighbors, every other vertex in $S(v^*,i)$ has $d+2$, and every vertex in $S(v,e,i,j)$ has degree equal to $2d+1$. For an example of the edges between exterior vertices of the same vertex gadget, see Figure 15.

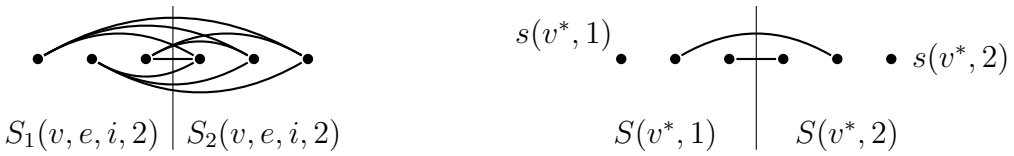

Figure 15: Relationships between exterior vertices of a vertex gadget ($d=3$).

For each color $i \in [2]$, add a $(d, n+2m)$-spool to G, where $n = |V(\mathcal{H})|$ and $m = |E(\mathcal{H})|$. Much like the exterior vertices of the vertex gadgets, we attribute unique labels: $C(v,i)$, for each $v \in V(\mathcal{H})$, and $C(e,i,j)$, for each $e \in E(\mathcal{H})$ and $j \in [2]$. Now, split the remaining vertices of each labeled set into two equal-sized parts $C_1(\cdot), C_2(\cdot)$ and label one vertex of each $C_\ell(e,i,j)$ with the label $c_\ell(e,i,j)$ and one of each $C_\ell(v,i)$ with $c_\ell(v,i)$. To conclude, add all edges from $c_\ell(v,i)$ to $C_\ell(v,i)$, add the edge $c_\ell(v,i)c_{3-\ell}(v,i)$, make each $C_\ell(e,i,j)\setminus\{c_\ell(e,i,j)\}$ into a clique, and, between $C_\ell(e,i,j)\setminus\{c_\ell(e,i,j)\}$ and $C_r(e,i,k)\setminus\{c_r(e,i,k)\}$, add edges to form a perfect matching, for $\ell,j,r,k \in [2]$. That is, each $C_\ell(e,i,j)$ forms a perfect matching with three other sets of exterior vertices. So

far, each $c_\ell(v,i)$ has degree $(d+1)+(d-1)+1 = 2d+1$, other vertices of $C_\ell(v,i)$ have degree $d+2$, each vertex in $C_\ell(e,i,j)\setminus\{c_\ell(e,i,j)\}$ has degree $(d+1)+(d-2)+3 = 2d+2$, and each vertex labeled $c_\ell(e,i,j)$ has degree $d+1$.

We now add edges between vertices of different color gadgets. In particular, we add every edge between $C_1(v,2)\setminus\{c_1(v,2)\}$ and $C_2(v,1)\setminus\{c_2(v,1)\}$. This increases the degree of these vertices to $2d+1$. An example when $d=3$ is illustrated in Figure 16.

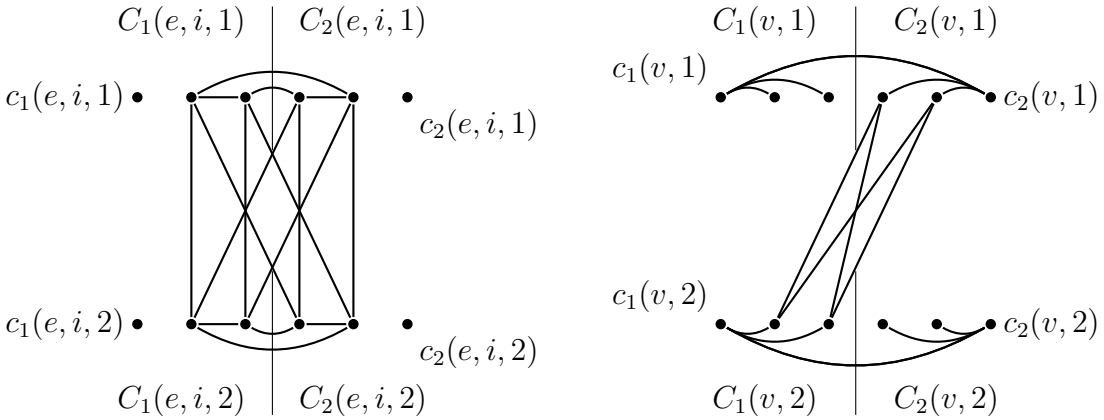

Figure 16: Relationships between exterior vertices of color gadgets ($d=3$).

As a first step to connect color gadgets and vertex gadgets, we add every edge between $s(v^*,i)$ and $C_i(v,i)$, every edge between $S(v^*,i)\setminus\{s(v^*,i)\}$ and $C_i(v,i)\setminus\{c_i(v,i)\}$, a perfect matching between $S(v^*,i)\setminus\{s(v^*,i)\}$ and $C_{3-i}(v,i)\setminus\{c_{3-i}(v,i)\}$, and the edge $s(v^*,i)c_i(v,3-i)$. Note that this last edge is fundamental, not only because it increases the degrees to the desired value, but also because, if both color gadgets belong to the same side of the cut, every $s(v^*,i)$ will have the same color and, since spools are monochromatic, so would be the *entire* graph, as discussed in more detail below. Also note that, aside from $s(v^*,i)$, no other vertex has more than d neighbors outside of its spool. The edges described in this paragraph increase the degree of every $s(v^*,i)$ by $d+1$, yielding a total degree of $2d+2$, of every vertex in $S(v^*,i)\setminus\{s(v^*,i)\}$ to $(d+2)+(d-1)+1 = 2d+2$, of every vertex in $C_i(v,i)\setminus\{c_i(v,i)\}$ to $(d+2)+d = 2d+2$, of every vertex in $C_i(v,3-i)\setminus\{c_i(v,3-i)\}$ to $(2d+1)+1 = 2d+2$, and of every $c_\ell(v,i)$ to $(2d+1)+1 = 2d+2$. Figure 17 gives an example of these connections.

For the final group of gadgets, namely hyperedge gadgets, for each $\{x,y,z\} \in E(\mathcal{H})$, each color i, and each pair $j,\ell \in [2]$, we add one additional vertex $c'_\ell(e,i,j)$ adjacent to $c_\ell(e,i,j)$, $S_\ell(x,e,i,j)$, $S_\ell(y,e,i,j)$, and $c'_{3-\ell}(e,i,j)$; finally, we add every edge between $c_\ell(e,i,j)$ and $S_\ell(z,e,i,j)$. See Figure 18 for an illustration. Note that $c'_\ell(e,i,j)$ has degree $2d+2$; the degree of $c_\ell(e,i,j)$ increased from $d+1$ to $2d+2$, and

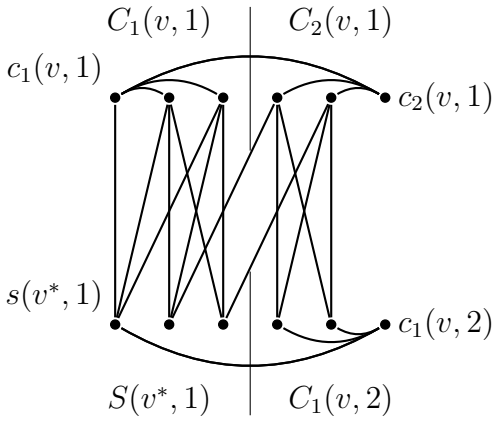

Figure 17: Relationships between exterior vertices of color and vertex gadgets ($d = 3$).

the degree of each vertex of $S_\ell(x, e, i, j)$ increased from $2d+1$ to $2d+2$. This concludes our construction of the $(2d + 2)$-regular graph G.

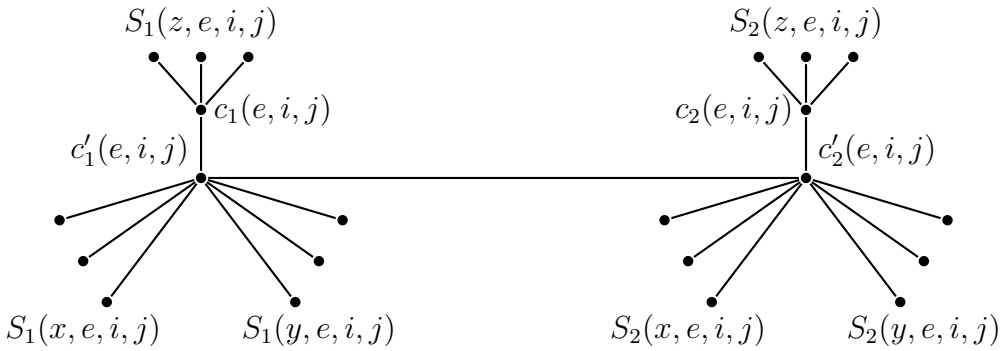

Figure 18: Hyperedge gadget ($d = 3$).

Now, suppose we are given a valid bicoloring φ of $\mathcal{H}$, and our goal is to construct a d-cut (A, B) of G. Put the gadget of color 1 in A and the other one in B. For each vertex $v \in V(\mathcal{H})$, if $\varphi(v) = 1$, put the gadget corresponding to v in A, otherwise put it in B. In the interface between these gadgets, no vertex from the color gadgets has more than d neighbors in a single vertex gadget, therefore none violates the d-cut property. As to the vertices coming from the vertex gadgets, only $s_\ell(v^*, i)$ has more than d neighbors outside of its gadget; however, it has d neighbors in the color gadget for color i and only one in color $3 - i$. Since each color gadget is in a different side of the partition, $s_\ell(v^*, i)$ does not violate the degree constraint. For each hyperedge $e = \{x, y, z\}$, put $c'_\ell(e, i, j)$ in the same set as the majority of its neighbors, this way, it will not violate the property – note that its other neighbor, $c'_{3-\ell}(e, i, j)$, will be in the same set because it will have the exact same amount of vertices on each side of the partition in its neighborhood. So, if $\varphi(x) = \varphi(y) = 1$, $c'_\ell(e, i, j) \in A$; however, since

e is not monochromatic, $\varphi(z) = 2$, so $c_\ell(e, i, j)$ has at most d neighbors in the other set. The case where $\varphi(x) \neq \varphi(y)$ is similar. Thus, we conclude that (A, B) is indeed a d-cut of G.

Conversely, take a d-cut (A, B) of G and construct a bicoloring of $\mathcal{H}$ such that $\varphi(v) = 1$ if and only if the spool corresponding to v is in A. Suppose that this process results in some hyperedge $e = \{x, y, z\} \in E(\mathcal{H})$ begin monochromatic. That is, there is some hyperedge gadget where $S_\ell(x, e, i, j)$, $S_\ell(y, e, i, j)$, and $S_\ell(z, e, i, j)$ are in A, which implies that $c'_\ell(e, i, j) \in A$ and, consequently, that $c_\ell(e, i, j) \in A$ for every $\ell, i, j \in [2]$. However, since $c_\ell(e, 1, j)$ and $c_\ell(e, 2, j)$ are in A and a color gadget is monochromatic, both color gadgets belong to A, which in turn implies that every $s(v^*, i)$ has $d + 1$ neighbors in A and, therefore, must also be in A by Observation 42. Moreover, since spools are monochromatic, every vertex gadget is in A, implying that the entire graph belongs to A, contradicting the hypothesis that (A, B) is a d-cut of G. □

The graphs constructed by the above reduction are neither planar nor bipartite, but they are regular, a result that we were unable to find in the literature for MATCHING CUT. Note that every planar graph has a d-cut for every $d \geq 5$, so only the cases $d \in \{2, 3, 4\}$ remain open, as the case $d = 1$ is known to be NP-hard Bonsma [2009]. Concerning graphs of bounded diameter, Le and Le Le and Le [2016] prove the NP-hardness of MATCHING CUT for graphs of diameter at least three by reducing MATCHING CUT to itself. It can be easily seen that the same construction given by Le and Le Le and Le [2016], but reducing d-CUT to itself, also proves the NP-hardness of d-CUT for every $d \geq 1$.

Corollary 46. *For every integer $d \geq 1$, d-CUT is NP-hard for graphs of diameter at least three.*

We leave as an open problem to determine whether there exists a polynomial-time algorithm for d-CUT for graphs of diameter at most two for every $d \geq 2$, as it is the case for $d = 1$ Le and Le [2016].

3.2.2 Polynomial algorithm for graphs of bounded degree

Our next result is a natural generalization of Chvátal's algorithm Chvátal [1984] for MATCHING CUT on graphs of maximum degree three.

Theorem 47. *For any graph G and integer $d \geq 1$ such that $\Delta(G) \leq d + 2$, it can be decided in polynomial time if G has a d-cut. Moreover, for $d = 1$ any graph G with*

$\Delta(G) \leq 3$ and $|V(G)| \geq 8$ has a matching cut, for $d = 2$ any graph G with $\Delta(G) \leq 4$ and $|V(G)| \geq 6$ has a 2-cut, and for $d \geq 3$ any graph G with $\Delta(G) \leq d+2$ has a d-cut.

Proof. We may assume that G is connected, as otherwise it always admits a d-cut. If G is a tree, any edge is a cut edge and, consequently, a d-cut is easily found. So let C be a shortest cycle of G. If $d = 1$ we use Chvátal's result [Chvátal, 1984] together with the size bound of eight observed by Moshi [1989]; hence, we may assume that $d \geq 2$. In the case that $V(G) = C$, we may pick any vertex v and note that $(\{v\}, C \setminus \{v\})$ is a d-cut.

Suppose first that $|C| = 3$ and $d = 2$. If $(C, V(G) \setminus C)$ is a 2-cut, we are done. Otherwise, there is some vertex $v \notin C$ with three neighbors in C (since by the hypothesis on $\Delta(G)$, every vertex in C has at most two neighbors in $G - C$) and, consequently, $Q := C \cup \{v\}$ induces a K_4. If $V(G) = Q$, we can arbitrarily partition Q into two sets with two vertices each and get a 2-cut of G. Also, if no other $u \notin Q$ has three neighbors in Q, $(Q, V(G) \setminus Q)$ is a 2-cut of G. If there is such a vertex u, let $R := Q \cup \{u\}$. If $V(G) = R$, then clearly G has no 2-cut. Note that $|Q| = 5$, and this will be the only case in the proof where G does not have a d-cut. Otherwise, if $V(G) \neq R$, $(R, V(G) \setminus R)$ is a 2-cut, because no vertex outside of R can be adjacent to more than two vertices in R, and we are done.

If $|C| = 3$ and $d \geq 3$, then clearly $(C, V(G) \setminus C)$ is a d-cut, and we are also done.

Otherwise, that is, if $|C| \geq 4$, we claim that $(C, V(G) \setminus C)$ is always a d-cut. For $v \in C$, note that $\mathsf{deg}(v) \leq d + 2$, hence v has at most d neighbors in $G - C$. For $v \in V(G) \setminus C$, if $|C| \geq 5$, necessarily $\mathsf{deg}_C(v) \leq 1$, as otherwise we would find a cycle in G shorter than C, and therefore $(C, V(G) \setminus C)$ is a d-cut. By a similar argument, if $|C| = 4$, then $\mathsf{deg}_C(v) \leq 2$, and the theorem follows as we assume that $d \geq 2$. □

Theorems 45 and 47 present a "quasi-dichotomy" for d-cut on graphs of bounded maximum degree. Specifically, for $\Delta(G) \in \{d + 3, \ldots, 2d + 1\}$, the complexity of the problem remains unknown. However, we believe that most, if not all, of these open cases can be solved in polynomial time; see the discussion in Section 3.5.

3.2.3 Exact exponential algorithm

To conclude this section, we present a simple exact exponential algorithm which, for every $d \geq 1$, runs in time $\mathcal{O}^*(c_d^n)$ for some constant $c_d < 2$. For the case $d = 1$, the currently known algorithms Kratsch and Le [2016]; Komusiewicz et al. [2018] exploit structures that appear to get out of control when d increases, and so has a better running time than the one described below.

When an instance of size n branches into t subproblems of sizes at most $n - s_1, \ldots, n - s_t$, respectively, the vector $(s_1, \ldots, s_t)$ is called the *branching vector* of this branching rule, and the unique positive real root of the equation $x^n - \sum_{i \in [t]} x^{n-s_i} = 0$ is called the *branching factor* of the rule. The total complexity of a branching algorithm is given by $\mathcal{O}^*(\alpha^n)$, where α is the largest branching factor among all rules of the algorithm. For more on branching algorithms, we refer to Fomin and Kratsch [2010].

Theorem 48. *For every fixed integer $d \geq 1$ and n-vertex graph G, there is an algorithm that solves d-*CUT *in time $\mathcal{O}^*(c_d^n)$, for some constant $1 < c_d < 2$.*

Proof. Our algorithm takes as input G and outputs a d-cut (A, B) of G, if it exists. To do so, we build a branching algorithm that maintains, at every step, a tripartition of $V(G) = A \dot\cup B \dot\cup D$ such that (A, B) is a d-cut of $G \setminus D$. The central idea of our rules is to branch on small sets of vertices (namely, of size at most $d + 1$) at each step such that either at least one bipartition of the set forces some other vertex to choose a side of the cut, or we can conclude that there is at least one bipartition that violates the d-cut property. First, we present our reduction rules, which are applied following this order at the beginning of each recursive step.

R1 If (A, B) violates the d-cut property, output NO.

R2 If $D = \emptyset$, we have a d-cut of G. Output (A, B).

R3 If there is some $v \in D$ with $\mathsf{deg}_A(v) \geq d + 1$ and $\mathsf{deg}_B(v) \geq d + 1$, output NO.

R4 While there is some $v \in D$ with $\mathsf{deg}_A(v) \geq d + 1$ (resp. $\mathsf{deg}_B(v) \geq d + 1$), add v to A (resp. B).

Our branching rules, and their respective branching vectors, are listed below.

B1 If there is some $v \in A \cup B$ with $\mathsf{deg}_D(v) \geq d + 1$, choose a set $X \subseteq N_D(v)$ of size d and branch on all possible possible bipartitions of X. Note that, if all vertices of X are in the other side of v, at least one vertex of $N_D(v) \setminus X$ must be in the same side as v. As such, this branching vector is of the form $\{d + 1\} \times \{d\}^{2^d - 1}$.

B2 If there is some $v \in A$ (resp. B) such that $\mathsf{deg}_B(v) + \mathsf{deg}_D(v) \geq d + 1$ (resp. $\mathsf{deg}_A(v) + \mathsf{deg}_D(v) \geq d + 1$), choose a set $X \subseteq N_D(v)$ of size $s = d + 1 - \mathsf{deg}_B(v)$ (resp. $s = d + 1 - \mathsf{deg}_A(v)$) and branch on every possible bipartition of X. Since rule B1 was not applied, we have that $\mathsf{deg}_D(v) \leq d$, $\mathsf{deg}_B(v) \geq 1$ (resp. $\mathsf{deg}_A(v) \geq 1$), and $s \leq d$. If all vertices of X were placed in B (resp. A), we would violate the d-cut property and, thus, do not need to investigate this branch

of the search. In the worst case, namely when $s = d$, this yields the branching vector $\{d\}^{2^d-1}$.

We now claim that, if none of the above rules is applicable, we have that $(A \cup D, B)$ is a d-cut of G. To see that this is the case, suppose that there is some vertex $v \in V(G)$ that violates the d-cut property; that is, it has a set Y of $d+1$ neighbors across the cut.

Suppose that $v \in B$. Then $Y \subseteq A \cup D$, so we have $\mathsf{deg}_A(v) + \mathsf{deg}_D(v) \geq d+1$, in which case rule B2 could be applied, a contradiction. Thus, we have that $v \notin B$, so $Y \subseteq B$ and either $v \in A$ or $v \in D$; in the former case, again by rule R1, (A, B) would not be a d-cut. In the latter case, we would have that $\mathsf{deg}_B(v) \geq d+1$, but then rule R4 would still be applicable. Consequently, $v \notin A \cup B \cup D = V(G)$, so such a vertex does not exist, and thus we have that $(A \cup D, B)$ is a d-cut of G. Note that a symmetric argument holds for the bipartition $(A, B \cup D)$. Before executing the above branching algorithm, we need to ensure that $A \neq \emptyset$ and $B \neq \emptyset$. To do that, for each possible pair of vertices $u, v \in V(G)$, we execute the entire algorithm starting with $A := \{u\}$ and $B := \{v\}$.

As to the running time of the algorithm, for rule B2 we have that the unique positive real root of $x^n - (2^d - 1)x^{n-d} = 0$ is of the closed form $x = \sqrt[d]{2^d - 1} < 2$. For rule B1, we have that the polynomial associated with the recurrence relation, $p_d(x) = x^n - (2^d - 1)x^{n-d} - x^{n-d-1}$, verifies $p_d(1) = 1 - 2^d < 0$ and $p_d(2) = 2^{n-d-1} > 0$. Since it is a continuous function and $p_d(x)$ has an unique positive real root c_d, it holds that $1 < c_d < 2$. The final complexity of our algorithm is $\mathcal{O}^*(c_d^n)$, with $\sqrt[d]{2^d - 1} < c_d < 2$, since $p_d\left(\sqrt[d]{2^d - 1}\right) = -(2^d - 1)^{\frac{n-d-1}{d}} < 0$. Table 5 presents the branching factors for some values of d for our two branching rules.

d	1	2	3	4	5	6	7
B1	1.6180	1.8793	1.9583	1.9843	1.9937	1.9973	1.9988
B2	1.0000	1.7320	1.9129	1.9679	1.9873	1.9947	1.9977

Table 5: Branching factors for some values of d.

□

3.3 Parameterized algorithms and kernelization

In this section we focus on the parameterized complexity of d-Cut. More precisely, in Section 3.3.1 we consider as the parameter the number of edges crossing the cut, in

Section 3.3.2 the treewidth of the input graph, in Section 3.3.3 the distance to cluster (in particular, we provide a quadratic kernel), and in Section 3.3.4 the distance to co-cluster.

3.3.1 Crossing edges

In this section we consider as the parameter the maximum number of edges crossing the cut. In a nutshell, our approach is to use as a black box one of the algorithms presented by Marx et al. Marx et al. [2010] for a class of separation problems. Their fundamental problem is $\mathcal{G}$-MINCUT, for a fixed class of graphs $\mathcal{G}$, which we state formally, along with their main result, below.

$\mathcal{G}$-MINCUT
Instance: A graph G, vertices s, t, and an integer k.
Parameter: Is there an induced subgraph H of G with at most k vertices such that $H \in \mathcal{G}$ and H is an $s-t$ separator?
Question: The integer k.

13.5

Theorem 49 (Theorem 3.1 in Marx et al. [2010])**.** *If $\mathcal{G}$ is a decidable and hereditary graph class, $\mathcal{G}$-MINCUT is* FPT*.*

To be able to apply Theorem 49, we first need to specify a graph class to which, on the line graph, our separators correspond. We must also be careful to guarantee that the removal of a separator in the line graph leaves non-empty components in the input graph. To accomplish that, for each $v \in V(G)$, we add a private clique of size $2d$ adjacent only to it, choose one arbitrary vertex v' in each of them, and our algorithm will ask for the existence of a "special" separator of the appropriate size between every pair of chosen vertices of two distinct private cliques. We assume henceforth that these private cliques have been added to the input graph G.

For each integer $d \geq 1$, we define the graph class $\mathcal{G}_d$ as follows.

Definition 50. A graph H belongs to $\mathcal{G}_d$ if and only if its maximum clique size is at most d.

Note that $\mathcal{G}_d$ is clearly decidable and hereditary for every integer $d \geq 1$.

Lemma 51. *G has a d-cut if and only if $L(G)$ has a vertex separator belonging to $\mathcal{G}_d$.*

Proof. Let $H = L(G)$, (A, B) be a d-cut of G, and $F \subseteq V(H)$ be the set of vertices such that $e_{uv} \in F$ if and only if $u \in A$ and $v \in B$, or vice-versa. The fact that F is a separator of H follows directly from the hypothesis that (A, B) is a cut of G. Now, to show that $H[F] \in \mathcal{G}_d$, suppose for contradiction that $H[F]$ contains a clique Q with more than d vertices. That is, there are at least $d + 1$ edges of G that are pairwise intersecting and with one endpoint in A and the other in B. Note, however, that for at least one of the parts, say A, there is also *at most* one vertex with an edge in $Q \subseteq E(G)$, as otherwise there would be two non-adjacent vertices in the clique $Q \subseteq V(H)$. As such, A has only one vertex and we conclude that every edge in Q has an endpoint in A, but this, on the other hand, implies that A has $d+1$ neighbors in B, contradicting the hypothesis that (A, B) is a d-cut of G.

For the converse, take a vertex separator $S \subseteq V(H)$ such that $H[S] \in \mathcal{G}_d$ and let E_S be the edges of G corresponding to S. Let G' be the graph where each vertex corresponds to a connected component of $G - E_S$ and two vertices are adjacent if and only if there is an edge in E_S between vertices of the respective components. Let Q_r be an arbitrarily chosen connected component of $G - E_S$. Now, for each component at an odd distance from Q_r in G', add that component to B; all other components are placed in A. We claim that (A, B) is a d-cut of G. Let $F \subseteq E_S$ be the set of edges with one endpoint in A and the other in B. Note that $G - F$ is disconnected due to the construction of A and B. If there is some $v \in A$ with more than d neighbors in B, we obtain that there is some clique of equal size in $H[S]$, contradicting the hypothesis that this subgraph belongs to $\mathcal{G}_d$. □

Theorem 52. *For every $d \geq 1$, there is an* FPT *algorithm for d-*Cut *parameterized by k, the maximum number of edges crossing the cut.*

Proof. For each pair of vertices $s, t \in V(G)$ that do not belong to the private cliques, our goal is to find a subset of vertices $S \subseteq V(L(G))$ of size at most k that separates s and t such that $L(G)[S] \in \mathcal{G}_d$. This is precisely what is provided by Theorem 49, and the correctness of this approach is guaranteed by Lemma 51. Since we perform a quadratic number of calls to the algorithm given by Theorem 49, our algorithm still runs in FPT time. □

As to the running time of the FPT algorithm given by Theorem 52, the treewidth reduction technique of Marx et al. [2010] relies on the construction of a monadic second order logic (MSOL) expression and Courcelle's Theorem Courcelle [1990] to guarantee fixed-parameter tractability, and therefore it is hard to provide an explicit running time in terms of k.

3.3.2 Treewidth

We proceed to present an algorithm for d-CUT parameterized by the treewidth of the input graph that, in particular, improves the running time of the best known algorithm for MATCHING CUT Aravind et al. [2017]. For the definitions of treewidth we refer to Robertson and Seymour [1986]; Cygan et al. [2015b]. We state here an adapted definition of nice tree decomposition which shall be useful in our algorithm.

Definition 53. (Nice tree decomposition) A tree decomposition $(T, \mathcal{B})$ of a graph G is said to be *nice* if it T is a tree rooted at an empty bag $r(T)$ and each of its bags is from one of the following four types:

1. *Leaf node*: a leaf x of T with $|B_x| = 2$ and no children.
2. *Introduce node*: an inner node x of T with one child y such that $B_x \setminus B_y = \{u\}$, for some $u \in V(G)$.
3. *Forget node*: an inner node x of T with one child y such that $B_y \setminus B_x = \{u\}$, for some $u \in V(G)$.
4. *Join node*: an inner node x of T with two children y, z such that $B_x = B_y = B_z$.

In the next theorem, note that the assumption that the given tree decomposition is *nice* is not restrictive, as any tree decomposition can be transformed into a nice one of the same width in polynomial time Kloks [1994].

Theorem 54. *For every integer $d \geq 1$, given a nice tree decomposition of G of width $\mathsf{tw}(G)$, d-CUT can be solved in time $\mathcal{O}^*\big(2^{\mathsf{tw}(G)+1}(d+1)^{2\mathsf{tw}(G)+2}\big)$.*

Proof. As expected, we will perform dynamic programming on a nice tree decomposition. For this proof, we denote a d-cut of G by (L, R) and suppose that we are given a total ordering of the vertices of G. Let $(T, \mathcal{B})$ be a nice tree decomposition of G rooted at a node $r \in V(T)$. For a given node $x \in T$, an entry of our table is indexed by a triple $(A, \boldsymbol{\alpha}, t)$, where $A \subseteq B_x$, $\boldsymbol{\alpha} \in (\{0\} \cup [d])^{\mathsf{tw}(G)+1}$, and t is a binary value. Each coordinate a_i of α indicates how many vertices *outside* of B_x the i-th vertex of B_x has in the other side of the partition. More precisely, we denote by $f_x(A, \boldsymbol{\alpha}, t)$ the binary value indicating whether or not $V(G_x)$ has a bipartition (L_x, R_x) such that $L_x \cap B_x = A$, every vertex $v_i \in B_x$ has exactly a_i neighbors in the other side of the partition (L_x, R_x) outside of B_x, and both L_x and R_x are non-empty if and only if $t = 1$. Note that G admits a d-cut if and only if $f_r(\emptyset, \mathbf{0}, 1) = 1$. Figure 19 gives an example of an entry in the dynamic programming table and the corresponding solution on the subtree.

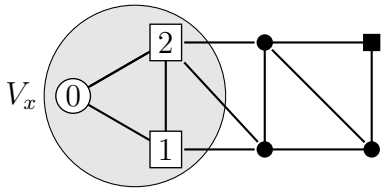

Figure 19: Example of dynamic programming state and corresponding solution on the subtree. Square vertices belong to A, circles to B. Numbers indicate the respective value of α_i ($d = 3$).

We say that an entry $(A, \boldsymbol{\alpha}, t)$ for a node x is *valid* if for every $v_i \in A$, $|N(v_i) \cap (B_x \setminus A)| + a_i \leq d$, for every $v_j \in B_x \setminus A$, $|N(v_i) \cap A| + a_j \leq d$, and if $B_x \setminus A \neq \emptyset$ then $t = 1$; otherwise the entry is *invalid*. Moreover, note that if $f_x(A, \boldsymbol{\alpha}, t) = 1$, the corresponding bipartition (L_x, R_x) of $V(G_x)$ is a d-cut if and only if $(A, \boldsymbol{\alpha}, t)$ is valid and $t = 1$.

We now explain how the entries for a node x can be computed, assuming recursively that the entries for their children have been already computed. We distinguish the four possible types of nodes. Whenever $(A, \boldsymbol{\alpha}, t)$ is invalid or absurd (with, for example, $a_i < 0$) we define $f_x(A, \alpha, t)$ to be 0, and for simplicity we will not specify this in the equations stated below.

- Leaf node: Since $|B_x| = 2$, for every $A \subseteq B_x$, we can set $f_x(A, \mathbf{0}, t) = 1$ with $t = 1$ if and only if $B_x \setminus A \neq \emptyset$. These are all the possible partitions of B_x, taking $\mathcal{O}(1)$ time to be computed.

- Introduce node: Let y be the child of x and $B_x \setminus B_y = \{v_i\}$. The transition is given by the following equation, where $\boldsymbol{\alpha}^*$ has entries equal to $\boldsymbol{\alpha}$ but without the coordinate corresponding to v_i. If $a_i > 0$, $f_x(A, \boldsymbol{\alpha}, t)$ is invalid since v_i has no neighbors in $G_x - B_x$.

$$f_x(A, \boldsymbol{\alpha}, t) = \begin{cases} f_y(A \setminus \{v\}, \boldsymbol{\alpha}^*, t), & \text{if } A = B_x \text{ or } A = \emptyset. \\ \max_{t' \in \{0,1\}} f_y(A \setminus \{v\}, \boldsymbol{\alpha}^*, t'), & \text{otherwise.} \end{cases}$$

  For the first case, G_x has a bipartition (which will also be a d-cut if $t = 1$) represented by $(A, \boldsymbol{\alpha}, t)$ only if G_y has a bipartition (d-cut), precisely because, in both G_x and G_y, the entire bag is in one side of the cut. For the latter case, if G_y has a bipartition, regardless if it is a d-cut or not, G_x has a d-cut because B_x is not contained in a single part of the cut, unless the entry is invalid. The computation for each of these nodes takes $\mathcal{O}(1)$ time per entry.

- Forget node: Let y be the child of x and $B_y \setminus B_x = \{v_i\}$. In the next equation, $\boldsymbol{\alpha}'$ has the same entries as $\boldsymbol{\alpha}$ with the addition of entry a_i corresponding to v_i and, for each $v_j \in A \cap N(v_i)$, $a'_j = a_j - 1$. Similarly, for $\boldsymbol{\alpha}''$, for each $v_j \in (B_x \setminus A) \cap N(v_i)$, $a''_j = a_j - 1$.

$$f_x(A, \boldsymbol{\alpha}, t) = \max_{a_i \in \{0\} \cup [d]} \max\{f_y(A, \boldsymbol{\alpha}', t),\ f_y(A \cup \{v_i\}, \boldsymbol{\alpha}'', t)\}.$$

  Note that $\boldsymbol{\alpha}'$ and $\boldsymbol{\alpha}''$ take into account the forgetting of v_i; its neighbors get an additional neighbor outside of B_x that is in the other side of the bipartition. Moreover, since we inspect the entries of y for every possible value of a_i, if at least one of them represented a feasible bipartition of G_y, the corresponding entry on $f_y(\cdot)$ would be non-zero and, consequently, $f_x(A, \boldsymbol{\alpha}, t)$ would also be non-zero. Computing an entry for a forget node takes $\mathcal{O}(d)$ time.

- Join node: Finally, for a join node x with children y and z, a *splitting* of $\boldsymbol{\alpha}$ is a pair $\boldsymbol{\alpha}_y, \boldsymbol{\alpha}_z$ such that for every coordinate a_j of $\boldsymbol{\alpha}$, it holds that the sum of j-th coordinates of $\boldsymbol{\alpha}_y$ and $\boldsymbol{\alpha}_z$ is equal to a_j. The set of all splittings is denoted by $S(\boldsymbol{\alpha})$ and has size $\mathcal{O}\big((d+1)^{\mathsf{tw}(G)+1}\big)$. As such, we define our transition function as follows.

$$f_x(A, \boldsymbol{\alpha}, t) = \max_{t \le t_y + t_z \le 2t} \max_{S(\boldsymbol{\alpha})} f_y(A, \boldsymbol{\alpha}_y, t_y) \cdot f_z(A, \boldsymbol{\alpha}_z, t_z).$$

  The condition $t \le t_y + t_z \le 2t$ enforces that, if $t = 1$, at least one of the graphs G_y, G_z must have a d-cut; otherwise, if $t = 0$, neither of them can. When iterating over all splittings of $\boldsymbol{\alpha}$, we are essentially testing all possible counts of neighbors outside of B_y such that there exists some entry for node z such that $\boldsymbol{\alpha}_y + \boldsymbol{\alpha}_z = \boldsymbol{\alpha}$. Finally, $f_x(A, \boldsymbol{\alpha}, t)$ is feasible if there is at least one splitting and t_y, t_z such that both G_y and G_z admit a bipartition. This node type, which is the bottleneck of our dynamic programming approach, takes $\mathcal{O}\big((d+1)^{\mathsf{tw}(G)+1}\big)$ time per entry.

Consequently, since we have $\mathcal{O}(\mathsf{tw}(G)) \cdot n$ nodes in a nice tree decomposition, spend $\mathcal{O}(\mathsf{tw}(G)^2)$ to detect an invalid entry, have $\mathcal{O}\big(2^{\mathsf{tw}(G)+1}(d+1)^{\mathsf{tw}(G)+1}\big)$ entries per node, each taking at most $\mathcal{O}\big((d+1)^{\mathsf{tw}(G)+1}\big)$ time to be computed, our algorithm runs in time $\mathcal{O}\big(\mathsf{tw}(G)^3 2^{\mathsf{tw}(G)+1}(d+1)^{2\mathsf{tw}(G)+2} \cdot n\big)$, as claimed. □

From Theorem 54 we immediately get the following corollary, which improves over the algorithm given by Aravind et al. Aravind et al. [2017].

Corollary 55. *Given a nice tree decomposition of G of width $\mathsf{tw}(G)$,* Matching Cut *can be solved in time $\mathcal{O}^*\left(8^{\mathsf{tw}(G)}\right)$.*

3.3.3 Kernelization and distance to cluster

The proof of the following theorem consists of a simple generalization to every $d \geq 1$ of the construction given by Komusiewicz et al. Komusiewicz et al. [2018] for $d = 1$.

Theorem 56. *For any fixed $d \geq 1$, d-*Cut *does not admit a polynomial kernel when simultaneously parameterized by k, Δ, and $\mathsf{tw}(G)$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

Proof. We show that the problem cross-composes into itself. Start with t instances $G_1, \ldots, G_t$ of d-Cut. First, pick an arbitrary vertex $v_i \in V(G_i)$, for each $i \in [t]$. Second, for $i \in [t-1]$, add a copy of K_{2d}, call it $K(i)$, every edge between v_i and $K(i)$, and every edge between $K(i)$ and v_{i+1}. This concludes the construction of G, which for $d = 1$ coincides with that presented by Komusiewicz et al. Komusiewicz et al. [2018].

Suppose that (A, B) is a d-cut of some G_i and that $v_i \in A$. Note that $(G \setminus B, B)$ is a d-cut of G since the only edges in the cut are those between A and B. For the converse, take some d-cut (A, B) of G and note that every vertex in the set $\{v_t\} \bigcup_{i \in [t-1]} \{v_i\} \cup K(i)$ is contained in the same side of the partition, say A. Since $B \neq \emptyset$, for any edge uv crossing the cut, there is some i such that $\{u, v\} \in V(G_i)$, which implies that there is some i (possibly more than one) such that $(A \cap V(G_i), B \cap V(G_i))$ must also be a d-cut of G_i.

That the treewidth, maximum degree, and number of edges crossing the partition are bounded by n, the maximum number of vertices of the graphs G_i, is a trivial observation. □

We now proceed to show that d-Cut admits a polynomial kernel when parameterizing by the *distance to cluster* parameter, denoted by dc. A *cluster graph* is a graph such that every connected component is a clique; the *distance to cluster* of a graph G is the minimum number of vertices we must remove from G to obtain a cluster graph. Our results are heavily inspired by the work of Komusiewicz et al. Komusiewicz et al. [2018]. Indeed, most of our reduction rules are natural generalizations of theirs. However, we need some extra observations and rules that only apply for $d \geq 2$, such as Rule 8.

We denote by $U = \{U_1, \ldots, U_t\}$ a set of vertices such that $G - U$ is a cluster graph, and each U_i is called a *monochromatic part* or *monochromatic set* of U, and we will maintain the invariant that these sets are indeed monochromatic. Initially, we set each U_i as a singleton. In order to simplify the analysis of our instance, for each U_i of

size at least two, we will have a private clique of size $2d$ adjacent to every vertex of U_i, which we call X_i. The *merge* operation between U_i and U_j is the following modification: delete $X_i \cup X_j$, set U_i as $U_i \cup U_j$, U_j as empty, and add a new clique of size $2d$, $X_{i,j}$, which is adjacent to every element of the new U_i. We say that an operation is *safe* if the resulting instance is a YES instance if and only if the original instance was.

Observation 57. *If $U_i \cup U_j$ is monochromatic, merging U_i and U_j is safe.*

It is worth mentioning that the second case of the following rule is not needed in the corresponding rule in Komusiewicz et al. [2018]; we need it here to prove the safeness of Rules 7 and 8.

Reduction Rule 1. *Suppose that $G - U$ has some cluster C such that*

1. *$(C, V(G) \setminus C)$ is a d-cut, or*
2. *$|C| \leq 2d$ and there is $C' \subseteq C$ such that $(C', G \setminus C')$ is a d-cut.*

Then output YES.

After applying Rule 1, for every cluster C, C has some vertex with at least $d+1$ neighbors in U, or there is some vertex of U with $d+1$ neighbors in C. Moreover, note that no cluster C with at least $2d+1$ vertices can be partitioned in such a way that one side of the cut is composed only by a proper subset of vertices of C.

The following definition is a natural generalization of the definition of the set N^2 given by Komusiewicz et al. Komusiewicz et al. [2018]. Essentially, it enumerates some of the cases where a vertex, or set of vertices, is monochromatic, based on its relationship with U. However, there is a crucial difference that keeps us from achieving equivalent bounds both in terms of running time and size of the kernel, and which makes the analysis and some of the rules more complicated than in Komusiewicz et al. [2018]. Namely, for a vertex to be forced into a particular side of the cut, it must have at least $d+1$ neighbors in that side; moreover, a vertex of U being adjacent to $2d$ vertices of a cluster C implies that C is monochromatic. Only if $d = 1$, i.e., when we are dealing with matching cuts, the equality $d+1 = 2d$ holds. This gap between $d+1$ and $2d$ is the main difference between our kernelization algorithm for general d and the one shown in Komusiewicz et al. [2018] for MATCHING CUT, and the main source of the differing complexities we obtain. In particular, for $d = 1$ the fourth case of the following definition is a particular case of the third one, but this is not true anymore for $d \geq 2$. For an example of what the set induced by Definition 58 looks like, please refer to Figure 20.

Definition 58. For a monochromatic part $U_i \subseteq U$, let $N^{2d}(U_i)$ be the set of vertices $v \in V(G) \setminus U$ for which at least one of the following holds:

1. v has at least $d+1$ neighbors in U_i.
2. v is in a cluster C of size at least $2d+1$ in $G-U$ such that there is some vertex of C with at least $d+1$ neighbors in U_i.
3. v is in a cluster C of $G-U$ and some vertex in U_i has $2d$ neighbors in C.
4. v is in a cluster C of $G-U$ of size at least $2d+1$ and some vertex in U_i has $d+1$ neighbors in C.

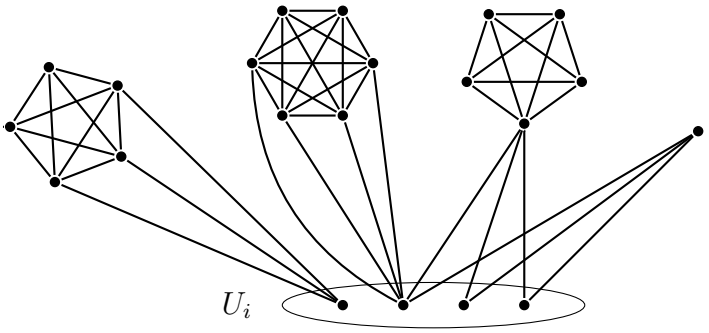

Figure 20: The four cases that define membership in $N^{2d}(U_i)$ for $d = 2$.

Observation 59. *For every monochromatic part U_i, $U_i \cup N^{2d}(U_i)$ is monochromatic.*

The next rules aim to increase the size of monochromatic sets. In particular, Rule 2 translates the transitivity of the monochromatic property, while Rule 3 identifies a case where merging the monochromatic sets is inevitable.

Reduction Rule 2. *If $N^{2d}(U_i) \cap N^{2d}(U_j) \neq \emptyset$, merge U_i and U_j.*

Reduction Rule 3. *If there there is a set of $2d+1$ vertices $L \subseteq V(G)$ with two common neighbors u, u' such that $u \in U_i$ and $u' \in U_j$, merge U_i and U_j.*

Proof of safeness of Rule 3. Suppose that in some d-cut (A, B), $u \in A$ and $u' \in B$, this implies that at most d elements of L are in A and at most d are in B, which is impossible since $|L| = 2d+1$. □

We say that a cluster is *small* if it has at most $2d$ vertices, and *big* otherwise. Moreover, a vertex in a cluster is *ambiguous* if it has neighbors in more than one U_i. A cluster is *ambiguous* if it has an ambiguous vertex, and *fixed* if it is contained in some $N^{2d}(U_i)$.

Observation 60. *If G is reduced by Rule 1, every big cluster is ambiguous or fixed.*

Proof. Since Rule 1 cannot be applied, every cluster C has either one vertex v with at least $d+1$ neighbors in U or there is some vertex of a set U_i with $d+1$ neighbors in C. In the latter case, by applying the fourth case in the definition of $N^{2d}(U_i)$, we conclude that C is fixed. In the former case, either v has $d+1$ neighbors in the same U_i, in which case C is fixed, or its neighborhood is spread across multiple monochromatic sets, and so v and, consequently, C are ambiguous. □

Our next goal is to bound the number of vertices outside of U.

Reduction Rule 4. *If there are two clusters C_1, C_2 contained in some $N^{2d}(U_i)$, then add every edge between C_1 and C_2.*

Proof of safeness of Rule 4. It follows directly from the fact that $C_1 \cup C_2$ is a larger cluster, $C_1 \cup C_2 \subseteq N^{2d}(U_i)$, and that adding edges between vertices of a monochromatic set preserves the existence of a d-cut. □

The next lemma follows from the pigeonhole principle and exhaustive application of Rule 4.

Lemma 61. *If G has been reduced by Rules 1 through 4, then G has $\mathcal{O}(|U|)$ fixed clusters.*

Reduction Rule 5. *If there is some cluster C with at least $2d+2$ vertices such that there is some $v \in C$ with no neighbors in U, remove v from G.*

Proof of safeness of Rule 5. That G has a d-cut if and only if $G - v$ has a d-cut follows directly from the hypothesis that C is monochromatic in G and the fact that $|C \setminus \{v\}| \geq 2d+1$ implies that $C \setminus \{v\}$ is monochromatic in $G - v$. □

By Rule 5, we now have the additional property that, if C has more than $2d+1$ vertices, all of them have at least one neighbor in U. The next rule provides a uniform structure between a big cluster C and the sets U_i such that $C \subseteq N^{2d}(U_i)$.

Reduction Rule 6. *If a cluster C has at least $2d+1$ elements and there is some U_i such that $C \subseteq N^{2d}(U_i)$, remove all edges between C and U_i, choose $u \in U_i$, $\{v_1, \ldots, v_{d+1}\} \subseteq C$ and add the edges $\{uv_i\}_{i \in [d+1]}$ to G.*

Proof of safeness of Rule 6. Let G' be the graph obtained after the operation is applied. If G has some d-cut (A, B), since $U_i \cup N^{2d}(U_i)$ is monochromatic, no edge between U_i and C crosses the cut, so (A, B) is also a d-cut of G'. For the converse,

take a d-cut (A', B') of G'. Since C has at least $2d+1$ vertices and there is some $u \in U_i$ such that $|N(u) \cap C| = d + 1$, $C \in N^{2d}(U_i)$ in G'. Therefore, no edge between C and U_i crosses the cut and (A', B') is also a d-cut of G. □

We have now effectively bounded the number of vertices in big clusters by a polynomial in U, as shown below.

Lemma 62. *If G has been reduced by Rules 1 through 6, then G has $\mathcal{O}(d|U|^2)$ ambiguous vertices and $\mathcal{O}(d|U|^2)$ big clusters, each with $\mathcal{O}(d|U|)$ vertices.*

Proof. To show the bound on the number of ambiguous vertices, take any two vertices $u \in U_i$, $u' \in U_j$. Since we have $\binom{|U|}{2}$ such pairs, if we had at least $(2d+1)\binom{|U|}{2}$ ambiguous vertices, by the pigeonhole principle, there would certainly be $2d + 1$ vertices in $V \setminus U$ that are adjacent to one pair, say u and u'. This, however, contradicts the hypothesis that Rule 3 has been applied, and so we have $\mathcal{O}(d|U|^2)$ ambiguous vertices.

The above discussion, along with Lemma 61 and Observation 60, imply that the number of big clusters is $\mathcal{O}(d|U|^2)$. For the bound on their sizes, take some cluster C with at least $2d + 2$ vertices. Due to the application of Rule 5, every vertex of C has at least one neighbor in U. Moreover, there is at most one U_i such that $C \subseteq N^{2d}(U_i)$, otherwise we would be able to apply Rule 2.

Suppose first that there is such a set U_i. By Rule 6, there is only one $u \in U_i$ that has neighbors in C; in particular, it has $d + 1$ neighbors. Now, every $v \in U_j$, for every $j \neq i$, has at most d neighbors in C, otherwise $C \subseteq N^{2d}(U_j)$ and Rule 2 would have been applied. Therefore, we conclude that C has at most $(d+1)+\sum_{v \in U \setminus U_i} |N(u) \cap C| \leq (d + 1) + d|U| \in \mathcal{O}(d|U|)$ vertices.

Finally, suppose that there is no U_i such that $C \subseteq N^{2d}(U_i)$. A similar analysis from the previous case can be performed: every $u \in U_i$ has at most d neighbors in C, otherwise $C \subseteq N^{2d}(U_i)$ and we conclude that C has at most $\sum_{v \in U} |N(u) \cap C| \leq d|U| \in \mathcal{O}(d|U|)$ vertices. □

We are now left only with an unbounded number of small clusters. A cluster C is *simple* if it is not ambiguous, that is, if for each $v \in C$, v has neighbors in a single U_i. Otherwise, C is ambiguous and, because of Lemma 62, there are at most $\mathcal{O}(d|U|^2)$ such clusters. As such, for a simple cluster C and a vertex $v \in C$, we denote by $U(v)$ the monochromatic set of U to which v is adjacent.

Reduction Rule 7. *If C is a simple cluster with at most $d + 1$ vertices, remove C from G.*

Proof of safeness of Rule 7. Let $G' = G - C$. Suppose G has a d-cut (A, B) and note that $A \not\subseteq C$ and $B \not\subseteq C$ since Rule 1 does not apply. This implies that $(A \setminus C, B \setminus C)$ is a valid d-cut of G'. For the converse, take a d-cut (A', B') of G', define $C_A = \{v \in C \mid U(v) \subseteq A\}$, and define C_B similarly; we claim that $(A' \cup C_A, B' \cup C_B)$ is a d-cut of G. To see that this is the case, note that each vertex of C_A (resp. C_B) has at most d edges to C_B (resp. C_A) and, since C is simple, C_A (resp. C_B) has no other edges to B' (resp. A'). □

After applying the previous rule, every cluster C not yet analyzed has size $d+2 \leq |C| \leq 2d$ which, in the case of the MATCHING CUT problem, where $d = 1$, is empty. To deal with these clusters, given a d-cut (A, B), we say that a vertex v is in its *natural assignment* if $v \cup U(v)$ is in the same side of the cut; otherwise the vertex is in its *unnatural assignment*. Similarly, a cluster is *unnaturally assigned* if it has an unnaturally assigned vertex, otherwise it is *naturally assigned*.

Observation 63. *Let $\mathcal{C}$ be the set of all simple clusters with at least $d + 2$ and no more than $2d$ vertices, and (A, B) a partition of $V(G)$. If there are $d|U| + 1$ edges uv, $v \in C \in \mathcal{C}$ and $u \in U$, such that uv is crossing the partition, then (A, B) is not a d-cut.*

Proof. Since there are $d|U| + 1$ edges crossing the partition between $\mathcal{C}$ and U, there must be at least one $u \in U$ with $d + 1$ neighbors in the other set of the partition. □

Corollary 64. *In any d-cut of G, there are at most $d|U|$ unnaturally assigned vertices.*

Our next lemma limits how many clusters in $\mathcal{C}$ relate in a similar way to U; we say that two simple clusters C_1, C_2 have the same *pattern* if they have the same size s and there is a total ordering of C_1 and another of C_2 such that, for every $i \in [s]$, $v_i^1 \in C_1$ and $v_i^2 \in C_2$ satisfy $U(v_i^1) = U(v_i^2)$. Essentially, clusters that have the same pattern have neighbors in exactly the same monochromatic sets of U and the same multiplicity in terms of how many of their vertices are adjacent to a same monochromatic set U_i. Note that the actual neighborhoods in the sets U_i's do not matter in order for two clusters to have the same pattern. Figure 21 gives an example of a maximal set of unnaturally assigned clusters; that is, any other cluster with the same pattern as the one presented must be naturally assigned, otherwise some vertex of U will violate the d-cut property.

Lemma 65. *Let $\mathcal{C}^* \subseteq \mathcal{C}$ be a subfamily of simple clusters, all with the same pattern, with $|\mathcal{C}^*| > d|U| + 1$. Let C be some cluster of $\mathcal{C}^*$, and $G' = G - C$. Then G has a d-cut if and only if G' has a d-cut.*

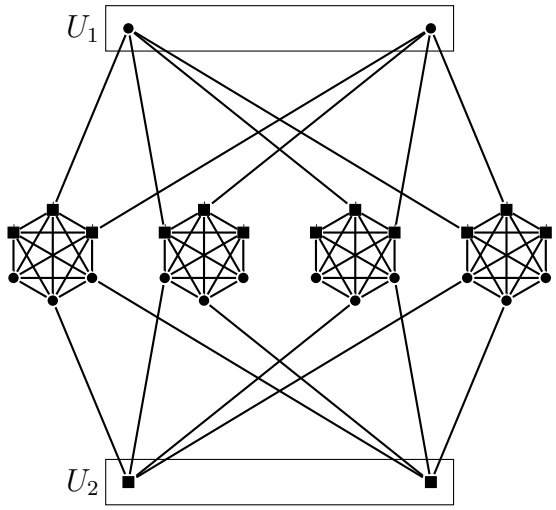

Figure 21: Example of a maximal set of unassigned clusters. Square vertices would be assigned to A, circles to B ($d = 4$).

Proof. Since by Rule 1 no subset of a small cluster is alone in a side of a partition and, consequently, U intersects both sides of the partition, if G has a d-cut, so does G'.

For the converse, let (A', B') be a d-cut of G'. First, by Corollary 64, we know that at least one of the clusters of $\mathcal{C}^* \setminus \{C\}$, say C_{n}, is naturally assigned. Since all the clusters in $\mathcal{C}^*$ have the same pattern, this guarantees that *any* of the vertices of a naturally assigned cluster cannot have more than d neighbors in the other side of the partition.

Let (A, B) be the bipartition of $V(G)$ obtained from (A', B') such that $u \in C$ is in A (resp. B) if and only if $U(u) \subseteq A$ (resp. $U(u) \subseteq B$); that is, C is naturally assigned. Define $C_A = C \cap A$ and $C_B = C \cap B$. Because $|C| = |C_{\mathsf{n}}|$ and both belong to $\mathcal{C}^*$, we know that for every $u \in C_A$, it holds that $|N(u) \cap C_B| \leq d$; moreover, note that $N(u) \cap (B \setminus C) = \emptyset$. A symmetric analysis applies to every $u \in C_B$. This implies that no vertex of C has additional neighbors in the other side of the partition outside of its own cluster and, therefore, (A, B) is a d-cut of G. □

The safeness of our last rule follows directly from Lemma 65.

Reduction Rule 8. *If there is some pattern such that the number of simple clusters with that pattern is at least $d|U| + 2$, delete all but $d|U| + 1$ of them.*

Lemma 66. *After exhaustive application of Rules 1 through 8, G has $\mathcal{O}\big(d|U|^{2d}\big)$ small clusters and $\mathcal{O}\big(d^2|U|^{2d+1}\big)$ vertices in these clusters.*

Proof. By Rule 7, no small cluster with less than $d + 2$ vertices remains in G. Now, for the remaining sizes, for each $d + 2 \leq s \leq 2d$, and each pattern of size s, by Rule 8

we know that the number of clusters with s vertices that have the same pattern is at most $d|U|+1$. Since we have at most $|U|$ possibilities for each of the s vertices of a cluster, we end up with $\mathcal{O}(|U|^s)$ possible patterns for clusters of size s. Summing all of them up, we get that we have $\mathcal{O}\big(|U|^{2d}\big)$ patterns in total, and since each one has at most $d|U|+1$ clusters of size at most $2d$, we get that we have at most $\mathcal{O}\big(d^2|U|^{2d+1}\big)$ vertices in those clusters. □

The exhaustive application of all the above rules and their accompanying lemmas are enough to show that indeed, there is a polynomial kernel for d-CUT when parameterized by distance to cluster.

Theorem 67. *When parameterized by distance to cluster* $\mathsf{dc}(G)$*,* d*-*CUT *admits a polynomial kernel with* $\mathcal{O}\big(d^2\mathsf{dc}(G)^{2d+1}\big)$ *vertices that can be computed in* $\mathcal{O}\big(d^4\mathsf{dc}(G)^{2d+1}(n+m)\big)$ *time.*

Proof. The algorithm begins by finding a set U such that $G-U$ is a cluster graph. Note that $|U| \leq 3\mathsf{dc}(G)$ since a graph is a cluster graph if and only if it has no induced path on three vertices: while there is some P_3 in G, we know that at least one its vertices must be removed, but since we don't know which one, we remove all three; thus, U can be found in $\mathcal{O}(\mathsf{dc}(G)(n+m))$ time. After the exhaustive application of Rules 1 through 8, by Lemma 62, $V(G) \setminus U$ has at most $\mathcal{O}(d^2\mathsf{dc}(G)^3)$ vertices in clusters of size at least $2d+1$. By Rule 7, G has no simple cluster of size at most $d+1$. Ambiguous clusters of size at most $2d$, again by Lemma 62, also comprise only $\mathcal{O}(d^2\mathsf{dc}(G)^2)$ vertices of G. Finally, for simple clusters of size between $d+2$ and $2d$, Lemmas 65 and 66 guarantee that there are $\mathcal{O}\big(d^2\mathsf{dc}(G)^{2d+1}\big)$ vertices in small clusters and, consequently, this many vertices in G.

As to the running time, first, computing and maintaining $N^{2d}(U_i)$ takes $\mathcal{O}(d\mathsf{dc}(G)n)$ time. Rule 1 is applied only at the beginning of the kernelization, and runs in $\mathcal{O}\big(2^{2d}d(n+m)\big)$ time. Rules 2 and 3 can both be verified in $\mathcal{O}(d\mathsf{dc}(G)^2(n+m))$ time, since we are just updating $N^{2d}(U_i)$ and performing merge operations. Both are performed only $\mathcal{O}(\mathsf{dc}(G)^2)$ times, because we only have this many pairs of monochromatic parts. The straightforward application of Rule 4 would yield a running time of $\mathcal{O}(n^2)$. However, we can ignore edges that are interior to clusters and only maintain which vertices belong together; this effectively allows us to perform this rule in $\mathcal{O}(n)$ time, which, along with its $\mathcal{O}(n)$ possible applications, yields a total running time of $\mathcal{O}(n^2)$ for this rule. Rule 5 is directly applied in $\mathcal{O}(n)$ time; indeed, all of its applications can be performed in a single pass. Rule 6 is also easily applied in $\mathcal{O}(n+m)$ time. Moreover, it is only applied $\mathcal{O}(\mathsf{dc}(G))$ times, since, by Lemma 62, the number of fixed

clusters is linear in $\mathsf{dc}(G)$; furthermore, we may be able to reapply Rule 6 directly to the resulting cluster, at no additional complexity cost. The analysis for Rule 7 follows the same argument as for Rule 5. Finally, Rule 8 is the bottleneck of our kernel, since it must check each of the possible $\mathcal{O}\big(\mathsf{dc}(G)^{2d}\big)$ patterns, spending $\mathcal{O}(n)$ time for each of them. Each pattern is only inspected once because the number of clusters in a pattern can no longer achieve the necessary bound for the rule to be applied once the excessive clusters are removed. □

In the next theorem we provide an FPTalgorithm for d-Cut parameterized by the distance to cluster, running in time $\mathcal{O}\big(4^d(d+1)^{\mathsf{dc}(G)}2^{\mathsf{dc}(G)}\mathsf{dc}(G)n^2\big)$. Our algorithm is based on dynamic programming, and is considerably simpler than the one given by Komusiewicz et al. Komusiewicz et al. [2018] for $d = 1$, which applies four reduction rules and an equivalent formulation as a 2-SAT formula. However, for $d = 1$ our algorithm is slower, namely $\mathcal{O}^*\big(4^{\mathsf{dc}(G)}\big)$ compared to $\mathcal{O}^*\big(2^{\mathsf{dc}(G)}\big)$.

Recall that minimum distance to cluster sets and minimum distance to co-cluster sets can be computed in $1.92^{\mathsf{dc}(G)}\cdot\mathcal{O}(n^2)$ time and $1.92^{\mathsf{d\bar{c}}(G)}\cdot\mathcal{O}(n^2)$ time, respectively Boral et al. [2016]. Thus, in Theorems 68 and 69 we can safely assume that we have these sets at hand.

Theorem 68. *For every integer $d \geq 1$, there is an algorithm that solves d-Cut in time $\mathcal{O}\big(4^d(d+1)^{\mathsf{dc}(G)}2^{\mathsf{dc}(G)}\mathsf{dc}(G)n^2\big)$.*

Proof. Let U be a set such that $G - U$ is a cluster graph, $\mathcal{Q} = \{Q_1, \ldots, Q_p\}$ be the family of clusters of $G - U$ and $\mathcal{Q}_i = \bigcup_{i \leq j \leq p} Q_j$. Essentially, the following dynamic programming algorithm attempts to extend a given partition of U in all possible ways by partitioning clusters, one at a time, while only keeping track of the degrees of vertices that belong to U. Recall that we do not need to keep track of the degrees of the cluster vertices precisely because $G - U$ has no edge between clusters.

Formally, given a partition $U = A \,\dot\cup\, B$, our table is a mapping $f : [p] \times \mathbb{Z}^{|A|} \times \mathbb{Z}^{|B|} \to \{0, 1\}$. Each entry is indexed by $(i, \boldsymbol{d}_A, \boldsymbol{d}_B)$, where $i \in [p]$, $\boldsymbol{d}_A$ is a $|A|$-dimensional vector with the j-th coordinate begin denoted by $\boldsymbol{d}_A[j]$; $\boldsymbol{d}_B$ is defined analogously. Our goal is to have $f(i, \boldsymbol{d}_A, \boldsymbol{d}_B) = 1$ if and only if there is a partition (X, Y) of $U \cup \mathcal{Q}_i$ where $A \subseteq X$, $B \subseteq Y$ and $v_j \in A$ ($u_\ell \in B$) has at most $\boldsymbol{d}_A[j]$ ($\boldsymbol{d}_B[\ell]$) neighbors in $\mathcal{Q}_i$.

We denote by $P_d(i, \boldsymbol{d}_A, \boldsymbol{d}_B)$ the set of all partitions $L \,\dot\cup\, R = Q_i$ such that every vertex $v \in L$ has $d_{B\cup R}(v) \leq d$, every $u \in R$ has $d_{A\cup L}(u) \leq d$, every $v_j \in A$, $d_R(v_j) \leq \boldsymbol{d}_A[j]$ and every $u_\ell \in B$, $d_L(u_\ell) \leq \boldsymbol{d}_B[\ell]$; note that, due to this definition, $(L, R) \neq (R, L)$. In the following equations, which give the computations required to build our

table, $\boldsymbol{d}_A(R)$ and $\boldsymbol{d}_B(L)$ are the updated values of the vertices of A and B after R is added to Y and L to X, respectively.

$$f(i, \boldsymbol{d}_A, \boldsymbol{d}_B) = 0 \bigvee_{(L,R)\in P_d(i,\boldsymbol{d}_A,\boldsymbol{d}_B)} f(i+1, \boldsymbol{d}_A(R), \boldsymbol{d}_B(L)) \tag{3.1}$$

$$f(p, \boldsymbol{d}_A, \boldsymbol{d}_B) = 1, \text{if and only if } P_d(i, \boldsymbol{d}_A, \boldsymbol{d}_B) \neq \emptyset. \tag{3.2}$$

We proceed to show the correctness of the above by induction. For the base case, i.e., when $|\mathcal{Q}| = p = 1$, we have that for $v_j \in A$ ($u_l \in B$), $\boldsymbol{d}_A[j] = d - d_B(v_j)$ ($\boldsymbol{d}_B[j] = d - d_A(u_l)$) and a partition of $V(G)$ exists if and only if there is some partition $(L, R) \in P_d(1, \boldsymbol{d}_A, \boldsymbol{d}_B)$, where. This case is covered by equation (3.2).

So let $p > 1$ and $(i, \boldsymbol{d}_A, \boldsymbol{d}_B)$ be an entry of our table. First, if $|Q_i| \geq 2d + 1$, Q_i is monochromatic, which implies that $|P_d(i, \boldsymbol{d}_A, \boldsymbol{d}_B)| \leq 2$. Therefore, we may assume that, $|P_d(i, \boldsymbol{d}_A, \boldsymbol{d}_B)| \leq 2^{2d}$. $P_d(i, \boldsymbol{d}_A, \boldsymbol{d}_B) = \emptyset$ implies that any partition (L, R) of Q_i causes a vertex in L (R) to have more than d neighbors in $B \cup R$ ($A \cup L$), which is easily checked for in $\mathcal{O}(n|U|)$-time, or some vertex $v_j \in A$ ($u_l \in B$) has $d_{Y\cup R}(v_j) > \boldsymbol{d}_A[j]$ ($d_{X\cup L}(u_l) > \boldsymbol{d}_B[l]$). Either way, we have that no matter how we partition $\mathcal{Q}_i$, the available degree of some vertex is not enough, equation (3.1) yields the correct answer.

However, if $P_d(i, \boldsymbol{d}_A, \boldsymbol{d}_B) \neq \emptyset$, the subgraph induced by $U \cup \mathcal{Q}_i$ has a d-cut separating A and B and respecting the limits of $\boldsymbol{d}_A$ and $\boldsymbol{d}_B$ if and only if there is some $(L, R) \in P_d(i, \boldsymbol{d}_A, \boldsymbol{d}_B)$ such that $U \cup \mathcal{Q}_{i+1}$ has a d-cut and each vertex of A (B) has the size of its neighborhood in $\mathcal{Q}_{i+1}$ bounded by the respective coordinate of $\boldsymbol{d}_A(R)$ ($\boldsymbol{d}_B(L)$). By the inductive hypothesis, there is such a partition of $\mathcal{Q}_{i+1}$ if and only if $f(i + 1, \boldsymbol{d}_A(R), \boldsymbol{d}_B(L)) = 1$, concluding the proof of correctness. Clearly, there is a d-cut separating A and B if $f(1, \boldsymbol{d}_A, \boldsymbol{d}_B) = 1$ where for every $v_j \in A$ ($u_l \in B$), $\boldsymbol{d}_A[j] = d - d_B(v_j)$ ($\boldsymbol{d}_B[j] = d - \mathsf{deg}_A(u_\ell)$).

The complexity analysis is straightforward. Recalling that $|P_d(i, \boldsymbol{d}_A, \boldsymbol{d}_B)| \leq 2^{2d}$, we have that each $f(i, \boldsymbol{d}_A, \boldsymbol{d}_B)$ can be computed in time $\mathcal{O}\big(4^d|U|n\big)$ and, since we have $\mathcal{O}\big((d+1)^{|A|+|B|}p\big) \in \mathcal{O}\big((d+1)^{|U|}p\big)$, given a partition (A, B) of U, we can decide if there is d-cut separating A and B in $\mathcal{O}\big(4^d(d+1)^{|U|}|U|n^2\big)$-time. To solve d-CUT itself, we guess all $2^{|U|}$ partitions of U and, since $|U| \in \mathcal{O}(\mathsf{dc}(G))$, we obtain a total running time of $\mathcal{O}\big(4^d(d+1)^{\mathsf{dc}(G)}2^{\mathsf{dc}(G)}\mathsf{dc}(G)n^2\big)$. □

3.3.4 Distance to co-cluster

A graph is a *co-cluster* graph if only if its the complement of a cluster graph; that is, if it is a complete multipartite graph. Our next theorem complements the results

of our previous section and shall help establish the membership in FPT of d-Cut parameterized by the vertex cover number.

Theorem 69. *For every integer $d \geq 1$, there is an algorithm solving d-Cut in time $\mathcal{O}\big(32^d 2^{\mathsf{dc}(G)}(d+1)^{\mathsf{dc}(G)+d}(\mathsf{dc}(G)+d)n^2\big)$.*

Proof. Let $U \subseteq V(G)$ be a set of $\mathcal{O}(\mathsf{dc}(G))$ vertices such that $G - U$ is a co-cluster graph with color classes $\varphi = \{F_1, \ldots, F_t\}$. Define $\mathcal{F} = \bigcup_{i \in [t]} F_i$ and suppose we are given a d-cut (A, B) of $G[U]$. First, note that if $t \geq 2d+1$, we have that some of the vertices of $\mathcal{F}$ form a clique of size $2d+1$, which is a monochromatic set; furthermore, every vertex $v \in \mathcal{F}$ but not in Q has at least $d+1$ neighbors in Q. This implies that $Q \cup \{v\}$ is monochromatic and, thus, $\mathcal{F}$ is a monochromatic set. Checking if either $(A \cup \mathcal{F}, B)$ or $(A, B \cup \mathcal{F})$ is a d-cut can be done in $\mathcal{O}(n^2)$ time.

If the above does not apply, we have that $t \leq 2d$.

- Case 1: If $|\mathcal{F}| \leq 4d$ we can just try to extend (A, B) with each of the $2^{|\mathcal{F}|}$ bipartitions of $\mathcal{F}$ in $\mathcal{O}\big(16^d n^2\big)$ time.

So now, let $\varphi_1 \dot\cup \varphi_2 = \varphi$ be a bipartition of the color classes, $\mathcal{F}_i = \{v \in F_j \mid F_j \in \varphi_i\}$, and, for simplicity, suppose that $|\mathcal{F}_1| \leq |\mathcal{F}_2|$.

- Case 2: If $|\mathcal{F}_1| \geq d+1$ and $|\mathcal{F}_2| \geq 2d+1$, we know that there is a set $Q \subseteq \mathcal{F}$ forming a (not necessarily induced) complete bipartite subgraph $K_{d+1,2d+1}$, which is a monochromatic set. Again, any $v \notin Q$ has at least $d_Q(v) \geq d+1$, from which we conclude that $Q \cup \{v\}$ is also monochromatic, implying that $\mathcal{F}$ is monochromatic.

If Case 2 is not applicable, either $|\mathcal{F}_1| \leq d$ and $|\mathcal{F}_2| \geq 2d+1$, or $|\mathcal{F}_2| \leq 2d$. For the latter, note that this implies $|\mathcal{F}| \leq 4d$, which would have been solved by Case 1. For the former, two cases remain:

- Case 3: Every $F_i \in \varphi_2$ has $|F_i| \leq 2d$. This implies that every $F \in \varphi$ has size bounded by $2d$ and that $|\mathcal{F}| \leq 4d^2$; we can simply try to extend (A, B) with each of the $\mathcal{O}\Big(2^{d^2}\Big)$ partitions of $\mathcal{F}$, which can be done in $\mathcal{O}\Big(2^{d^2} n^2\Big)$ time.
- Case 4: There is some $F_i \in \varphi_2$ with $|F_i| \geq 2d+1$. Its existence implies that $|\mathcal{F}| - |F_i| \leq d$, otherwise we would have concluded that $\mathcal{F}$ is a monochromatic set. Since $\mathcal{F} \setminus F_i$ has at most d vertices, the set of its bipartitions has size bounded by 2^d. So, given a bipartition $\mathcal{F}_A \dot\cup \mathcal{F}_B = \mathcal{F} \setminus F$, we define $A' := A \cup \mathcal{F}_A$ and $B' := B \cup \mathcal{F}_B$. Finally, note that $G \setminus (A' \cup B')$ is a cluster graph where every

cluster is a single vertex; that is, $\mathsf{dc}(G) \leq \mathsf{d\overline{c}}(G) + d$. In this case, we can apply Theorem 68, and obtain the running time of $\mathcal{O}\big(4^d(d+1)^{\mathsf{d\overline{c}}(G)+d}(\mathsf{d\overline{c}}(G)+d)n^2\big)$; we omit the term $2^{\mathsf{d\overline{c}}(G)+d}$ since we already have an initial partial d-cut (A', B').

For the total complexity of the algorithm, we begin by guessing the initial partition of U into (A, B), spending $\mathcal{O}(n^2)$ time for each of the $\mathcal{O}\big(2^{\mathsf{d\overline{c}}(G)}\big)$ possible bipartitions. If $t \geq 2d+1$ we give the answer in $\mathcal{O}(n^2)$ time. Otherwise, $t \leq 2d$. If $|\mathcal{F}| \leq 4d$, then we spend $\mathcal{O}\big(16^d n^2\big)$ time to test all partitions of $\mathcal{F}$ and return the answer. Else, for each of the $\mathcal{O}\big(4^d\big)$ partitions of φ, if one of them has a part with $d+1$ vertices and the other part has $2d+1$ vertices, we respond in $\mathcal{O}(n^2)$ time. Finally, for the last two cases, we either need $\mathcal{O}\Big(2^{d^2}n^2\Big)$ time, or $\mathcal{O}\big(8^d(d+1)^{\mathsf{d\overline{c}}(G)+d}(\mathsf{d\overline{c}}(G)+d)n^2\big)$. This yields a final complexity of $\mathcal{O}\big(32^d 2^{\mathsf{d\overline{c}}(G)}(d+1)^{\mathsf{d\overline{c}}(G)+d}(\mathsf{d\overline{c}}(G)+d)n^2\big)$. □

Using Theorems 68, 69, and the relation $\tau(G) \geq \max\{\mathsf{dc}(G), \mathsf{d\overline{c}}(G)\}$ Komusiewicz et al. [2018], we obtain fixed-parameter tractability for the vertex cover number $\tau(G)$.

Corollary 70. *For every $d \geq 1$, d-CUT parameterized by the vertex cover number is in* FPT.

3.4 Other Generalizations for Matching Cut

In this section, we describe two other generalizations of MATCHING CUT that we have investigated. For the first, ℓ-NESTED MATCHING CUT, we present an exponential time algorithm and an attempt at using algorithms for MATCHING CUT as a black box for this problem. Most results given for d-CUT and MATCHING CUT can be adapted for this problem, but the arguments are very similar and do not appear to provide additional insights on the problem. The second problem we discuss here has been dubbed as p-WAY MATCHING CUT. Unlike d-CUT and ℓ-NESTED MATCHING CUT, this version is much more challenging, and we limit ourselves to some attempts on tackling the problem.

3.4.1 Nested Cuts

We have already discussed d-CUT at length throughout this chapter, so this section will detail some other directions we attempted to explore. Recall the definition of a matching cut: we would like each vertex of the graph, on the cut (A, B), to have at most one neighbor across the cut. This can be rephrased to the following: a cut (A, B) is a matching cut if and only if each vertex has at most one neighbor *outside* of its

part. Through this perspective, there is nothing special about the number of parts we want to partition our graph into. A cut on ℓ parts satisfying the above is called an *ℓ-nested matching cut*, and the decision problem for this generalization is dubbed the ℓ-Nested Matching Cut problem.

ℓ-Nested Matching Cut
Instance: A graph G.
Question: Does G admit an ℓ-nested matching cut?

Let $\varphi = (A_1, \ldots, A_\ell)$ be a partition of $V(G)$ and $\mathsf{border}(A_i)$ be the vertices of A_i with one neighbor outside of A_i. The following observation gives some intuition as to the structure of the positive instances of ℓ-Nested Matching Cut.

Observation 71. *Let G be a graph and $\ell \geq 3$ an integer. G admits an ℓ-nested matching cut φ if and only if there is an $(\ell-1)$-nested matching cut $\varphi' = \{A'_1, \ldots, A'_{\ell-1})\}$ with one A'_i such that the subgraph induced by the vertices in A'_i admits a matching cut (B_1, B_2) where, for every $v \in \mathsf{border}(A'_i)$, it holds that $N[v] \subseteq B_1$ or $N[v] \subseteq B_2$.*

Proof. We shall build φ' from φ as follows: for every $i \in [\ell-2]$, $A'_i = A_i$, and $\varphi'_{\ell-1} = \varphi_{\ell-1} \cup \varphi_\ell$. That φ' is an $(\ell-1)$-nested matching cut of G and the subgraph induced by the vertices of $\varphi'_{\ell-1}$ has a matching cut is a straightforward observation. Now, for each $v \in \mathsf{border}(\varphi'_{\ell-1})$, note that $v \in \mathsf{border}(\varphi_{\ell-1}) \cup \mathsf{border}(\varphi_\ell)$. Consequently, every neighbor of v is in either the same side of the cut $(\varphi_{\ell-1}, \varphi_\ell)$ as v, as we wanted.

For the converse, it suffices to note that $\mathsf{border}(A') \cap \mathsf{border}(\varphi'_i) = \emptyset$ and $\mathsf{border}(B') \cap \mathsf{border}(\varphi'_i) = \emptyset$ precisely because of the constraint that $N[v] \subseteq A'$ or $N[v] \subseteq B'$. As such, we can construct the desired ℓ-nested matching cut as $\varphi = \{\varphi'_1, \ldots, \varphi'_{\ell-2}, A', B'\}$. □

Observation 71 is a first step towards an algorithm for ℓ-Nested Matching Cut. Ideally, we would use the algorithms for Matching Cut as a black box, and then choose one of the available parts of the cut and repeat the process. What is problematic is that there may be multiple possible matching cuts at a given step, and testing all of them would be quite expensive. As such, since we still do not know how to exploit Observation 71, we turn our attention to an exact exponential algorithm, through a similar approach used by Komusiewicz et al. [2018]. Our algorithm consists of four stopping rules, seven reduction and nine branching rules. At every step of the algorithm we have the sets $\{A_1, \ldots, A_\ell, F\}$ such that $\varphi = (A_1, \ldots, A_\ell)$ (unless rule R3 applies) is a ℓ-nested matching cut of the vertices of $V(G) \setminus F$. For simplicity, we assume that $\delta(G) \geq 2$. Most of the arguments presented here work with slight modifications to

graphs of minimum degree one, but they would unnecessarily complicate the description of the algorithm.

S1 If there is some $v \in F$ and $i, j \in [\ell]$ such that $\deg_{A_i}(v) \geq 2$ and $\deg_{A_j}(v) \geq 2$, STOP: there is no ℓ-nested matching cut extending φ.

S2 If there is a vertex $v \in F$ with neighbors in three different parts of φ, STOP: there is no ℓ-nested matching cut extending φ.

S3 If there is an edge uv with $u \in A_i$ and $v \in A_j$ such that $N(u) \cap N(v) \cap F \neq \emptyset$, STOP: there is no ℓ-nested matching cut extending φ.

S4 If there is some $v \in A_i$ with two neighbors outside of $A_i \cup F$, STOP: there is no ℓ-nested matching cut extending φ.

R1 If there exists some $v \in A_i$ such that $N(v) \supseteq \{x, y\}$ and $x, y \in F$ and $xy \in E(G)$, add x, y to A_i.

R2 If there exists $v \in F$ and an unique $i \in [\ell]$ with $\deg_{A_i}(v) \geq 2$, add v to A_i.

R3 For every edge uv with $u \in A_i$ and $v \in A_j$, add $N(u) \cap F$ to A_i and $N(v) \cap F$ to A_j.

R4 If there is a pair $u, v \in F$ with $N(u) = N(v) = \{x, y\}$ with $x \in A_i$ and $y \in A_j$, add u to A_i and v to A_j.

R5 If there is a pair $u, v \in F$ with $N(u) = N(v) = \{x, y\}$ with $x \in A_i$ and $y \in F$, add x to A_i.

R6 If there is a vertex $v \in F$ with $N(v) = \{x, y\}$, $x \in A_i$, $y \in A_j$, $N(x) \subseteq A_i \cup \{v\}$, and $N(y) \subseteq A_j \cup \{v\}$, add v to A_i.

R7 If there are vertices $u, v, w \in F$ with $\deg(u) = \deg(v) = \deg(w)$ arranged as in Figure 22, add $\{u, v\}$ to A_i and w to A_j.

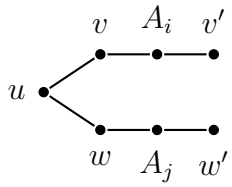

Figure 22: Rule 7 configuration.

For our branching rules, we follow the configurations given by Figure 23, and always branch on vertex v_1. We set the size of the instance as the size of the set F, that is, how many free vertices are assigned to one of the parts.

B1 If we put v_1 in A_j, $j \neq i$, we infer that v_3, and v_4 must also be added to A_j, and that v_2 must be added to A_i. Otherwise, v_1 is in A_i, which does not given us any additional information. Our branching vector is, thus, of the form $\{1\} \times \{4\}^{\ell-1}$.

B2 Note that v_1 must be placed in either A_i or A_j. For the first case, we conclude that v_4 must also be in A_i, while for the later, v_4 must be added to A_j and v_2 to A_i, yielding the branching vector $(2, 3)$ and the branching factor 1.3247.

B3 Again, v_1 is in either A_i or A_j. In either case, we conclude that v_2 must be in the same part as v_1, resulting in the branching vector $(2, 2)$, which has a branching factor of $\sqrt{2}$.

B4 and B4' By adding v_1 to A_i, we conclude that v_2 must also be placed in A_i; a similar analysis is performed when v_1 is added to A_j. Otherwise, if we add v_1 to A_k, at most one of v_2 and v_3 may be added to a set different from A_k. If both are in A_k, we have that v'_2 belongs in A_i and v'_3 in A_j. Otherwise, if v_2 is added to A_i, we conclude that v_3, v_4 belong in A_k and that v'_3 belongs in A_j; similarly if v_3 is assigned to A_j. This results in a branching vector of the form $\{2\}^2 \times \{5\}^{3\ell-6}$, with unique positive real root of the polynomial associated with it satisfying $\alpha_\ell \leq \sqrt[3]{\ell}$. Rule B4' clearly has a better branching factor than B4, but rule B4 dominates the running time of B4'.

B5 In case we assign v_1 to A_i, we have that both v_2, and v_3 must also be in A_i; if v_1 is assigned to A_j, nothing else can be inferred; for all other A_k, we have that both v_2 and v_3 must be assigned to A_k, and that v'_2 and v'_3 belong in A_i. The branching vector for this rule is given by $\{1\} \times \{3\} \times \{5\}^{\ell-2}$, which, for large values of ℓ, has a branching factor of at most $\sqrt[3]{\ell}$. Again, it can be verified that, for each ℓ, it holds that the branching factor for this rule is $\leq \sqrt[3]{\ell}$.

B6 If v_1 is assigned to A_i (resp. A_j), we have that both v_2, and v_3 (resp. v_4, and v_5) must also be assigned to A_i (resp. A_j); otherwise, for every other A_k, either $\{v_p\}_{p\in[5]}$ belongs to A_k, in which case the vertices $\{v'_p\}_{p\in\{2,3,4,5\}}$ are assigned to the same set as their neighbor, or at most one $v_p \in \{v_2, v_3, v_4, v_5\}$ is not assigned to A_k, in which case the set to which v'_p should be assigned is not determined. This rule produces a branching vector of the form $\{3\}^2 \times \{8\}^{4\ell-8} \times \{9\}^{\ell-2}$,

B7 Once again, we only have two options for v_1. So, if v_1 is added to A_i, we have that v_3 must be added to A_j, otherwise v_1 is added to A_j and v_2 to A_i. This rule's branching vector is $(2, 2)$, with factor equal to $\sqrt{2}$.

B8 If v_1 is assigned to A_i, we are done; otherwise, if v_1 is assigned to A_k, with $k \neq i$, we have that v_2 belongs in A_i and v_3 in A_i. This yields the branching vector $\{1\} \times \{3\}^{\ell-1}$, and branching factor $\sqrt[3]{\ell} \leq \alpha_\ell \leq \sqrt{\ell}$.

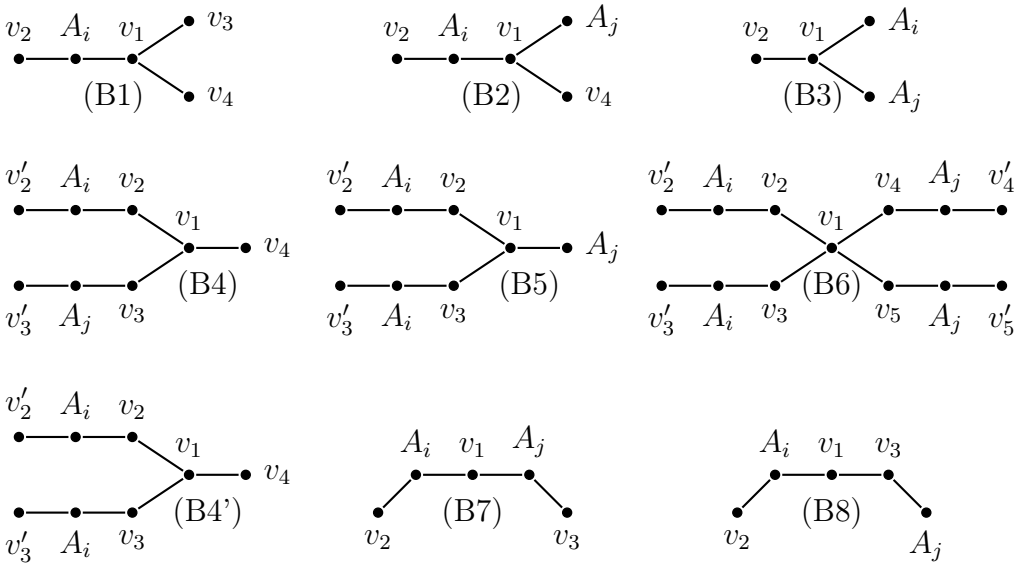

Figure 23: Branching configurations for ℓ-NESTED MATCHING CUT.

Given all of the above rules, we must show that, if none of them are applicable, we have an ℓ-nested matching cut. In order to do so, we require some additional definitions: let $A'_i = \{v \in A_i \mid \deg_F(v) \geq 2\}$, $F'_i = F \cap N(A'_i)$, $F''_i = \{v \in F \mid \deg_{F'_i}(v) \geq 2\}$, and $F^* = F \setminus \bigcup_{i \in \ell}(F'_i \cup F''_i)$. Also, we say that A_i is *final* if, for all $v \in F'_i$, $\deg(v) = 2$.

Lemma 72. *If there is some A_i of φ which is not final and no Stopping/ Reduction Rule is applicable, then configurations B1, or B2 exist in the partitioned graph.*

Proof. Let v_1 be a degree three vertex of F'_i, a_i its neighbor in A'_i and v_2 the other neighbor of A'_i in F. We know that $vv' \notin E(G)$, otherwise rule R1 would be applicable. Now, let v_3, v_4 be two of the other neighbors of v. If both are in F, we have a configuration B1; otherwise at most one of them is not in $F \cup A_i$, say v_3, since we would have applied rule S2 or rule R2 if this observation did not hold, implying that a configuration B2 is present. □

We may now assume that every A_i is final, and that no reduction or stopping rule is applicable. Our goal is to show that, if none of our branching configurations

exist, then $\varphi^* = (A_1 \cup F'_1 \cup F''_1, \ldots, A_{\ell-1} \cup F'_{\ell-1} \cup F''_{\ell-1}, A_\ell \cup F'_\ell \cup F''_\ell \cup F^*)$ is an ℓ-nested matching cut of G. Before proving that, however, we have to guarantee that the sets F'_i, F''_i are a partition of F.

Lemma 73. *If there exists $i, j \in [\ell]$ with $F'_i \cap F'_j \neq \emptyset$, then rule B7 is applicable.*

Proof. Let $v_1 \in F'_i \cap F'_j$; since A'_i and A'_j are final, $\deg(v_1) = 2$ and its two neighbors, a_i, a_j, have one extra neighbor each, say v_2 and v_3. If $v_2 = v_3$, however, $v_2 \in F'_i$, and has degree equal to two; but this implies that $N(v_1) = N(v_2) = \{a_i, a_j\}$, and rule R4 could have been applied. All that remains now is the case where $v_2 \neq v_3$, but this is precisely configuration B7, as desired. □

Lemma 74. *If A_i and A_j are final, $F'_i \cap F''_j = \emptyset$.*

Proof. Suppose that there is some $v \in F'_i \cap F''_j$. By the definitions of F'_i and F''_j, v has degree two, one neighbor in A_i, and *two* neighbors in F'_j, a contradiction. □

Lemma 75. *If there exists $i, j \in [\ell]$ with $F''_i \cap F''_j \neq \emptyset$, rule B6 is applicable.*

Proof. Let v_1 a vertex of $F''_i \cap F''_j$. By the previous lemma and the definition of F''_i, it is straightforward to check that v_1 has four distinct neighbors: v_2, v_3, v_4, v_5, such that $v_2, v_3 \subseteq F'_i$ and $v_4, v_5 \subseteq F'_j$. Let a_i be the neighbor of v_2 in A_i, v'_2 the other neighbor of a_i in F. Define a'_i and v'_3 similarly for v_3; a_j and v'_4 for v_4; and a'_j and v'_5 for v_5. Note that $v'_2 \neq v'_3$ (resp. $v'_4 \neq v'_5$), or rule R2 would be applicable. Consequently, $\{v_1, v_2, a_i, v'_2, v_3, a'_i, v'_3, v_4, a_j, v'_4, v_5, a'_j, v'_5\}$ form a configuration B6. □

These last few results prove that φ^* is a partition of $V(G)$. Define $A^*_i = A_i \cup F'_i \cup F''_i$ and $A^*_\ell = A_\ell \cup F'_\ell \cup F''_\ell \cup F^*$. What remains to be shown is that it is, in fact, an ℓ-nested matching cut of G.

Lemma 76. *If no more branching rules are applicable, then for every i and every $v \in A_i$, $\deg_{V(G)\setminus A^*_i}(v) \leq 1$.*

Proof. First, v has at most one neighbor in $\bigcup_{j \neq i} A_j$, otherwise rule S4 would have stopped the algorithm. The case where v has one neighbor in A_j and one neighbor $u \in F$, since rule S3 is not applicable, by rule R3, u must have been added to A_i, and so u does not exist. Thus, the only possibility is that v has more than one neighbor in F, implying $N_F(v) \subseteq F'_i$, but F'_i is in the same part as v. □

Lemma 77. *If no more branching rules are applicable, then for every i and every $v \in F'_i$, $\deg_{V(G)\setminus A^*_i}(v) \leq 1$.*

Proof. Trivial due to the hypothesis that F'_i is final. □

Lemma 78. *If no more branching rules are applicable, then for every i and every $v \in F''_i$, $\deg_{V(G)\setminus A^*_i}(v) \leq 1$.*

Proof. If v has a neighbor in A_j, rule B5 is applicable, since $v \in F''_i$. On the other hand, if v has a neighbor in F, we can apply rule B4' with $v = v_1$. □

Lemma 79. *If no more branching rules are applicable, then for every $v \in F^*$, $\deg_{V(G)\setminus A^*_\ell}(v) \leq 1$.*

Proof. We know that v does not have two neighbors in some A_j, but it could be the case that v has neighbors $a_i \in A_i$, $a_j \in A_j$. Note that if v a_i cannot have a second neighbor in F, otherwise v would be in F'_i. As such, if $\deg(v) = 2$, we can still apply rule R6. Otherwise, if $\deg(v) \geq 3$, configuration B3 shows up with $v = v_1$. This allows us to conclude that v does not have a neighbor in more than one A_i. Suppose now that $u \in F \setminus F^*$ is a neighbor of v, and $a_j \in A_j \cap N(v)$. If $u \in F'_i$ (i may be equal to j), it follows that rule B8 is applicable with $v = v_3$ and $u = v_1$. If, on the other hand, $u \in F''_i$, we have configuration B4' where $u = v_1$ and $v = v_4$. Consequently, v has no neighbor in A_j, for $j \neq \ell$.

Now, it must be the case that both neighbors x, y of v are in $F \setminus F^*$. Note that $\{x, y\} \not\subseteq F'_i$, otherwise $v \in F''_i$. For now, suppose that $x \in F'_i$ and $y \in F'_j$. If $\deg(v) = 2$, rule R7 may be applied (with $u = v$); so $\deg(v) \geq 3$ and we have configuration B4, again, with $v = v_1$. Suppose, then that x is actually in F''_i; by the exact same argument, it holds that B4' is applicable with $v = v_4$ and $x = v_1$. The case where x and y are in F''_i and F''_j, respectively, is identical. □

Theorem 80. *If no Stopping, Reduction, or Branching rule is applicable, φ^* is an ℓ-nested matching cut of G. Moreover, ℓ-*NESTED MATCHING CUT *can be solved in $\mathcal{O}^*(\alpha_\ell^n)$, with $\alpha_\ell \leq \sqrt{\ell}$ for a graph on n vertices.*

3.4.2 Multiway Cuts

The previous section dealt with partitions φ such that each vertex has at most one neighbor in another part. We may relax this constraint, and ask that each vertex has at most one neighbor in *each part other than its own.* Equivalently, given the integer $p \geq 2$, we want an p-partition of the vertices of the graph such that (A_i, A_j) is a matching cut of $G[A_i \cup A_j]$. A cut that satisfies this property is called a *p-Way matching cut*, with the problem of deciding whether or not a graph admits such a partition defined below.

p-Way Matching Cut
Instance: A graph G
Question: Does G admit a p-Way matching cut?

It is not hard to adapt either of the reductions given by Chvátal [1984] or us to this generalization, although a bit more of care must be taken when designing color selection gadgets. The hard part, however, is finding an FPT algorithm for p-Way Matching Cut when parameterized by the number of edges crossing the cut. The powerful machinery provided by Marx et al. [2010] does not appear to be capable of handling the constraint of each pair of parts is a matching cut. Whenever we attempted to give a graph class that captured this notion, we were either unable to reconstruct the cut on the original graph, or we found counter-examples that the treewidth reduction technique could produce that did not represent a p-way matching cut of the graph. Adapting the exact exponential algorithm of Komusiewicz et al. [2018] also seems a challenging task; while we were successful for nested cuts, the structures used as branching rules appear to explode rapidly with the growth of p, a similar phenomenon is observed when trying to devise a kernelization algorithm Aside from these challenging questions, most parameterized algorithms for Matching Cut can be adapted for p-Way Matching Cut, such as the ones parameterized by treewidth or distance to cluster, without much difficulty, but, much like d-Cut all such algorithms have exponential dependencies on p, and asserting whether this is necessary or not would be nice.

3.5 Concluding remarks

We presented a series of algorithms and complexity results; many questions, however, remain open. For instance, all of our algorithms have an exponential dependency on d on their running times. While we believe that such a dependency is an intrinsic property of d-cut, we have no proof for this claim. Similarly, the existence of a *uniform* polynomial kernel parameterized by the distance to cluster, i.e., a kernel whose degree does not depend on d, remains an interesting open question.

Also in terms of running time, we expect the constants in the base of the exact exponential algorithm to be improvable. However, exploring small structures that yield non-marginal gains as branching rules, as done by Komusiewicz et al. Komusiewicz et al. [2018] for $d = 1$ does not seem a viable approach, as the number of such structures appears to rapidly grow along with d.

The distance to cluster kernel is hindered by the existence of clusters of size between $d+2$ and $2d$, an obstacle that is not present in the MATCHING CUT problem. Aside from the extremal argument presented, we know of no way of dealing with them. We conjecture that it should be possible to reduce the total kernel size from $\mathcal{O}\big(d^2\mathsf{dc}(G)^{2d+1}\big)$ to $\mathcal{O}\big(d^2\mathsf{dc}(G)^{2d}\big)$, matching the size of the smallest known kernel for MATCHING CUT Komusiewicz et al. [2018].

We also leave open to close the gap between the known polynomial and NP-hard cases in terms of maximum degree. We showed that, if $\Delta(G) \leq d+2$ the problem is easily solvable in polynomial time, while for graphs with $\Delta(G) \geq 2d+2$, it is NP-hard. But what about the gap $d+3 \leq \Delta(G) \leq 2d+1$? After much effort, we were unable to settle any of these cases. In particular, we are very interested in 2-CUT, which has a single open case, namely when $\Delta(G) = 5$. After some weeks of computation, we found no graph with more than 18 vertices and maximum degree five that had no 2-cut, in agreement with the computational findings of Ban and Linial Ban and Linial [2016]. Interestingly, all graphs on 18 vertices without a 2-cut are either 5-regular or have a single pair of vertices of degree 4, which are actually adjacent. In both cases, the graph is maximal in the sense that we cannot add edges to it while maintaining the degree constraints. We recall the initial discussion about the INTERNAL PARTITION problem; closing the gap between the known cases for d-CUT would yield significant advancements on the former problem.

Finally, the smallest d for which G admits a d-cut may be an interesting additional parameter to be considered when more traditional parameters, such as treewidth, fail to provide FPT algorithms by themselves. Unfortunately, by Theorem 45, computing this parameter is not even in XP, but, as we have shown, it can be computed in FPT time under many different parameterizations.

Chapter 4

On the intersection graph of maximal stars

Intersection graphs form the basis for much of the theory on graph classes. For instance, the class of chordal graphs, which is one of the most fundamental and broadly studied classes Brandstädt et al. [1999], can be defined as precisely the family of intersection graphs of all subtrees of some tree. Interval graphs, in turn, are defined as the family of intersection graphs of subpaths of some path. Line graphs are the intersection graphs of the edges of some graph. Unlike chordal graphs, there are known characterizations for line graphs that make use of a finite family of forbidden induced subgraphs [Roussopoulos, 1973]. Moreover, line graphs were one of the first classes to be characterized in terms of edge clique covers that satisfy some properties pertinent to the intersection definition; results of this form are known as *Krausz-type characterizations*.

All of the aforementioned classes are easily recognizable in polynomial time Brandstädt et al. [1999]; Naor and Novick [1990]. The complexity of recognizing clique graphs – the intersection graphs of the maximal cliques of some graph – was an open problem for decades, with a very complicated argument, due to Alcón et al. [2009], showing that the problem is NP-complete. Many other aspects of clique graphs have been investigated in the literature. Such is the case for, clique-critical graphs – graphs whose clique graph is different from the clique graph of all of its proper induced subgraphs. This graph class has its own characterizations Escalante and Toft [1974] and bounds Alcón [2006] which were central in the proof of the complexity of the clique graph recognition problem. Another common line of investigation on intersection graphs is the study of iterated intersection graphs, i.e., of the behavior of a graph that undergoes the operation multiple consecutive times. Results of this flavor usually come in the form of convergence, divergence, and periodicity theorems, relating properties

of the input graph to the behavior of the limit graph (after applying the operator an infinite number of times). A closely related intersection class to star graphs is that of biclique graphs – the intersection graph of the maximal induced complete bipartite graphs of a graph. The introductory paper by Groshaus and Szwarcfiter [2010] gives a Krausz-type characterization of the class and some properties of its members; these results, however, are not very useful from the algorithmic point of view, and appear to not yield many insights on the recognition problem.

Star graphs and biclique graphs coincide for C_4-free graphs and our initial hope was that results on the former would yield advancements on the latter. While we were unable to achieve our original goal, we present an introductory study of the intersection graphs of maximal stars, providing answers to some difficulties we encountered when working with the class. After some standard definitions of the theory of intersection graphs, we begin the discussion with a bound on the number of vertices of star-critical graphs by a quadratic function of the size of its set of maximal stars. Afterwards, we give a Krausz-type characterization, which, when combined to the previous result, shows that the recognition problem belongs to NP. We then shift the focus, to properties of star graphs. In particular, we show that they are biconnected, that every edge belongs to at least one triangle, we characterize the structures that the pre-image must have in order to generate degree two vertices, and bound the diameter of the star graph with respect to the diameter of its pre-image is given. Finally, we give a monotonicity theorem, which is used to generate all star graphs on no more than eight vertices and prove that the class of star graphs and square graphs are not properly contained in each other.

4.1 Intersection Graphs

The *intersection graph* of a multifamily $\mathcal{F} \subseteq 2^S$, denoted by $G = \Omega(\mathcal{F})$ is the graph of order $|\mathcal{F}|$ and, for every $F_u, F_v \in \mathcal{F}$, $uv \in E(G) \Leftrightarrow F_u \cap F_v \neq \emptyset$. Any $\mathcal{F}$ such that $\Omega(\mathcal{F}) \simeq G$ is a *set representation* of G. A known theorem states that every graph is the intersection graph of a family of subgraphs of a graph [A. McKee and McMorris, 1999].

An *edge clique cover* $\mathcal{Q} = \{Q_1, \ldots, Q_n\}$ of a graph G is a (multi)family of cliques of G such that every edge of G is contained in at least one element of $\mathcal{Q}$. The *dual edge clique cover* of a set representation $\mathcal{F} = \{F_1, \ldots, F_n\}$, with $|\bigcup_{i \leq n} F_i| = m$, is defined as $\mathcal{Q}(\mathcal{F}) = \{Q_1, \ldots, Q_m\}$ such that $Q_j = \{i \mid j \in F_i\}$. The *dual set representation* of an edge clique cover $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$, with $|\bigcup_{j \leq m} Q_j| = n$, is $\mathcal{F}(\mathcal{Q}) = \{F_1, \ldots, F_n\}$

with $F_i = \{j \mid i \in Q_j\}$.

Some interesting intersection graphs are usually defined in terms of the intersection of structures of other graphs. For instance, *line graphs* are precisely the graphs that are the intersection graphs of the edges of a graph; *clique graphs* are the intersection graphs of the maximal induced cliques of a graph. Both of these classes, however have nice characterizations in terms of edge clique covers, which are commonly called *Krausz-type characterizations.*

Line Graph *G is a line graph if and only if there is an edge clique cover $\mathcal{Q}$ of G such that both conditions hold:*

(i) Every vertex of G appears in exactly two members of $\mathcal{Q}$;

(ii) Every edge of G is in only one member of $\mathcal{Q}$.

Clique Graph *G is a clique graph if and only if it there is an edge clique cover of G satisfying the Helly property.*

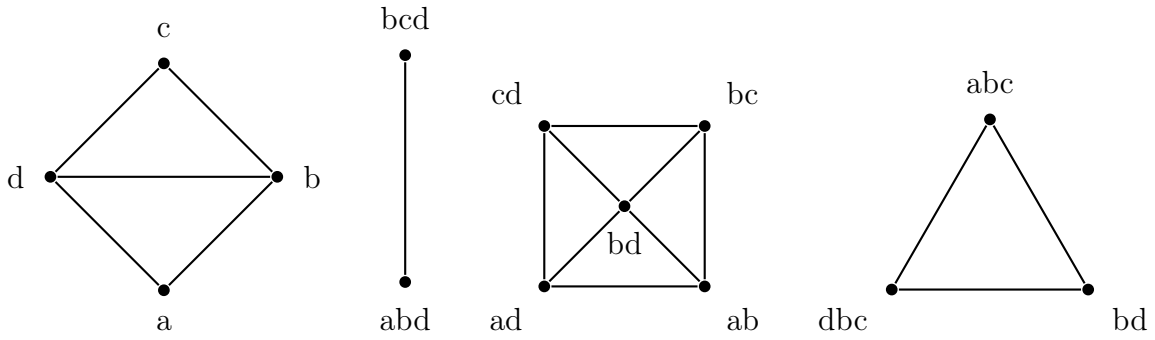

Figure 24: A graph, its clique graph, its line graph, and its star graph

The recognition of line graphs is known to be efficient [Degiorgi and Simon, 1995; Roussopoulos, 1973; Naor and Novick, 1990]. For clique graphs, however, the situation was not so simple, and the complexity of clique graph recognition was left open for several years, finally being proven to be NP-complete by Alcón et al. [2009] with a quite complicated argument.

Aside from the complexity point of view, many different properties of intersection graphs have been investigated in the literature. For instance, clique-critical graphs – graphs whose clique graph is different from the clique graph of all of its proper induced subgraphs – have different characterizations [Escalante and Toft, 1974] and bounds [Alcón, 2006] which were crucial in the proof of the complexity of the recognition problem. Another common line of investigation on intersection graphs is the behaviour

of iterated applications of the operators. For instance, Frías-Armenta et al. [2004], and Larrión and Neumann-Lara [2002] study iterated applications of the clique operator. Biclique graphs – the intersection graph of the maximal induced complete bipartite graphs of a graph – were first characterized and studied by Groshaus and Szwarcfiter [2010]. Their results, however, are not very useful from the algorithmic point of view, and appear to not yield many insights on the recognition problem. Nevertheless, they study the behavior of biclique graphs, showing that every edge is contained either in a diamond or a 3-fan and specialize their general characterization for biclique graphs of bipartite graphs. As was done for clique graphs, the iterated biclique operator has also been studied by Groshaus et al. in multiple papers [Groshaus and Montero, 2013; Groshaus et al., 2016], with results ranging from characterizations of divergence, divergence type verification algorithms, and other structural results.

For other classical results in the area we point to [A. McKee and McMorris, 1999], from where most of the given definitions come from.

4.1.1 Maximal Stars

Regarding stars, previous work handled the intersection graphs of (not necessarily maximal) substars of a tree [Joos, 2014] and of a star [Cerioli and Szwarcfiter, 2006]. For the first, a minimal infinite family of forbidden induced subgraphs was given, while, for the latter, a series of characterizations were shown (including a finite family of forbidden induced subgraphs). Stars are a particular case of bicliques, and both the biclique graph and star graph coincide for C_4-free graphs. In fact, this relationship was successfully applied to determine the complexity of biclique coloring [Groshaus et al., 2014], using a reduction from QSAT_2 to star coloring (a coloring of the vertices of a graph such that no maximal star is monochromatic). To the best of our knowledge, these are the main topics discussed in the literature that involve maximal stars in some way.

The *star operator* $K_{\mathcal{S}}$ applied on H is the intersection graph of the induced maximal stars of H. The result of $K_{\mathcal{S}}$ is called the *image graph*, or simply *image*. A graph G is a *star graph* if there is some graph H such that $G \simeq K_{\mathcal{S}}(H)$. Given a star graph G, any H such that $K_{\mathcal{S}}(H) \simeq G$ is called a *pre-image* of G. The *center* of a star $s = K_{1,n}$ is the vertex of the partition of size one. A *leaf* of a star $K_{1,n}$ is one of the vertices of degree one. We say that star s_a *absorbs* star s_b if, by removing one leaf of s_b, it becomes a substar of s_a. The *iterated star operator* $K^i_{\mathcal{S}}$ is defined as $K^1_{\mathcal{S}}(G) = K^1_{\mathcal{S}}(G)$ and $K^i_{\mathcal{S}}(G) = K_{\mathcal{S}}(K^{i-1}_{\mathcal{S}}(G))$.

A vertex v is said to be *star-critical* if its removal changes the resulting star

graph; that is, the star graph of H and the star graph of $H \setminus \{v\}$ are not isomorphic. Similarly to clique-critical graphs [Escalante and Toft, 1974; Alcón, 2006], a graph is *star-critical* if all of its vertices are star-critical. It is not hard to see that the only vertices which may be non-star-critical are simplicial vertices; for example, if there is a class of false twin simplicial vertices, all but one of them are certainly non-star-critical.

When detailing which vertices belong to a star, we shall describe it by $\{v_1\}\{v_2, \ldots, v_{n+1}\}$, with v_1 being its center and the other n vertices its leaves. If the star is a single edge, choose one of the vertices to be the center and the other to be the leaf arbitrarily. Unless noted, G will be our star graph and H the pre-image of G. The family of all maximal stars of G is denoted by $\mathcal{S}(G)$. For the entirety of this work, we assume that all of our graphs are connected.

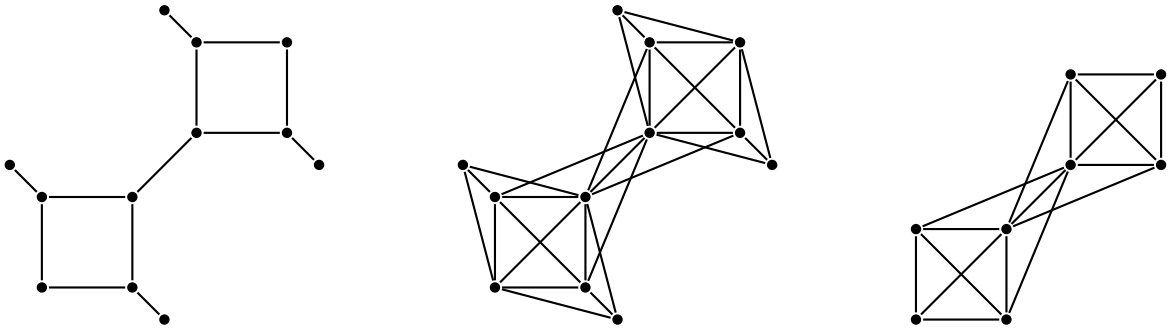

Figure 25: A triangle-free graph (left), its square (center) and its star graph (right).

Before proceeding to the main results of this chapter, we make the following remark.

Observation 81. *Every vertex of degree at least two in a triangle free graph is the center of exactly one maximal star.*

The above observation immediately leads us to the property that every star graph of a triangle-free graph is closely related to the square of one of its induced subgraphs.

Observation 82. *If H is a K_3-free graph with at least 3 vertices, D are its vertices of degree at least 2 and $G = K_{\mathcal{S}}(H)$, it holds that $G \simeq H[D]^2$.*

As such, every hardness result or polynomial time algorithm for the recognition of squares of triangle-free graphs immediately applies to the class of star graphs of triangle-free graphs. For an illustration of the previous observation, we refer to Figure 25. For a far more complicated star graph, we refer to Figure 26.

However, star graphs appear to be natural generalizations of square graphs [Bondy and Murty, 1976] in the sense that, when applying the squaring operation, for each vertex v only the largest, non-induced star centered at v is selected,

Figure 26: A graph (left) and its star graph (right).

and the intersection graph of these stars is generated. On the other hand, for star graphs, every *induced* maximal star is used in the construction of the intersection graph. Despite the classes of star graphs and biclique graphs being equivalent when restricting the pre-image domain to C_4-free graphs, we were unable to deepen the study biclique graphs; our efforts were hindered by some of the questions posed and developed upon in this work.

4.2 A bound for star-critical pre-images

Our first results shows an upper bound on the number of vertices of a star-critical graph in terms of its number of maximal stars. For an arbitrary graph H the difference $|V(H)| - |\mathcal{S}(H)|$ could be arbitrarily big, but some vertices of H would have to be non-star-critical for such a property to occur (e.g. if $H \simeq K_{1,r}$ there are $r-1$ non-star-critical vertices). In a sense, star-critical graphs are minimal with respect to the star graph obtained with the application of the star operator $K_{\mathcal{S}}$. Recall that maximal star s_a absorbs maximal star s_b if, by removing one leaf of s_b, it becomes a substar of s_a.

Theorem 83. *If H is a n-vertex star-critical graph, $n \leq \frac{1}{2}\left(3|\mathcal{S}(H)|^2 - |\mathcal{S}(H)|\right)$.*

Proof. We begin by partitioning $V(H)$ in $K = \{v \in V(H) \mid \exists s_a \in \mathcal{S}(H), v = c(s_a)\}$, which contains the center of every maximal star of H; and $I = V(H) \setminus K$, which is a subset of its simplicial vertices. Note that I is an independent set of H, otherwise there would be an edge with endpoints $\{u, v\} \subseteq I$ and either u or v would be in K. I is partitioned in I_A, I_E: the removal of a vertex in I_A cause the absorption of at least

one star, the removal of a vertex in I_E causes the disappearance of some edge of the star graph.

$|K| \leq |\mathcal{S}(H)|$ since each maximal star has a center. To bound $|I|$, we divide the analysis in the two possible cases for a vertex to be star-critical.

- Suppose that the removal of some $z \in I_A$ causes s_a, with $u = c(s_a)$, to be absorbed by s_b. One of two possibilities arise: if z has only one neighbor then z is the only neighbour of u with this property; therefore there are at most $|K|$ such vertices. Otherwise, if z has at least two neighbors, there must be some $v \in N(z) \cap N(u)$ with $v \in s_b \setminus s_a$. However, since I is an independent set, $v \in K$. Therefore, for each maximal star s_a, since H is star-critical, there is at most one different $z \in I_A$ for each $v \in (N(u) \cap N(z) \cap K) \setminus s_a$ preventing v from being added to s_a. This implies that the number of vertices required to avoid absorption is at most $|\mathcal{S}(H)|(|K \setminus \{u\}|) \leq 2\binom{|\mathcal{S}(H)|}{2}$.

- For the other condition, each $z \in I_E$ could be responsible for the intersection of a different pair of stars of H; i.e., $\exists s_a, s_b \in \mathcal{S}(H)$ such that $s_a \cap s_b = \{z\}$. Since we have $\binom{|\mathcal{S}(H)|}{2}$ pairs, we may have as many vertices in I.

Summing both cases, we have $|I| \leq \frac{3}{2}\binom{|\mathcal{S}(H)|}{2}$ and since $n = |K| + |I|$, it holds that $n \leq \frac{3|\mathcal{S}(H)|^2 - |\mathcal{S}(H)|}{2}$. □

Corollary 84. *If H is star-critical and has no simplicial vertex, $|V(H)| \leq |\mathcal{S}(H)|$. If the only simplicial vertices of H are leaves, $|V(H)| \leq 2|\mathcal{S}(H)|$.*

Improvements to the bound given by Theorem 83 appear to require a complete characterization of non-star-critical vertices. Also, a better understanding of vertices that are required only for the intersection of some stars to be non-empty seems necessary in order to approach the problem through induction. We believe that a reasonable bound would be of the form $|V(H)| \leq 2|V(G)| + \sqrt{|E(G)|}$, where $G \simeq K_{\mathcal{S}}(H)$.

Conjecture 1. *For every star-critical graph H and its star graph G, it holds that $|V(H)| \leq 2|V(G)| + \sqrt{|E(G)|}$.*

If this result indeed holds, it would configure an important difference from other intersection graphs. For instance, there are clique-critical graphs which require a quadratic number of vertices on the pre-image Alcón [2006]. However, we conjecture an even stronger result, which we back with the experiments we describe later in this chapter.

Conjecture 2. *For every star-critical graph H and its star graph G, it holds that $|V(H)| \leq 2|V(G)|$.*

4.3 Characterization

Much of the following discussion will be about edge clique covers, a central piece on the characterization of many intersection graph classes. We denote this family of subgraphs of G by $\mathcal{Q} = \{Q_1, \ldots, Q_n\}$. The usual strategy in these constructions is to use each clique as a vertex of the pre-image; this is also our approach. Since each vertex $a \in V(G)$ must be a star in H, it is reasonable to partition each clique as $Q_i \sim \{Q_i^c, Q_i^f\}$, that is, the vertices $a \in Q_i^c$ correspond to the stars of G with center in $v_i \in V(H)$, while the vertices $a \in Q_i^f$ correspond to the stars of G where $v_i \in V(H)$ is a leaf. We call such an edge clique cover a *star-partitioned edge clique cover* of $\mathcal{Q}$.

To simplify our notation, for each $a \in V(G)$, we denote its *center* by $c(a)$, i.e. $c(a)$ is the unique i such that $a \in Q_i^c$, its *leaf set* by $F(a) = \{i \mid a \in Q_i^f\}$ and its *cover* by $Q(a) = F(a) \cup \{c(a)\}$. For each pair of cliques $Q_i, Q_j \in \mathcal{Q}$, their *leaf-leaf intersection* is given by $\mathsf{ff}(i,j) = Q_i^f \cap Q_j^f$ and its *center-leaf intersection* by $\mathsf{cf}(i,j) = \left(Q_i^c \cap Q_j^f\right) \cup \left(Q_i^f \cap Q_j^c\right)$.

Definition 85 (Star-compatibility)**.** Given a graph G and a star-partitioned edge clique cover $\mathcal{Q}$ of G, we say that $\mathcal{Q}$ is *star-compatible* if, for every $a \in V(G)$, $|Q(a)| \geq 2$, $\exists!\ i$ such that $a \in Q_i^c$ and if, for every $Q_i, Q_j \in \mathcal{Q}$, if $Q_i \cap Q_j \neq \emptyset$, either $\mathsf{cf}(i,j) = \emptyset$ or $\mathsf{ff}(i,j) = \emptyset$.

Definition 86 (Star-differentiability)**.** Given a graph G and a star-partitioned edge clique cover $\mathcal{Q}$ of G, we say that $\mathcal{Q}$ is *star-differentiable* if for every $Q_i \in \mathcal{Q}$ and for every pair $\{a, a'\} \subseteq Q_i$ the following conditions hold:

1. If $\{a, a'\} \subseteq Q_i^c$, there exists $Q_j, Q_k \in \mathcal{Q}$ such that $a \in Q_j^f$, $a' \in Q_k^f$, $a \notin Q_k^f$, $a' \notin Q_j^f$ and $\mathsf{cf}(j,k) \neq \emptyset$. Moreover, if $Q_i^c \cap Q_j^f \cap Q_k^f = \emptyset$, $\mathsf{cf}(j,k) \neq \emptyset$.

2. If $a \in Q_i^c$, $a' \in Q_k^c$ and $a \notin Q_k^f$, then there is some $j \in F(a)$ with $\mathsf{cf}(j,k) \neq \emptyset$, $j \notin Q(a')$ and, for every $j' \in F(a)$ with $\mathsf{cf}(j',k) = \emptyset$, $Q_i^c \cap \bigcap_{j'} \mathsf{ff}(j',k) \neq \emptyset$.

3. If $a \in Q_i^c$, $a' \in Q_k^c$ and $a \in Q_k^f$, for every $j \in F(a) \setminus \{k\}$, $\mathsf{cf}(j,k) = \emptyset$.

4. If $\{a, a'\} \subseteq Q_i^f$ and $j = c(a) \neq c(a') = k$, then either $Q_i^c \cap \mathsf{ff}\,(j,k) \neq \emptyset$ or $\mathsf{cf}\,(j,k) \neq \emptyset$.

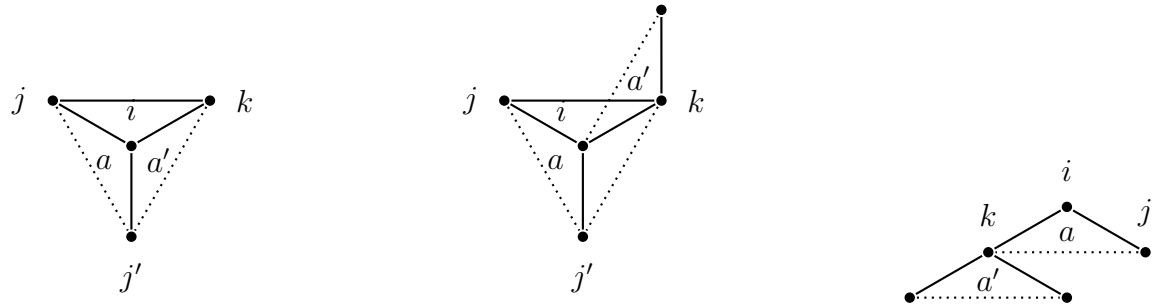

Figure 27: The first three cases of Definition 86. The first (left), second (center) and third (right).

Figure 28: The fourth case of Definition 86.

Figures 27 and 28 show the four cases of Definition 86 as seen on the pre-image of the star graph we shall build from $\mathcal{Q}$.

We emphasize that: (i) star-compatibility translates the structural properties of stars; and (ii) star-differentiability enumerates the possible ways that two stars that share at least one vertex are different. Note that, the "missing case", where $\{a, a'\} \in Q_i^f$ and $c(a) = c(a') = k$ is exactly the same case as 1, but with $\{a, a'\} \in Q_k^c$ instead of Q_i^c.

Lemma 87. *Let G be a graph and $\mathcal{Q}$ a star-partitioned edge clique cover of G. If $\mathcal{Q}$ is star-compatible and star-differentiable then, for every pair $\{a, a'\} \subseteq V(G)$, $Q(a) \nsubseteq Q(a')$ and $Q(a') \nsubseteq Q(a)$.*

Proof. If a and a' do not share any clique, the result is trivial.

Otherwise they do share some clique, say Q_i. For properties 1, 2 and 4 of Definition 86, we conclude that $i \in Q(a) \cap Q(a')$ and that $j \in Q(a)$, $k \in Q(a')$ but $j \notin Q(a')$ and $k \notin Q(a)$, which implies $Q(a) \nsubseteq Q(a')$ and $Q(a') \nsubseteq Q(a)$.

For property 3, however, we first conclude that there is some $j \in Q(a)$ but $j \notin Q(a')$, otherwise we would have $\mathsf{cf}(j, k) \neq \emptyset$ and $\mathsf{ff}(j, k) \neq \emptyset$. Consequently, $Q(a) \nsubseteq Q(a')$. For the converse, we note that $\{a, a'\} \subseteq Q_k$ and, following the same argument, we conclude that there is $j' \in Q(a')$ but $j' \notin Q(a)$ and, finally, that $Q(a') \nsubseteq Q(a)$. □

We now present a Krausz-type characterization for the class of star graphs.

Theorem 88. *An n-vertex graph G is the star graph of some graph H if and only if there is a star-compatible and star-differentiable star-partitioned edge clique cover $\mathcal{Q}$ of G with at most $\frac{1}{2}(3n^2 - n)$ cliques.*

Proof. In this proof, we assume that H has m vertices (v_i). For simplicity, a star $s_a \in \mathcal{S}(H)$ corresponds to the vertex $a \in V(G)$.

For the first direction of the statement, assume H is a star-critical pre-image of G. For each $v_i \in V(H)$, let $S(v_i) = \{s_a \in \mathcal{S}(H) \mid v_i \in s_a\}$, that is, the maximal stars of H that contain v_i. Clearly, we can partition these sets as $S(v_i) \sim \{S^c(v_i), S^f(v_i)\}$, that is, the stars where v_i is the center and where it is a leaf, respectively. Our goal is to show that $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$, with $Q_i^c = S^c(v_i)$ and $Q_i^f = S^f(v_i)$ is a star-partitioned edge clique cover of G satisfying star-compatibility and star-differentiability which, by Theorem 83, is all that remains is to be proven, since $|\mathcal{Q}| = |V(H)| \leq \frac{1}{2}(3n^2 - n)$.

To verify that $\mathcal{Q}$ is a star-partitioned edge clique cover of G, first note that every Q_i is a clique of G, since the corresponding stars share at least $v_i \in V(H)$. For the coverage part, every $aa' \in E(G)$ has two corresponding stars $s_a, s_{a'} \in \mathcal{S}(H)$, which share at least one vertex, say $v_i \in V(H)$, since $G \simeq K_{\mathcal{S}}(H)$. By the construction of $\mathcal{Q}$, there is some $Q_i \in \mathcal{Q}$ which corresponds to every maximal star that contains v_i; this guarantees that aa' is covered by at least one clique of $\mathcal{Q}$.

For the other properties, first take two vertices $v_i, v_j \in V(H)$ with $v_iv_j \notin E(H)$ but $S(v_i) \cap S(v_j) \neq \emptyset$. Clearly, no star in $S(v_i) \cap S(v_j)$ may have v_i and v_j in different sides of its bipartition, thus $S(v_i) \cap S(v_j) = S^f(v_i) \cap S^f(v_j)$. Now, suppose that $v_iv_j \in E(H)$; since they are adjacent, any star in $S(v_i) \cap S(v_j)$ must have v_i and v_j in opposite sides of the bipartition and, thus, we have that $S(v_i) \cap S(v_j) = \big(S^c(v_i) \cap S^f(v_j)\big) \cup \big(S^f(v_i) \cap S^c(v_j)\big)$. Since each star has a single center, the above analysis shows that $\mathcal{Q}$ satisfies star-compatibility.

For star-differentiability, let $\{s_a, s_{a'}\} \subseteq S(v_i)$. We break our analysis in the same order as the one given in Definition 86.

1. If $\{s_a, s_{a'}\} \subseteq S^c(v_i)$ there must be at least one leaf in each star, say v_j and v_k, respectively, not in the other and these leaves must be adjacent to each other, otherwise at least one of the stars would not be maximal. That is, $\{a, a'\} \in Q_i^c$ imply that there is $Q_j, Q_k \in \mathcal{Q}$ with $a \in Q_j^f$, $a' \in Q_k^f$, $a \notin Q_k^f$, $a \notin Q_j^f$ and $\mathsf{cf}(j, k) \neq \emptyset$.

2. If $s_a \in S(v_i)^c$, $s_{a'} \in S(v_k)^c$ and $s_a \notin S(v_k)^f$, $v_iv_k \in E(H)$ and to keep v_k from being a leaf of s_a, one leaf of s_a, say v_j, must also be adjacent to v_k and not a leaf of $s_{a'}$, since v_i is. Now, for every $v_{j'} \in s_a$ and not adjacent to v_k, there is a clear

$P_3 = v_k v_i v_{j'}$, which must be part of some maximal star. Moreover, the set of all $v_{j'}$ non-adjacent to v_k will form a maximal star centered around v_i along with v_k. Thus, $a \in Q_i^c$, $a' \in Q_k^c$ and $a \notin Q_k^f$, imply that there is some $j \in F(a)$ with $\mathsf{cf}(j,k) \neq \emptyset$, $j \notin Q(a')$ and, for $j' \in F(a)$ with $\mathsf{cf}(j',k) = \emptyset$, $Q_i^c \cap \bigcap_{j'} \mathsf{ff}(j',k) \neq \emptyset$.

3. If $s_a \in S(v_i)^c$, $s_{a'} \in S(v_k)^c$ and $s_a \in S(v_k)^f$, we know that $s_a = \{v_i\}\{v_k, \dots\}$ and, since v_k is not adjacent to any other leaf v_j of s_a, we know that $S(v_j) \cap S(v_k) = S^f(v_j) \cap S^f(v_k)$ and, since v_k is the center of $s_{a'}$, v_j is not one of its leaves. Therefore, $a \in Q_i^c$, $a' \in Q_k^c$ and $a \in Q_k^f$, implies that for every $j \in F(a) \setminus \{k\}$, $\mathsf{cf}(j,k) = \emptyset$.

4. If $\{s_a, s_{a'}\} \subseteq Q_i^f$ and $s_a \in S^c(v_j)$, $s_{a'} \in S^c(v_k)$, either $v_j v_k \notin E(H)$, which induces the existence a star $\{v_i\}\{v_j, v_k, \dots\}$, or $v_j v_k \in E(H)$, which must be part of a star with either v_j or v_k as center and the other as a leaf. Hence, $\{a, a'\} \subseteq Q_i^f$ and $j = c(a) \neq c(a') = k$, implies that either $Q_i^c \cap \mathsf{ff}\,(j,k) \neq \emptyset$ or $\mathsf{cf}\,(j,k) \neq \emptyset$.

The above shows that $\mathcal{Q}$ is also star-differentiable, which completes this part of the proof.

For the converse, take $\mathcal{Q}$ a star-partitioned edge clique cover of G satisfying star-compatibility and star-differentiability of size at most $\frac{1}{2}(3n^2 - n)$ and let H be a graph with $V(H) = \{v_i \mid Q_i \in \mathcal{Q}\}$ and $E(H) = \{v_i v_j \mid \mathsf{cf}(i,j) \neq \emptyset\}$ and let us prove that $G \simeq K_{\mathcal{S}}(H)$.

Take $a \in V(G)$ with $c(a) = i$. Due to star-compatibility and the construction of H, we know that $H[\{v_j \mid j \in F(a)\}]$ is an independent set of H and that $s_a = \{v_i\}\{v_j \mid j \in F(a)\}$ is a star of H. Suppose, however, that s_a is not maximal, that is, there is some $v_k \in V(H)$ such that $v_i v_k \in E(H)$ and $s_b = s_a \cup \{v_k\}$ is a star of H. By the construction of H, either there is some $a' \in V(G)$ such that $Q(a) \subseteq Q(a')$, which is impossible due to Lemma 87, or some $a' \in \mathsf{cf}(i,k)$, which we analyze below. The following is based on the first two cases of Definition 86; the other two are impossible, since $k \notin Q(a)$ and $a \in Q_i^c$.

1. If $a' \in Q_i^c$, there is some $Q_j \in \mathcal{Q}$ such that $a \in Q_j^f$ and $\mathsf{cf}(j,k) \neq \emptyset$, which implies that $v_j v_k \in E(H)$ and s_b is not a star of H.

2. If $a' \in Q_k^c$ and $a \notin Q_k^f$, at least one $j \in F(a)$ satisfies $\mathsf{cf}(j,k) \neq \emptyset$ and $j \notin Q(a')$. This gives us that $v_j v_k \in E(H)$ and s_b is not a star of H.

Therefore, we conclude that a' cannot exist, that s_a is maximal and, consequently that $V(G) \subseteq V(K_{\mathcal{S}}(H))$.

To show that $V(K_{\mathcal{S}}(H)) \subseteq V(G)$, take $s = \{v_i\}L$, with $s \in \mathcal{S}(H)$, and suppose that there is some $j, k \in L$ and that for every pair $a \in \mathsf{cf}(i,j)$ and $a' \in \mathsf{cf}(i,k)$, $a \notin Q_k$ and $a' \notin Q_j$. That is, $Q_i \cap Q_j \cap Q_k = \emptyset$, due to star-compatibility and the hypothesis that $jk \notin E(H)$. Once again, we analyze the possibilities in terms of Definition 86.

1. If $c(a) = c(a') = i$, we have that $\mathsf{cf}(j,k) \neq \emptyset$, implying that $v_j v_k \in E(H)$, contradicting the hypothesis that s exists.

2. If $c(a) = i$ and $c(a') = k$, there is some $j' \in Q(a)$ with $\mathsf{cf}(j,k) \neq \emptyset$. To conclude that $j = j'$, we note that, if $j \neq j'$, it would be required that $Q_i^c \cap \mathsf{ff}(j,k) \neq \emptyset$, which is impossible since $Q_i \cap Q_j \cap Q_k = \emptyset$. Once again, contradicting the hypothesis that such an s exists.

3. Trivially impossible since $Q_i \cap Q_j \cap Q_k = \emptyset$.

4. If $j = c(a) \neq c(a') = k$, either $Q_i^c \cap \mathsf{ff}(j,k) \neq \emptyset$, which is impossible since $Q_i \cap Q_j \cap Q_k = \emptyset$, or $\mathsf{cf}(j,k) \neq \emptyset$, implies that $v_j v_k \in E(H)$ and that s is not a star.

The above allows us to conclude that there is no $s \in \mathcal{S}(H)$ generated by cliques not pairwise intersecting. Such intersection has a unique vertex of G in it due to Lemma 87, which allows us to conclude $V(K_{\mathcal{S}}(H)) \subseteq V(G)$ and, consequently, that $V(K_{\mathcal{S}}(H)) = V(G)$.

To show that $E(G) \subseteq E(K_{\mathcal{S}}(H))$, we first take an edge $ab \in E(G)$. Since $\mathcal{Q}$ is a star-partitioned edge clique cover of G, there is some i such that $\{a, b\} \subseteq Q_i$ and, because $V(G) = V(K_{\mathcal{S}}(H))$ and the construction of H, there are corresponding stars $s_a, s_b \in \mathcal{S}(H)$ with $v_i \in s_a \cap s_b$ which guarantee that $ab \in E(K_{\mathcal{S}}(H))$. For $E(K_{\mathcal{S}}(H)) \subseteq E(G)$, take two intersecting stars $s_a, s_b \in \mathcal{S}(H)$ and note that, since $a, b \in K_{\mathcal{S}}(H) = V(G)$ and $\mathcal{Q}$ is a star-partitioned edge clique cover of G, $ab \in E(G)$ and we conclude that $E(G) = E(K_{\mathcal{S}}(H))$, completing the proof. □

We now pose a version of the decision problem for star graph recognition, which we call STAR GRAPH, and will be the subject of further study. We will further require that the output for any algorithm for STAR GRAPH is already star-partitioned.

STAR GRAPH
Instance: A graph G.
Question: Is there a star-partitioned edge clique cover $\mathcal{Q}$ of G satisfying star-compatibility and star-differentiability?

Theorem 88 provides a straightforward verification algorithm to check if a star-partitioned edge clique cover is star-compatible and star-differentiable.

Theorem 89. *Given a graph G of order n, there is an $\mathcal{O}(\max\{n^2m, m^2\}n^2m)$ algorithm to decide if a star-partitioned family $\mathcal{Q} \subseteq 2^{V(G)}$ of size m is an edge clique cover of G satisfying star-compatibility and star-differentiability.*

Proof. The first task is to determine whether or not $\mathcal{Q}$ is a star-partitioned edge clique cover of G. The usual n^2 algorithm that tests if each Q_i is a clique suffices. To check if $\mathcal{Q}$ is an edge clique cover, for each of the $\mathcal{O}(n^2)$ edges, we test if one of the n cliques contains it. This simple test takes $\mathcal{O}(n^2m)$ time.

To check for star-compatibility: first, for each vertex a of G and each clique Q_i, verify if there is a single i such that $a \in Q_i^c$ and at least one j with $a \in Q_j^f$; afterwards, for each pair of intersecting cliques Q_i, Q_j, test if $\mathsf{cf}(i,j) = \emptyset$ or $\mathsf{ff}(i,j) = \emptyset$. The entire process takes $\mathcal{O}(nm^2)$ time.

For star-differentiability, we assume that every pairwise intersection of $\mathcal{Q}$ has already been computed in time $\mathcal{O}(nm^2)$, and each query $\mathsf{cf}(j,k)$ and $\mathsf{ff}(j,k)$ takes $\mathcal{O}(1)$ time. Now, for each clique Q_i and for each pair of vertices $\{a, a'\} \in Q_i$, we must check one of the four conditions as follows.

1. If $c(a) = c(a') = i$, for each pair $j \in Q(a)$, $k \in Q(a')$, check if $a' \notin Q_j^f$, $a \notin Q_k^f$ and $\mathsf{cf}(j,k) \neq \emptyset$; this case takes $\mathcal{O}(n^2)$.

2. If $c(a) = i, c(a') = k$ and $a \notin Q_k^f$, for each $j \in F(a)$, check if either $\mathsf{cf}(j,k) \neq \emptyset$ and $j \notin Q(a')$ or $\mathsf{cf}(j,k) = \emptyset$ and $Q_i^c \cap \mathsf{ff}(j,k) \neq \emptyset$; this case takes $\mathcal{O}(n^2m)$ time.

3. If $c(a) = i, c(a') = k$ and $a \in Q_k^f$, check for each $j \in F(a) \setminus \{k\}$, if $\mathsf{cf}(j,k) = \emptyset$, taking $\mathcal{O}(n)$ time.

4. If $j = c(a) \neq c(a') = k$, we check if $Q_i^c \cap \mathsf{ff}(j,k) \neq \emptyset$ in $\mathcal{O}(n)$ time, and if $\mathsf{cf}(j,k)$ in $\mathcal{O}(1)$ time.

In the worst case scenario, we will spend $\mathcal{O}(\max n^2m, m^2)$ time for each Q_i and each pair $\{a, a'\} \subseteq Q_i$, of which there are $\mathcal{O}(n^2m)$ combinations, and conclude that the whole algorithm takes no more than $\mathcal{O}(\max\{n^2m, m^2\}n^2m)$ time. □

Together with Theorem 83, Theorem 89 implies that deciding whether or not a graph is a star graph is in NP, as we summarize in the following statement.

Theorem 90. STAR GRAPH *is in* NP.

4.4 Properties

The next theorem uses the known result, due to Moon and Moser [1965], that a graph of order n has at most $3^{n/3}$ maximal independent sets.

Theorem 91. *If G is the star graph of a n vertex graph H, then $|V(G)| \leq n3^{\Delta(H)/3}$.*

Proof. For every $v \in V(H)$, define $H_v = H[N(v)]$ and note that each maximal independent set of H_v might induce a maximal star of H centered around v. Since $|V(H_v)| \leq \Delta(H)$, we have that H_v has at most $3^{\Delta(H)/3}$ maximal independent sets and, therefore, H has at most $3^{\Delta(H)/3}$ maximal stars centered around v. Summing for every $v \in V(H)$ we arrive at the $n3^{\Delta(H)/3}$ bound. □

Theorem 92. *If G is a connected star graph, G has no cut-vertex.*

Proof. If $|V(G)| \leq 4$, we are done as there are only 5 graphs that satisfy these constraints and none of them contain a cut-vertex. They are K_1, K_2, K_3, K_4 and K_4 with one missing edge (the diamond). The first three are trivial, while the last two are shown in Figure 32

For graphs with 5 or more vertices, suppose that there is some cut-vertex $x \in G$, that A, B are two of the connected components obtained after removing x from G and take a pair of vertices $a \in V(A) \cap N(x)$, $b \in V(B) \cap N(x)$. Suppose now that $G = K_{\mathcal{S}}(H)$ for some H and take the stars s_a, s_b, s_x corresponding to a, b, x, respectively. Since $ab \notin E(G)$ and $ax, bx \in E(G)$, it holds that $s_a \cap s_x \neq \emptyset$ and $s_b \cap s_x \neq \emptyset$ but $s_a \cap s_b = \emptyset$.

If $c(a) = c(x) = i$ and $k = c(b) \neq c(x)$, s_x and s_b share at least one leaf, say v_j, since they intercept at some vertex, and $v_j \notin s_a$. However, there is no leaf $v_{j'} \in s_a$ adjacent to v_j, otherwise there would be an edge $v_j v_{j'} \in E(H)$ and, consequently, some star s_y, corresponding to vertex $y \in V(G)$, that keeps A, B connected and intercepts s_a, s_b, s_x. Therefore, we conclude that no leaf of s_a is adjacent to v_j and, since $c(a) = c(x)$ and $v_i v_j \in E(H)$, we conclude that $v_j \in s_a$, otherwise it would not be maximal, and, consequently, $v_j \in s_a \cap s_b$ and $ab \in E(H)$, which contradicts the hypothesis that A, B are disconnected after removing x. The case where $c(x) = c(b) \neq c(a)$ follows the exact same argument.

Now if $c(a) \neq c(x) = i$ and $c(x) \neq c(b)$, it is easy to see that v_i cannot be a leaf of both s_a and s_b simultaneously, otherwise $v_i \in s_a \cap s_b$ and $ab \in E(H)$. So we have two cases to analyze:

1. If v_i is a leaf of s_a, $v_j \in s_x \cap s_b$ and $k = c(b)$, clearly $s_y = \{v_j\}\{v_i, v_k, \dots\}$ is a maximal star of H that intercepts s_a, s_b, s_x, keeping A, B from being disconnected. The case where v_i is a leaf s_b is the same, and we omit it for brevity.

2. If v_i not a leaf of neither s_a nor s_b, $c(a) = j$ and $c(b) = k$, we have leaves $v_{j'} \in s_a \cap s_x$, $v_{k'} \in s_b \cap s_x$ which form at least two intercepting maximal stars, $s_{a'} = \{v_{j'}\}\{v_i, v_j, \dots\}$ and $s_{b'} = \{v_{k'}\}\{v_i, v_k, \dots\}$, such that $s_{a'} \cap s_a \cap s_x \neq \emptyset$ and $s_{b'} \cap s_b \cap s_x \neq \emptyset$.

These cases allow us to conclude that A, B remains connected no matter the configuration of the intersection of the corresponding stars in H. Consequently, x cannot exist and we complete the proof. □

Theorem 93. *Every edge of a star graph G is contained in at least one triangle if $|V(G)| \geq 3$.*

Proof. The only connected star graph with 3 vertices is K_3, so take G with $|V(G)| \geq 4$. Take a pre-image H of G, $ab \in E(G)$, $s_a, s_b \in \mathcal{S}(H)$ the corresponding stars to a, b, and assume that ab is not contained in any triangle of G. Since G is connected, there is at least one $x \in V(G)$ adjacent to (w.l.o.g) a, but not to b, and a corresponding maximal star s_x of H. Below, we analyze the possible intersections between s_a and s_b and conclude that there is always some star s_y that shares one vertex with s_a and s_b.

1. If $c(a) = c(b) = i$ and the center of s_x is a leaf of s_a, clearly v_i is not a leaf of s_x, otherwise $s_x \cap s_b \neq \emptyset$, therefore there is some leaf $v_j \in s_x$ with $v_iv_j \in E(H)$, which must be part of at least one maximal star s_y of H, from which we conclude that $s_a \cap s_b \cap s_y \neq \emptyset$, $s_a \cap s_x \cap s_y \neq \emptyset$ and both ab and ax are in a triangle of G.

2. If $c(a) = c(b) = i$ and a leaf v_j of s_x is a leaf of s_a, either the center v_k of s_x is adjacent to v_i, in which case $v_iv_k \in E(H)$ and we follow the same argument as in the previous case, or they are not adjacent, implying that there is a maximal star $s_y = \{v_j\}\{v_i, v_k, \dots\}$ which intercepts s_a, s_b, s_x, which allows us to conclude that s_a, s_b, s_x, s_y is a clique of G.

3. if $i = c(a) \neq c(b) = k$, there is some leaf $v_j \in s_a \cap s_b$. Clearly, if $v_iv_k \in E(H)$, there is a star that intercepts both s_a and s_b; otherwise, $v_iv_k \notin E(H)$ and we conclude that $s_y = \{v_j\}\{v_i, v_k, \dots\} \in \mathcal{S}(H)$ intercepts s_a and s_b and creates a triangle that contains ab.

□

The previous theorem implies that the minimum degree of any star graph on at least three vertices is at least two. A natural question arises about the vertices of degree two and the structures on the pre-image that generate them.

A *pending-P_4* $\{u, v, w, z\}$ is an induced path on four vertices that satisfies $d(u) = 1$, $N(v) = \{u, w\}$, $N(w) = \{v, z\}$ and $N(z)$ is an independent set of H. A *terminal triangle* is a set $\{u, v, z\}$ such that $N[u] = N[v] = \{u, v, z\}$, and no other pair of vertices in $N(z)$ is adjacent. In both cases, z is called the *anchor* of the structure. Our next result shows that for nearly all star graphs, their degree two vertices are either generated by pending-P_4's or terminal triangles.

Lemma 94. *If H is star-critical, $G = K_{\mathcal{S}}(H)$, and G is not isomorphic to a diamond, then every vertex of degree two of G is generated by a pending-P_4, or by a terminal triangle. Moreover, for every degree two vertex $a \in V(G)$, it holds that a has a vertex a' not adjacent to another vertex of degree two.*

Proof. If $|V(G)| \leq 3$ the result holds, so suppose $|V(G)| \geq 4$, let a be a degree two vertex of G with $N(a) = \{b, d\}$, s_a be the corresponding maximal star of H, $c(s_a) = v$, and $u \in s_a$ be one of its leaves. Since $d_G(a) = 2$, neither v nor any of its leave can be contained in any other maximal star of H, aside possibly from b and d.

Suppose that $s_a = \{u, v\}$. In this case, we have that both u and v are true twins, with $w \in N_H(u)$. If u is simplicial, $|N_H(u)| = 2$, otherwise a would have more than two neighbors. If $N_H(u) \setminus \{v\}$ is not an independent set, it has at least two adjacent vertices w, z forming a K_4 with u and v; regardless of the neighborhood of w and z, at least four distinct maximal stars contain either u or v, implying $d_G(a) \geq 3$. If $N_H(u) \setminus \{v\}$ is an independent set, we have two options:

1. $N_H(u) \supseteq \{v, w, z\}$, in which case at least one of w or z, say w, has a neighbor other than u and v, since H is star-critical. This configuration, however, generates a K_5 in G: two stars centered at w, s_a, one centered at v containing all its neighbors, except u, and one centered at u with all its neighbors except v.

2. Otherwise, $N_H(u) = \{v, w\}$. Since $|V(G)| \geq 4$, w necessarily has an additional neighbor. If $N_H(w) \setminus \{u, v\}$ is not an independent set of H, w has a pair of adjacent neighbors x, y, which are not adjacent to u nor v. However, note that there are at least four stars centered at w that intersect s_a, contradicting the hypothesis that a has only two neighbors in G.

From the above, we conclude that if $|s_a| = 2$, it corresponds to an edge of a terminal triangle.

On the other hand, suppose now that $s_a \supseteq \{u, v, w\}$, and that $c(s_a) = v$. Towards showing that $N_H(v)$ is an independent set, suppose that v has at least one edge in its neighborhood.

3. If no such edge is incident to u or w, then there are two maximal stars centered at v containing $\{u, v, w\}$ but, in this case, neither u nor w may have a star centered at it and, consequently, one of them is non-star-critical.

4. If w is adjacent to some $z \in N_H(v)$ but $zu \notin E(H)$, again there are two stars centered at v (s_a and another one containing $\{u, v, z\}$) both being adjacent to any star that includes the edge wz. For s_a to have only two neighbors, neither u, nor v, nor w may be in another other star. Since G is connected and has at least four vertices, z must have another neighbor x. We subdivide our analysis on the neighborhood of x:

   a) If $xv \in E(H)$, either x or z must be part of another maximal star; actually, x cannot be the center of another star (note that x is part of s_a, or it would be in another star that intersects s_a), so z must be part of another star, that is, it has a neighbor y not adjacent to x; but, in this case, $\{z, x, y\}$ intersects s_a, increasing the degree of a to at least three.

   b) So $xv \notin E(H)$ and there is a star centered at z containing $\{v, z, x\}$ which does not contain w, this implies that s_a intersects at least three stars.

5. If z is adjacent to both u and w, s_a already intersects two maximal stars – one containing vz and another containing $\{u, z, w\}$. Note that neither w nor u may have another neighbor, as that would inevitably generate a third star that intersects s_a. The only possibility would be that z is part of a maximal star that does not contain neither u, nor v, nor w. That is, every star that contains z has it as one of its leaves (otherwise we would have leaves adjacent to u, v, and w). This implies that $N_H(z) \setminus \{u, v, w\}$ is an independent set. However, either u or w is non-star-critical, since its removal does not change the intersection graph, a contradiction.

To realize that $N_H(v) = \{u, w\}$, note that at most two of the neighbors of v may have a single star centered at each of them, all others would be of degree one and, consequently, non-star-critical.

We now show that one of the neighbors of v has degree one. To see that this is the case, note that, if neither has degree one, both have at least one neighbor not adjacent to v and, thus, centers of maximal stars containing v. However, neither may be in

any other star, as this would increase the degree of s_a to more than two, but this is impossible, since at least one of u and w must be in another star for G to have at least four vertices and remain connected. For the remainder of the proof, suppose that u has degree one. Together with the fact that v only has two neighbors, we conclude that w must be in precisely two maximal stars, one of them with w being its center, since no neighbor of w may be adjacent to v. This implies that $N_H(w)$ is either an independent set or that it has at most one edge. If $N_H(w)$ has an edge xy, however, neither x nor y may have a neighbor not adjacent to w, otherwise we would have another star containing w and s_a would intersect three stars. In this case, G would be precisely a diamond. So now we have that $N_H(w)$ is also an independent set and, furthermore, $N_H(w) = \{z, v\}$, since H is star-critical. Now, the only way for w to be in more than two stars is if there is more than one star centered at z containing w; which is possible only if z is part of a triangle; so we also conclude that $N(z)$ is an independent set. This configuration is precisely a pending-P_4.

For the second assertion, to show that every vertex $a \in V(G)$ of degree two has a neighbor not adjacent to another vertex of degree two, suppose that this is not the case, and let a' be such that $N(a) = N(a')$. We have three possible cases:

6. If both a and a' belong to pending-P_4's. Note that, if s_a is generated by the P_4 $\{u, v, w, z\}$, we have that $s_{a'}$ must be formed by the P_4 $\{u', v', z, w\}$, since $s_{a'}$ must intersect the star centered at z and the star centered at w. However, this implies that H is isomorphic to P_6, since the degree of every vertex, except u and u', is two, and we have that G is a diamond, contradicting the hypothesis.

7. If s_a belongs to a pending-P_4 $\{u, v, w, z\}$ and $s_{a'}$ belongs to a terminal triangle, the anchor z of the pending-P_4 cannot be the same as the anchor of the terminal triangle, as it would violate the requirement that $N(z)$ is an independent set. This, however, makes it impossible for a' to be adjacent to the neighborhood of a.

8. If both stars belong to terminal triangles, a similar analysis as the previous case follows.

Finally, we conclude that at most one of the neighbors of a degree two vertex has another degree two neighbor. □

Corollary 95. *Let $E_2(G) = \{uv \in E(G) \mid d_G(u) = 2$ or $d_G(v) = 2\}$ be the set of edges incident to at least one degree two vertex of the star graph G. Unless G is isomorphic to a diamond or a triangle $E_2(G) \leq \min\left\{|V(G)| - 1, \frac{4}{7}|E(G)|\right\}$. The bounds are tight.*

Proof. Let $V_2(G) = \{v \in V(G) \mid d_G(v) = 2\}$. By the previous Lemma, we have that for each vertex of degree two there is another vertex non-adjacent to another degree two vertex. As such, each pair of edges of $E_2(G)$ with a common endpoint is in one-to-one correspondence with a degree two vertex and its exclusive neighbor, i.e., $|E_2(G)| \leq 2|V_2(G)| \leq |V(G)| - 1$. For the second case, for each degree two vertex v, its exclusive neighbor has at least two other edges, otherwise the non-exclusive neighbor would be a cut-vertex, but these edges may be between exclusive neighbors. As such, we have that $|E_2(G)| + \frac{1}{2}|E_2(G)| + \frac{1}{4}|E_2(G)| \leq |E(G)|$, implying $|E_2(G)| \leq \frac{4}{7}|E(G)|$. For the tightness of the bounds, the star graph of P_7, the gem, satisfies both conditions. □

We conclude this section with a result about the diameter of a star graph. In fact, when considering the iterated star operator, it appears that the diameter converges to either three or four, depending on the graph from which the process began, even though the graph itself does not seem to converge. We highlight that the bound of Theorem 96 is tight, as shown by the example of Figure 29.

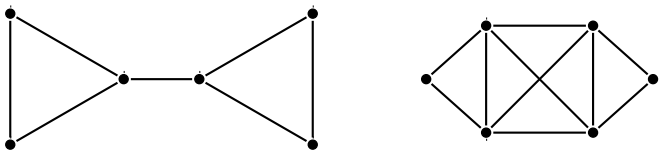

Figure 29: Problematic case of Theorem 96. The pre-image on the left and star graph on the right.

Theorem 96. *If H is a graph with diameter d and its star graph G is not a clique, then it holds that the diameter of G is at most $\lfloor \frac{d}{2} \rfloor + 2$.*

Proof. Let $P_G = \{s_1, \ldots, s_{k+1}\}$ be a diametrical path of G and $P'_G = P_G \setminus \{s_1, s_{k+1}\}$. For the following argumentation, we need to guarantee that the endpoints of the path in G have at least two vertices in a shortest path between their corresponding centers in H. Note that, in the case presented in Figure 29, neither of the degree two stars of the star graph satisfy the aforementioned condition. Let $u = c(s_2)$, $v = c(s_k)$, P_H be a shortest path between u and v with length, say r. If $r \geq 2k - 3$, we are done, as we would certainly have a path in G between u and v of length at least $\lceil \frac{2k-3}{2} \rceil = k - 2$ and, by adding stars s_1 and s_{k+1}, we would have a path of length at least k. Otherwise, $r < 2k - 3$, which directly implies that there is a path between s_2 and s_k of length at most $k - 3$, contradicting the hypothesis that P_G is a diametrical path. □

Corollary 97. *If H is a graph of diameter d and $G_k = K_S^k(H)$, for every $k \geq \lceil \log(d + 4) \rceil$, the diameter of G_k is either three or four, unless G_{k-1} is a clique, in which case the diameter of G is one.*

4.5 Small star graphs

By the observations made in the beginning of the chapter, star graphs and square graphs are quite similar, and even coincide under specific conditions, which is the case when the pre image is triangle-free. A natural question that thus arises is if the classes are actually the same. To see that there are star graphs which are not square graphs, Figure 30 presents a small example of such a graph.

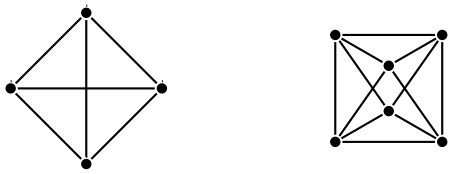

Figure 30: The star graph of K_4 is not a square graph.

Despite not coinciding for many classes of pre-images, it could be the case that every square graph also is a star graph, albeit for a different pre-image. The smallest example we found of a square graph which is not a star graph is shown in Figure 31: the square of the net. When attempting to show such a fact, without additional tools the combinatorial explosion of possible cases rapidly becomes intractable. At the same time, testing all graphs up to the bound given by Theorem 83 in search of a pre-image would be completely unfeasible, as we would need to test all connected graphs with up to *51 vertices*. However, Theorem 83 presents what we believe is a very loose value for the size of a pre-image, a claim we support with Theorem 98, its corollary, and some experiments we performed.

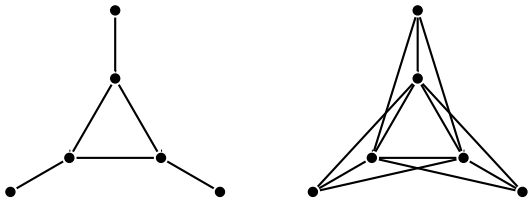

Figure 31: The square of the net is not a star graph.

Theorem 98. *Let H be an n-vertex graph, with $|\mathcal{S}(H)| = k$, and at least one non-star-critical vertex. For any H' with $n+1$ vertices such that H is an induced subgraph of H', at least one of the following holds:*

1. *H' has non-star-critical vertices; or*
2. *$|\mathcal{S}(H')| \geq k+1$.*

Proof. Since H is a proper induced subgraph of H', let y be the vertex in $V(H')\setminus V(H)$. If y is non-simplicial, or y is non-star-critical, or some vertex $x \in V(H)$ remains non-star-critical in H', we are done. The only cases that matter have y as a simplicial. Suppose that the statement is false, i.e. that every vertex of H' is useful and $|\mathcal{S}(H')| = k$; in particular, the vertex x, which is non-star-critical on H, becomes star-critical. Before proceeding, note that if $yx \in E(H')$, at least one of y or x must be the center of a star containing this edge and, therefore, we have a new star in H'. Moreover, y is not a false twin of x, otherwise y is non-star-critical. For the remainder of the proof, let $H'' = H \setminus \{x\}$ and $H^* = H'' \cup \{y\}$.

1. The removal of x from H' causes the absorption of $s'_a \in \mathcal{S}(H')$, such that $s'_a \supseteq s_a \in \mathcal{S}(H)$; note that this last assertion is true, otherwise s'_a is a new star generated by the addition of y.

   (a) If $s_a \supseteq \{x, z, w\}$ (centered at z, only w critical) we have that $(s_a \setminus \{x\}) \in \mathcal{S}(H'')$, because x is non-star-critical, implying that w and z form a pair of true twins, otherwise $s_a \setminus \{x\}$ would be absorbed by a star centered at w containing edge wz. However, $(s_a \setminus \{x\}) \notin \mathcal{S}(H^*)$ and $y \notin s'_a$; to see that this is the case, $y \in s'_a$ implies that $yz \in E(H')$, $yx, yw \notin E(H')$, and, thus, that x remains non-star-critical. Therefore, $(s_a \cup \{y\} \setminus \{x\}) \in \mathcal{S}(H')$, implying that $yx \in E(H')$, a contradiction.

   (b) If $s_a \supseteq \{x, z, w, \ell\}$ (centered at z, w, ℓ critical), we have that $(s_a \setminus \{x\}) \in \mathcal{S}(H'')$, but $(s_a \setminus \{x\}) \notin \mathcal{S}(H^*)$. Since s'_a has at least two useful vertices, whichever star absorbs s'_a must be centered at z; moreover, y must be in the star that absorbs $s'_a \setminus \{x\}$, otherwise, we would have that $s_a \setminus \{x\}$ is absorbed by a star that *already existed* in H''; consequently, we have $yx \in E(H')$ and a new star is formed by the addition of y.

2. There are two stars $s'_a, s'_b \in \mathcal{S}(H')$ such that $s'_a \cap s'_b = \{x\}$. Note that, at least one of s'_a and s'_b contains y, say s'_b, and we have that $(s'_b \setminus \{y\}) \notin \mathcal{S}(H)$. Therefore, s'_b is a new star that depends on y to exist and we have that $|\mathcal{S}(H')| \geq |\mathcal{S}(H)| + 1 = k + 1$.

□

To the best of our knowledge, analogous results to Theorem 98 are not known for clique or biclique graphs. These types of monotonicity properties are particularly useful when looking for small examples; the following statement is a direct corollary.

Figure 32: The two four vertex star graphs.

Figure 33: The four five vertex star graphs.

Corollary 99. *Let G be a k-vertex graph and $\mathcal{H}_n(k)$ be the set of all graphs on n vertices that has k maximal stars. If G is not isomorphic to the star graph of any star-critical $H \in \mathcal{H}_r(k)$, for any $r < n$ and every $H \in \mathcal{H}_n(k)$ is non-star-critical, then G is not a star graph.*

The above results allowed us to implement a procedure using McKay's Nauty package McKay and Piperno [2014]. Instead of only looking for the square of the net graph, we generated every single star graph on $k \leq 8$ vertices. In fact, for each k, no graph in $\mathcal{H}_{2k+1}(k)$ was star-critical. Figures 32, 33, and 34 present every star graph on four, five and six vertices, respectively. There are 46 star graphs on seven vertices, and 201 star graphs on eight vertices. Let $\mathcal{H}^*(k)$ denote the set of all star-critical pre-images for star graphs on k vertices. Our procedure also listed $\mathcal{H}^*(k)$ for every $k \leq 8$. In particular, there are 190 graphs in $\mathcal{H}^*(4)$, 1056 in $\mathcal{H}^*(5)$, 8876 in $\mathcal{H}^*(6)$, 76320 in $\mathcal{H}^*(7)$, and 892170 in $\mathcal{H}^*(8)$.

Corollary 100. *There are square graphs which are not star graphs (e.g. the square of the net), and there are star graphs which are not square graphs (e.g. the star graph of K_4).*

4.6 Concluding Remarks

This chapter introduced the class of star graphs – the intersection graphs of the induced maximal stars of some graph. We presented various results, beginning with a quadratic

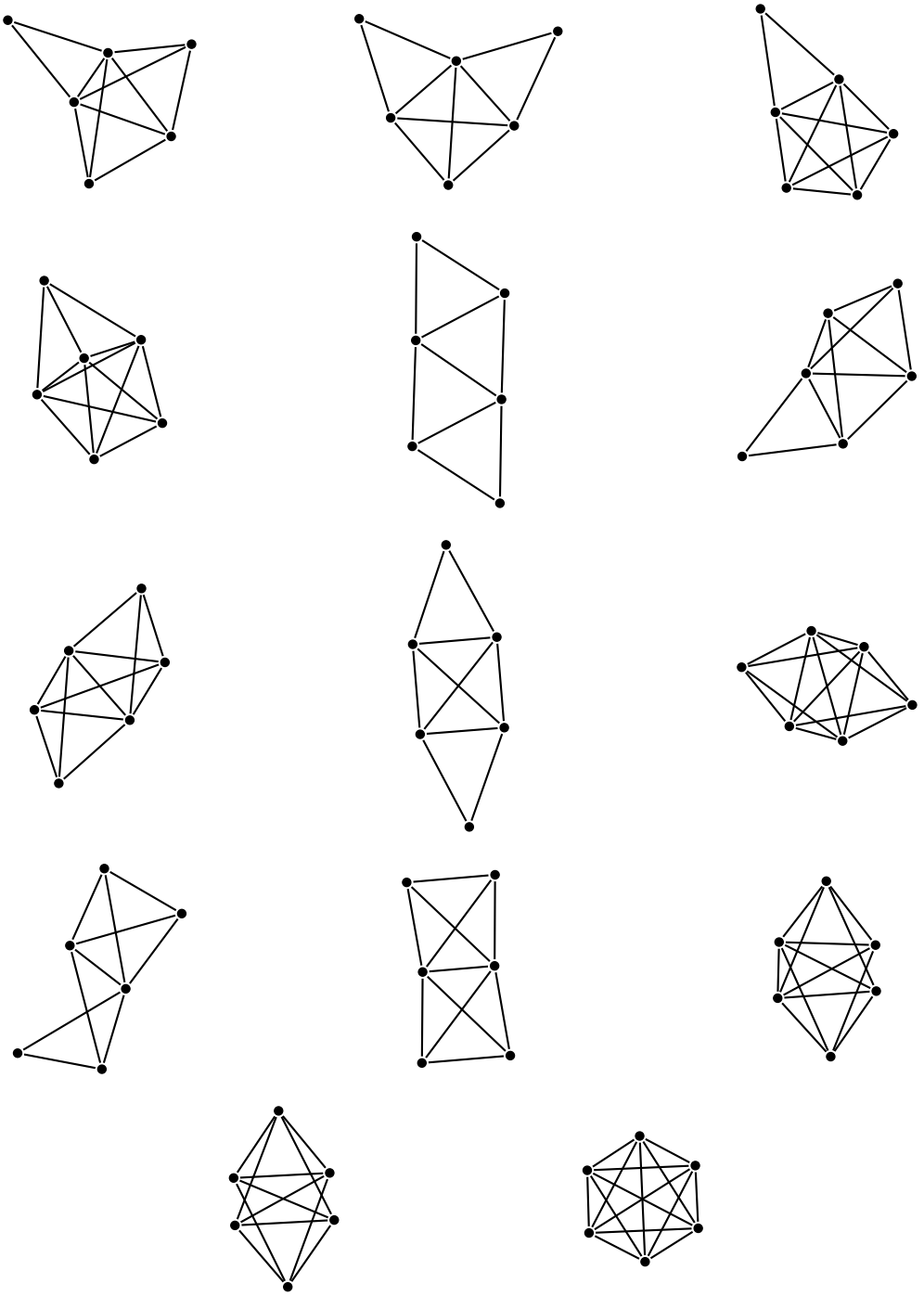

Figure 34: The fourteen six vertex star graphs.

bound on the size of minimal pre-images. Then, a Krausz-type characterization for the class was presented, which yielded membership of the recognition problem in NP. We also presented a series of properties the members of the class must satisfy, such as being biconnected, that every edge must belong to some triangle, and present a bound on the diameter of the star graph based on the diameter of the pre-image, which implies that the diameter of the iterated star graph converges to either three or four; when restricting the analysis to star-critical pre-images, we completely characterize the structures that the pre-image must have in order for the star graph to have a degree two vertex. We present a monotonicity theorem for star-critical pre-images, which allowed us to compute all 267 star graphs on at most eight vertices, as well as all 978612 star-critical pre-images of these graphs.

We leave two main open questions. The first, the complexity of the recognition problem, is perhaps the most challenging; for example, the complexity of the clique graph recognition problem was left open for many decades, only being settled recently [Alcón et al., 2009] through a series of non-intuitive gadgets. The second is the optimality of the quadratic bound, or if it can be improved. A complete characterization of both critical and non-critical vertices seems the biggest obstacle to obtain a linear bound on the size of critical pre-images; all our attempts to solve this task were thwarted by the amount of cases the analysis usually boils down to. Despite our special interests on these questions, many different directions are available for investigation, such as on iterated applications of the star operator or relationships with other classes.

Chapter 5

Final Remarks

This thesis dealt mainly with graph partitioning problems. Counted among them, are three coloring problems, multiple generalizations of the matching cut problem, and a broad study on a novel class of intersection graphs. The presented results vary quite a bit in nature: there are hardness reductions, exact, parameterized and polynomial algorithms, kernels, characterizations, and structural properties; connections with other problems and graph classes were established, and some previous results in the literature were significantly strengthened. The final section of the previous chapters gave an overview of the results presented in this thesis. We summarize in this last chapter some open problems and further research directions.

With respect to coloring problems, we presented a series of W[1]-hardness reductions for Equitable Coloring on different subclasses of chordal graphs, managing to present a hardness result for block graphs of diameter at least four. The notable difficulty of Equitable Coloring was already known, so our results do not come as so surprising. Problems that are still hard when parameterized by treewidth (e.g. List Coloring) usually have constraints beyond structural aspects of the input graph. The search for parameterizations that allow FPT algorithms is still necessary, and it appears that reasonable such parameterizations require some parameter that captures this non-topological flavor of the problem. Clique Coloring and Biclique Coloring, on the other hand, haven't been very explored in the literature, mostly because their placement on the second level of the polynomial hierarchy makes non-parameterized algorithmic analysis quite bleak, offering little to no hope of solving interesting instances in a feasible amount of time. These last two problems are, consequently, nice targets for further research on parameterized algorithms. Actually, W[1]-hardness proofs are far from trivial for these problems; by their very definitions, finding a clique or biclique coloring implies on guaranteeing that a quite large, and some time ill behaved, family

of subsets of vertices satisfies the desired coloring constraint.

For cut problems, despite adapting many results from MATCHING CUT, and even improving some of them, we leave many open questions related to d-CUT. First, we would like to close the gap between the known polynomial and NP-hard cases in terms of maximum degree, i.e., for each graph with maximum degree satisfying $d + 3 \leq \Delta(G) \leq 2d + 1$, we would like to how hard is it to find a d-cut. After much effort, we were unable to settle any of these cases. We are particularly interested in 2-CUT, where the only open case is for graphs of maximum degree equal to five. We recall the initial discussion about the INTERNAL PARTITION problem; closing the gap between the known cases for d-CUT would yield significant advancements on the internal partitions conjecture. As to the presented algorithms, all of them, in some way or another, have an exponential dependency on d. Answering whether or not this is necessary is interesting by itself, and merits further work; in particular, we would like to know if the kernel we presented can be improved. For ℓ-NESTED MATCHING CUT, we did not present nearly as many results as we did with d-CUT, but we do give a non-trivial exact exponential algorithm. Proving analogous results would be quite nice, but the real challenge in this problem is using the fact that what we are looking for is actually a special matching cut, where at least one of the parts admits even more matching cuts. Lastly, for p-WAY MATCHING CUT, we only offer a brief discussion, since most of the techniques we would use to prove results are analogous to the ones we described for d-CUT. As the central open question for this problem, we highlight the complexity of the algorithm parameterized by the number of edges crossing the cut, for which we were unable to either provide even an XP algorithm.

Finally, we also discussed the intersection graphs of maximal stars, which we called star graphs. We presented a series of properties of the class, a Krausz-type characterization, a bound on the size of minimal pre-images, membership of the recognition problem in NP, and even dabbled, albeit very briefly, on the iterated star operator. As mentioned in the conclusions of Chapter 4, we have two particular interests on future work on this graph class: (i) completely settling the complexity of the recognition problem, which we believe to NP-complete, and (ii) finding a linear upper bound on the size of star-critical pre-images. This last problem is backed by our computational experiments, where for each value $1 \leq k \leq 8$ of maximal stars, no star-critical graph on more than $2k$ vertices exists. To achieve this, we require a more thorough understanding of what star-critically implies and how we can identify non-star-critical vertices. Aside from these two problems, many other questions remain unanswered. Iterated biclique graphs have attracted a lot of attention recently, and iterated star graphs might benefit greatly from the reasoning strategies used in these studies. Also of interest is the

study of the star operator under restrictions either on the pre-image or on the image of the operator; as discussed, working on triangle-free pre-images is all about working on square graphs. On the other hand, if we only consider C_4-free pre-images, we are working on (possibly part of) the intersection between biclique and star graphs.

Bibliography

A. McKee, T. and McMorris, F. R. (1999). *Topics in Intersection Graph Theory*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Abboud, A., Bringmann, K., Hermelin, D., and Shabtay, D. (2019). Seth-based lower bounds for subset sum and bicriteria path. *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, page 41–57.

Alcón, L. (2006). Clique-critical graphs: Maximum size and recognition. *Discrete Applied Mathematics*, 154(13):1799 – 1802. ISSN 0166-218X. Traces of the Latin American Conference on Combinatorics, Graphs and Applications.

Alcón, L., Faria, L., de Figueiredo, C. M., and Gutierrez, M. (2009). The complexity of clique graph recognition. *Theoretical Computer Science*, 410(21):2072 – 2083.

Aravind, N. R., Kalyanasundaram, S., and Kare, A. S. (2017). On structural parameterizations of the matching cut problem. In *Proc. of the 11th International Conference on Combinatorial Optimization and Applications (COCOA)*, volume 10628 of *LNCS*, pages 475--482.

Bacsó, G., Gravier, S., Gyárfás, A., Preissmann, M., and Sebo, A. (2004). Coloring the maximal cliques of graphs. *SIAM Journal on Discrete Mathematics*, 17(3):361--376.

Baker, B. S. and Coffman, E. G. (1996). Mutual exclusion scheduling. *Theoretical Computer Science*, 162(2):225--243.

Ban, A. and Linial, N. (2016). Internal partitions of regular graphs. *Journal of Graph Theory*, 83(1):5--18.

Bellman, R. (1957). *Dynamic Programming (Princeton Landmarks in Mathematics and Physics)*. Princeton University Press.

Berge, C. (1984). *Hypergraphs: combinatorics of finite sets*, volume 45. Elsevier.

Berge, C. and Duchet, P. (1984). Strongly perfect graphs. *North-Holland mathematics studies*, 88:57--61.

Bertsimas, D. and Tsitsiklis, J. (1998). *Introduction to Linear Optimization*. Athena Scientific.

Björklund, A., Husfeldt, T., Kaski, P., and Koivisto, M. (2007). Fourier meets möbius: fast subset convolution. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 67--74. ACM.

Björklund, A., Husfeldt, T., and Koivisto, M. (2009). Set partitioning via inclusion-exclusion. *SIAM J. Comput.*, 39:546--563.

Blair, J. R. S. and Peyton, B. (1993). *An Introduction to Chordal Graphs and Clique Trees*, pages 1--29. Springer New York, New York, NY.

Bodlaender, H. L., Downey, R. G., Fellows, M. R., and Hermelin, D. (2009). On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423--434.

Bodlaender, H. L. and Fomin, F. V. (2004). *Equitable Colorings of Bounded Treewidth Graphs*, pages 180--190. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bodlaender, H. L., Jansen, B. M. P., and Kratsch, S. (2011). Cross-composition: A new technique for kernelization lower bounds. In *Proc. of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 9 of *LIPIcs*, pages 165--176.

Bodlaender, H. L., Jansen, B. M. P., and Kratsch, S. (2014). Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277--305.

Bodlaender, H. L. and Jansen, K. (1995). Restrictions of graph partition problems. part i. *Theoretical Computer Science*, 148(1):93 – 109. ISSN 0304-3975.

Bondy, J. A. and Murty, U. S. R. (1976). *Graph theory with applications*, volume 290. Macmillan London.

Bonsma, P. S. (2009). The complexity of the matching-cut problem for planar graphs and other graph classes. *Journal of Graph Theory*, 62(2):109--126.

Boral, A., Cygan, M., Kociumaka, T., and Pilipczuk, M. (2016). A fast branching algorithm for cluster vertex deletion. *Theory of Computing Systems*, 58(2):357--376.

Brandstädt, A., Le, V. B., and Spinrad, J. P. (1999). *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Brooks, R. L. (1941). On colouring the nodes of a network. In *Mathematical Proceedings of the Cambridge Philosophical Society*, pages 194--197. Cambridge University Press.

Campos, V., Lima, C., and Silva, A. (2013). B-coloring graphs with girth at least 8. In *The Seventh European Conference on Combinatorics, Graph Theory and Applications*, pages 327--332, Pisa. Scuola Normale Superiore.

Cerioli, M. R. and Korenchendler, A. L. (2009). Clique-coloring circular-arc graphs. *Electronic Notes in Discrete Mathematics*, 35(Supplement C):287 – 292. LAGOS'09 – V Latin-American Algorithms, Graphs and Optimization Symposium.

Cerioli, M. R. and Szwarcfiter, J. L. (2006). Characterizing intersection graphs of substars of a star. *Ars Combinatoria*, 79:21--31.

Chen, B.-L., Ko, M.-T., and Lih, K.-W. (1996). *Equitable and m-bounded coloring of split graphs*, pages 1--5. Springer Berlin Heidelberg, Berlin, Heidelberg.

Chen, B.-L., Lih, K.-W., and Wu, P.-L. (1994). Equitable coloring and the maximum degree. *European Journal of Combinatorics*, 15(5):443--447.

Chvátal, V. (1984). Recognizing decomposable graphs. *Journal of Graph Theory*, 8(1):51--53.

Cochefert, M. and Kratsch, D. (2014). Exact algorithms to clique-colour graphs. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 187--198. Springer.

Corneil, D., Lerchs, H., and Burlingham, L. (1981). Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163 – 174.

Courcelle, B. (1990). The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and computation*, 85(1):12--75.

Cygan, M., Fomin, F. V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., and Saurabh, S. (2015a). *Parameterized algorithms*, volume 3. Springer.

Cygan, M., Fomin, F. V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., and Saurabh, S. (2015b). *Parameterized Algorithms*. Springer.

de Werra, D. (1985). Some uses of hypergraphs in timetabling. *Asia-Pacific Journal of Operational Research*, 2(1):2--12.

Défossez, D. (2009). Complexity of clique-coloring odd-hole-free graphs. *Journal of Graph Theory*, 62(2):139--156.

Degiorgi, D. G. and Simon, K. (1995). A dynamic algorithm for line graph recognition. In *Proceedings of the 21st International Workshop on Graph-Theoretic Concepts in Computer Science*, WG '95, pages 37--48, London, UK, UK. Springer-Verlag.

DeVos, M. (2009). `http://www.openproblemgarden.org/op/friendly_partitions`.

Downey, R. G. and Fellows, M. R. (2013). *Fundamentals of parameterized complexity*, volume 4. Springer.

Duffus, D., Sands, B., Sauer, N., and Woodrow, R. E. (1991). Two-colouring all two-element maximal antichains. *Journal of Combinatorial Theory, Series A*, 57(1):109--116.

Escalante, F. and Toft, B. (1974). On clique-critical graphs. *Journal of Combinatorial Theory, Series B*, 17(2):170 – 182. ISSN 0095-8956.

Feige, U. and Kilian, J. (1996). Zero knowledge and the chromatic number. In *Computational Complexity, 1996. Proceedings., Eleventh Annual IEEE Conference on*, pages 278--287. IEEE.

Fellows, M., Heggernes, P., Rosamond, F., Sloper, C., and Telle, J. A. (2005). Finding k disjoint triangles in an arbitrary graph. In Hromkovič, J., Nagl, M., and Westfechtel, B., editors, *Graph-Theoretic Concepts in Computer Science*, pages 235--244, Berlin, Heidelberg. Springer Berlin Heidelberg.

Fellows, M. R., Fomin, F. V., Lokshtanov, D., Rosamond, F., Saurabh, S., Szeider, S., and Thomassen, C. (2011). On the complexity of some colorful problems parameterized by treewidth. *Information and Computation*, 209(2):143 – 153.

Fomin, F. V. and Kratsch, D. (2010). *Exact Exponential Algorithms*. Springer.

Fomin, F. V., Lokshtanov, D., Saurabh, S., and Zehavi, M. (2019). *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press.

FORD, L. R. and FULKERSON, D. R. (1962). *Flows in Networks*. Princeton University Press.

Frías-Armenta, M., Neumann-Lara, V., and Pizaña, M. (2004). Dismantlings and iterated clique graphs. *Discrete Mathematics*, 282(1):263 – 265. ISSN 0012-365X.

Ganian, R. (2012). Using neighborhood diversity to solve hard problems. *arXiv preprint arXiv:1201.3091*.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Gaspers, S. (2010). *Exponential Time Algorithms*. Serge Gaspers.

Godsil, C. and Royle, G. F. (2013). *Algebraic graph theory*, volume 207. Springer Science & Business Media.

Golovach, P. A., Johnson, M., Paulusma, D., and Song, J. (2017). A survey on the computational complexity of coloring graphs with forbidden subgraphs. *Journal of Graph Theory*, 84(4):331--363.

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

Graham, R. L. (1970). On primitive graphs and optimal vertex assignments. *Annals of the New York academy of sciences*, 175(1):170--186.

Groshaus, M., Guedes, A. L., and Montero, L. (2016). Almost every graph is divergent under the biclique operator. *Discrete Applied Mathematics*, 201:130 – 140. ISSN 0166-218X.

Groshaus, M. and Montero, L. P. (2013). On the iterated biclique operator. *Journal of Graph Theory*, 73(2):181–190.

Groshaus, M., Soulignac, F. J., and Terlisky, P. (2014). Star and biclique colouring and choosability. *Journal of Graph Algorithms and Applications*, 18(3):347--383.

Groshaus, M. and Szwarcfiter, J. L. (2010). Biclique graphs and biclique matrices. *Journal of Graph Theory*, 63(1):1--16.

Grötschel, M., Lovász, L., and Schrijver, A. (1984). Polynomial algorithms for perfect graphs. *North-Holland mathematics studies*, 88:325--356.

Gutner, S. (1996). The complexity of planar graph choosability. *Discrete Mathematics*, 159(1):119 – 130.

Hajnal, A. and Szemerédi, E. (1970). Proof of a conjecture of p. erdos. *Combinatorial theory and its applications*, 2:601--623.

Hermelin, D. and Wu, X. (2012). Weak compositions and their applications to polynomial lower bounds for kernelization. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 104--113, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Hua, Q.-S., Yu, D., Lau, F., and Wang, Y. (2009). Exact algorithms for set multicover and multiset multicover problems. *Algorithms and Computation*, pages 34--44.

Impagliazzo, R. and Paturi, R. (2001). On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367--375.

Jansen, K., Kratsch, S., Marx, D., and Schlotter, I. (2013). Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1):39 – 49.

Jarvis, M. and Zhou, B. (2001). Bounded vertex coloring of trees. *Discrete Mathematics*, 232(1-3):145--151.

Joos, F. (2014). A characterization of substar graphs. *Discrete Applied Mathematics*, 175:115 – 118. ISSN 0166-218X.

Kaneko, A. (1998). On decomposition of triangle-free graphs under degree constraints. *Journal of Graph Theory*, 27(1):7--9.

Karthick, T., Maffray, F., and Pastor, L. (2017). Polynomial cases for the vertex coloring problem. *arXiv preprint arXiv:1709.07712*.

Klein, S. and Morgana, A. (2012). On clique-colouring of graphs with few p4's. *Journal of the Brazilian Computer Society*, 18(2):113--119.

Kloks, T. (1994). *Treewidth. Computations and Approximations.* Springer-Verlag LNCS.

Komusiewicz, C., Kratsch, D., and Le, V. B. (2018). Matching Cut: Kernelization, Single-Exponential Time FPT, and Exact Exponential Algorithms. In *Proc. of the 13th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 115 of *LIPIcs*, pages 19:1--19:13.

Kratochvíl, J. and Tuza, Z. (2002). On the complexity of bicoloring clique hypergraphs of graphs. *Journal of Algorithms*, 45(1):40--54.

Kratsch, D. and Le, V. B. (2016). Algorithms solving the matching cut problem. *Theoretical Computer Science*, 609:328--335.

Larrión, F. and Neumann-Lara, V. (2002). On clique divergent graphs with linear growth. *Discrete Mathematics*, 245(1):139 – 153. ISSN 0012-365X.

Le, H. and Le, V. B. (2016). On the complexity of matching cut in graphs of fixed diameter. In *Proc. of the 27th International Symposium on Algorithms and Computation (ISAAC)*, volume 64 of *LIPIcs*, pages 50:1--50:12.

Le, V. B. and Randerath, B. (2003). On stable cutsets in line graphs. *Theoretical Computer Science*, 301(1-3):463--475.

Lih, K.-W. (2013). Equitable coloring of graphs. In *Handbook of combinatorial optimization*, pages 1199--1248. Springer.

Lokshtanov, D., Marx, D., and Saurabh, S. (2018). Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2):13:1--13:30. ISSN 1549-6325.

Lokshtanov, D., Marx, D., Saurabh, S., et al. (2013). Lower bounds based on the exponential time hypothesis. *Bulletin of EATCS*, 3(105).

Lonc, Z. (1992). *On complexity of some chain and antichain partition problems*, pages 97--104. Springer Berlin Heidelberg, Berlin, Heidelberg.

Lovász, L. (1973). Coverings and colorings of hypergraphs. In *Proc. of the 4th Southeastern Conference of Combinatorics, Graph Theory, and Computing*, pages 3--12. Utilitas Mathematica Publishing.

Ma, J. and Yang, T. (2019). Decomposing C_4-free graphs under degree constraints. *Journal of Graph Theory*, 90(1):13--23.

Macêdo Filho, H. B., Dantas, S., Machado, R. C., and Figueiredo, C. M. (2015). Biclique-colouring verification complexity and biclique-colouring power graphs. *Discrete Applied Mathematics*, 192:65--76.

Macêdo Filho, H. B., Machado, R. C., and de Figueiredo, C. M. (2016). Hierarchical complexity of 2-clique-colouring weakly chordal graphs and perfect graphs having cliques of size at least 3. *Theor. Comput. Sci.*, 618(C):122--134.

Macêdo Filho, H. B., Machado, R. C., and Figueiredo, C. M. (2012). Clique-colouring and biclique-colouring unichord-free graphs. In *Latin American Symposium on Theoretical Informatics*, pages 530--541. Springer.

Marx, D. (2006). Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394 – 406. Parameterized and Exact Computation.

Marx, D. (2011). Complexity of clique coloring and related problems. *Theoretical Computer Science*, 412(29):3487--3500.

Marx, D., O'Sullivan, B., and Razgon, I. (2010). Treewidth reduction for constrained separation and bipartization problems. In *Proc. of the 27th International Symposium on Theoretical Aspects of Computer Science, (STACS)*, volume 5 of *LIPIcs*, pages 561--572.

McKay, B. D. and Piperno, A. (2014). Practical graph isomorphism, {II}. *Journal of Symbolic Computation*, 60(0):94 – 112. ISSN 0747-7171.

Meyer, W. (1973). Equitable coloring. *The American Mathematical Monthly*, 80(8):920--922.

Mohar, B. and Skrekovski, R. (1999). The grötzsch theorem for the hypergraph of maximal cliques. *Electron. J. Combin*, 6(1):128.

Moon, J. W. and Moser, L. (1965). On cliques in graphs. *Israel Journal of Mathematics*, 3(1):23--28.

Moshi, A. M. (1989). Matching cutsets in graphs. *Journal of Graph Theory*, 13(5):527--536.

Naor, J. and Novick, M. B. (1990). An efficient reconstruction of a graph from its line graph in parallel. *Journal of algorithms*, 11(1):132--143.

Patrignani, M. and Pizzonia, M. (2001). The complexity of the matching-cut problem. In *Proc. of the 27th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 2204 of *LNCS*, pages 284--295.

Paulusma, D. (2016). *Open Problems on Graph Coloring for Special Graph Classes*, pages 16--30. Springer Berlin Heidelberg, Berlin, Heidelberg.

Pereira, F. M. Q. and Palsberg, J. (2005). *Register Allocation Via Coloring of Chordal Graphs*, pages 315--329. Springer Berlin Heidelberg, Berlin, Heidelberg.

Pătraşcu, M. and Williams, R. (2010). On the possibility of faster sat algorithms. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1065--1075. Society for Industrial and Applied Mathematics.

Robertson, N. and Seymour, P. (1986). Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309--322.

Roussopoulos, N. D. (1973). A max {m, n} algorithm for determining the graph h from its line graph g. *Information Processing Letters*, 2(4):108--112.

Shafique, K. H. and Dutton, R. D. (2002). On satisfactory partitioning of graphs. *Congressus Numerantium*, pages 183--194.

Smith, B., Bjorstad, P., and Gropp, W. (2004). *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*. Cambridge university press.

Stiebitz, M. (1996). Decomposing graphs under degree constraints. *Journal of Graph Theory*, 23(3):321--324.

Stockmeyer, L. J. (1976). The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1 – 22.

Thomassen, C. (1983). Graph decomposition with constraints on the connectivity and minimum degree. *Journal of Graph Theory*, 7(2):165--167.