Kernelization of Vertex Cover by Structural Parameters

Torstein Jarl Strømme
Department of Informatics
University of Bergen

A thesis submitted for the degree of
Master of Science

August 2015

This thesis is dedicated to my father

Stein Arild Strømme,

whose life will never cease to inspire me.

Acknowledgements

First and foremost, I must thank my supervisor Fedor V. Fomin for excellent guidance and valuable feedback. Your ability to listen actively and interrupt me when my thoughts were jumbled is really what made this thesis come together. I have thoroughly enjoyed your company, and I always felt encouraged leaving your office.

Thank you to the professors, researchers and students at the algorithms group for the great academic and non-academic discussions we had. I am especially thankful for the friendships I have grown with my fellow students; you made me look forward to learning something new every day, and the atmosphere you created in our study space was impeccable.

I would like to thank the Department of Informatics for accepting me as a student, and generously allowing me to teach as a group leader. You have also made the logistics of being a student as pleasant as it gets, and I am especially thankful for the smiles and helpfulness of your friendly advisors.

Finally, I must thank my family near and far for supporting me in this journey. In particular, I must express my deepest admiration for the patience and kindness my gorgeous wife Maria demonstrated whenever I came home tired after a long night of typing. However much I love computer science, I love you more.

Abstract

In the NP-complete problem Vertex Cover, one is given a graph G and an integer k and are asked whether there exists a vertex set $S \subseteq V(G)$ with size at most k such that every edge of the graph is incident to a vertex in S. In this thesis we explore techniques to solve Vertex Cover using parameterized algorithms, with a particular focus on kernelization by structural parameters. We present two new polynomial kernels for Vertex Cover, one parameterized by the size of a minimum degree-2 modulator, and one parameterized by the size of a minimum pseudoforest modulator. A degree-2 modulator is a vertex set $X \subseteq V(G)$ such that $G - X$ has maximum degree two, and a pseudoforest modulator is a vertex set $X \subseteq V(G)$ such that every connected component of $G - X$ has at most one cycle. Specifically, we provide polynomial time algorithms that for an input graph G and an integer k, outputs a graph G' and an integer k' such that G has a vertex cover of size k if and only if G' has a vertex cover of size k'. Moreover, the number of vertices of G' is bounded by $\mathcal{O}(|X|^7)$ where $|X|$ is the size of a minimum degree-2 modulator for G, or bounded by $\mathcal{O}(|X|^{12})$ where $|X|$ is the size a minimum pseudoforest modulator for G. Our result extends known results on structural kernelization for Vertex Cover. We also provide a $(d+2)$-approximation for the Minimum Degree-d Modulator problem.

Contents

Chapter 1

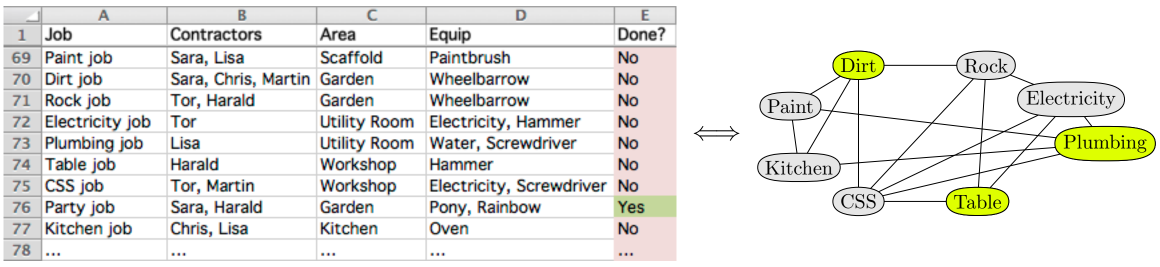
Introduction

Imagine you are a lead engineer in charge of a large renovation project, where you administer many small jobs that needs to be performed. You have specified the jobs in such a clever way that each job takes exactly one week to finish, so that every week you can put a nice check-mark in your spreadsheet for the jobs that were done that week.

Unfortunately, you cannot schedule every job at the same time since each job has some space, manpower and equipment requirements – if two jobs both require the wheelbarrow, then you can only schedule one of them any given week, or if two jobs require access to the same room or use the same sub-contractor, then you cannot schedule both of them at the same time. Admittedly you are a bit behind schedule, but your boss prefers watching ponies and rainbows on YouTube rather than checking in on your progress, so you are not especially worried.

Then one morning of particular misfortune, the company director told your boss to check in on your progress. Your boss observes the non-abundance of check-marks in your spreadsheet, and is abruptly outraged. He storms into your office and demands in flaming terms that at least 100 more jobs must be finished by next week, or else you will be fired. Is there any way you can save your job?

In order to answer this question, one could solve the Independent Set problem in the graph where each unfinished job is represented by a vertex, and where there is an edge between two vertices if their corresponding jobs have conflicting requirements; see Figure 1.1. In the this problem, one



	A	B	C	D	E
1	Job	Contractors	Area	Equip	Done?
69	Paint job	Sara, Lisa	Scaffold	Paintbrush	No
70	Dirt job	Sara, Chris, Martin	Garden	Wheelbarrow	No
71	Rock job	Tor, Harald	Garden	Wheelbarrow	No
72	Electricity job	Tor	Utility Room	Electricity, Hammer	No
73	Plumbing job	Lisa	Utility Room	Water, Screwdriver	No
74	Table job	Harald	Workshop	Hammer	No
75	CSS job	Tor, Martin	Workshop	Electricity, Screwdriver	No
76	Party job	Sara, Harald	Garden	Pony, Rainbow	Yes
77	Kitchen job	Chris, Lisa	Kitchen	Oven	No
78	...	...	...	...	...

Figure 1.1: Each unfinished job is represented as a vertex in the graph, and a conflict preventing two jobs to be executed at the same time is represented by an edge. The highlighted vertices shows an independent set, i. e. a set of vertices where there is no edge between any of them; the corresponding jobs may thus be done the same week.

wants to find the largest subset of vertices in a graph such that there is no edge between any of the chosen vertices. If your boss had rather required that there should be at most 60 jobs *left* by next week, then one could instead solve the Vertex Cover problem in the same graph. This is just a toy example, but there are plenty of real-world applications of these problems within areas such as scheduling, pattern recognition, molecular biology, and more.

Notice that if the total number of unfinished jobs is 160, then the two questions your boss could have asked above are equivalent; this hints to the close relationship between Independent Set and Vertex Cover. In Figure 1.1, the highlighted vertices form an independent set, whereas the gray vertices form a vertex cover. In this thesis we will state our results in terms of Vertex Cover, but more often than not we will work with Independent Set in obtaining our results.

The Vertex Cover problem is one of the first problems to be proved NP-complete [23], and it is one of the problems in Garey and Johnson's list of "core" NP-complete problems [15, Section 3.1]. This implies that we do not know of any efficient algorithm to solve general instances of the problem, and in practice we can only solve sufficiently small instances on modern computers. However, large instances still exists, and the algorithm designer must thus cope with the hardness in one way or another. The way of coping with NP-hardness employed in this thesis is the one of *parameterized* algorithms, and in particular the technique of *kernelization* by *structural* parameters.

In this field, Vertex Cover is one of the most well-studied problems, and it is often used as a testbed for developing and demonstrating new algorithmic ideas in the area. Various kernelization techniques, like reductions [1, 2], crown rule [5], and linear programming based kernelization [3] were initially developed for Vertex Cover.

Traditionally, kernelization is done with the parameter being the size of the solution, sometimes referred to as the *natural* parameter. The best kernel for Vertex Cover using the solution size as a parameter has $2k$ vertices, and there is complexity-theoretic evidence that this is optimal [16]. One might at first be tempted think that this is the the best one can do. However, in 2011 Jansen and Bodlaender [20] showed that Vertex Cover admits a kernel on $\mathcal{O}(|X|^3)$ vertices when parameterized by the size of a *feedback vertex set* X rather than the natural parameter. Since every vertex cover is also a feedback vertex set, the size of a minimum feedback vertex set is a *structurally smaller* parameter than the size of a minimum vertex cover, and the difference can be arbitrarily large. It may therefore make sense to kernelize using the smaller parameter, even when the degree of the polynomial is higher for kernels of this type.

This thesis introduces new reductions which in conjunction with those of Jansen and Bodlaender [20] show that Vertex Cover admits a kernel on $\mathcal{O}(|X|^7)$ vertices parameterized by the size of a degree-2 modulator, and a kernel on $\mathcal{O}(|X|^{12})$ vertices parameterized by the size of a pseudoforest modulator. A degree-2 modulator is structurally smaller than a vertex cover, but it is not structurally related to a feedback vertex set. A pseudoforest modulator, however, is structurally smaller than both a degree-2 modulator *and* a feedback vertex set. Thus, the results of this thesis extend graph classes where Vertex Cover can be considered tractable.

For the practitioner, these kernels provide means with which to solve a particular instance of Vertex Cover, since he may first approximate all the relevant structural parameters and then choose which kernelization(s) to perform. For this purpose we will provide a $(d+2)$-approximation to Degree-d modulator, which is a generalized version of the well-known 2-approximation to Vertex Cover by Gavril [15, page 134]. Even though it is not covered here, a pseudoforest modulator may also be approximated in polynomial time, seeing that this is a special case of the $\mathcal{F}$-Deletion problem [14].

1.1 Overview of the Thesis

In the remainder of this chapter, terminology and necessary definitions are introduced. In Chapter 2, we will give an introduction to the field of parameterized algorithms, introduce the concept of kernelization, and give examples of how Vertex Cover has been solved previously in this framework. We will also present a cubic kernel for Vertex Cover parameterized by a degree-1 modulator, which is a simplified version of the kernel parameterized by a feedback vertex set by Jansen and Bodlaender [20], and which serves as a gentle introduction to the techniques employed in our main results.

Starting in Chapter 3, we present the results of this thesis. First we show that Vertex Cover admits a kernel on $\mathcal{O}(|X|^7)$ vertices parameterized by the size of a degree-2 modulator X. Then, in Chapter 4, we show that Vertex Cover admits a kernel on $\mathcal{O}(|X|^{12})$ vertices parameterized by the size of a pseudoforest modulator X.

Finally, we give a summary, present some comments to our work, and suggest a few open problems for further research in Chapter 5.

1.2 Terminology

We will for the most part use standard mathematical terminology found in any modern textbook on the appropriate subject [27]. While the reader is expected to be familiar with these terms, a short recap of the concepts we will encounter is provided to avoid ambiguity.

1.2.1 Set Theory

A *set* is a collection of objects which makes a unit. The objects in a set S are also referred to as *members* of S. An object can be a anything, e. g. a number, a letter, and in the context of graphs, a vertex or an edge. An object can also be another set. The number of members in a set S, called the *cardinality* of S, is denoted by $|S|$. The set with zero members is called the *empty set* and is written $\emptyset$. Given two sets A and B, we use the following operations:

Subset	$A \subseteq B$ if for all members $x \in A$, we have $x \in B$.
Equality	$A = B$ if both $A \subseteq B$ and $B \subseteq A$.
Strict subset	$A \subset B$ if both $A \subseteq B$ and $A \neq B$.
Union	$A \cup B = \{x \mid x \in A \vee x \in B\}$
Intersection	$A \cap B = \{x \mid x \in A \wedge x \in B\}$
Difference	$A \setminus B = \{x \mid x \in A \wedge x \notin B\}$
Power set	$2^A = \{A' \mid A' \subseteq A\}$
Size k subsets	$\binom{A}{k} = \{A' \mid A' \in 2^A \wedge \lvert A'\rvert = k\}$ where k is a non-negative integer.

A *set family*, also called *collection*, is a set containing other sets as its objects. Objects which are not sets are called *elements*. We attempt to use the convention where calligraphic upper-case letters are used for set families, upper case letters are used for sets, and lower case letters are used for elements. Some operations are applicable for collections. Let $\mathcal{X} = \{X_1, X_2, \ldots, X_k\}$ be a collection. Then:

Union	$\bigcup_{X \in \mathcal{X}} X = X_1 \cup X_2 \cup \cdots \cup X_k$
Intersection	$\bigcap_{X \in \mathcal{X}} X = X_1 \cap X_2 \cap \cdots \cap X_k$

The operations presented here is only a tiny subset of what can be done within the framework of set theory. For an extensive overview, we refer to a textbook on the subject [30].

1.2.2 Graph Theory

In this thesis we will consider only finite, simple, loopless, undirected and unweighted graphs. The theory of graphs is wide and extensive, and the reader should consult a textbook on the subject for a complete overview [11].

Graph A *graph* G is a pair $(V(G), E(G))$ where $V(G)$ is a set of *vertices*, and $E(G) \subseteq \binom{V(G)}{2}$ is a set of *edges*. Each edge has two distinct vertices associated with it, called its endpoints. The two endpoints of an edge are said to be *adjacent* in G. We say that an edge is *incident* to its endpoints.

Neighborhood For a graph G and a vertex $v \in V(G)$, the neighborhood of v, denoted $N_G(v)$, is the set of all vertices adjacent to v in G. The closed neighborhood of v is denoted by $N_G[v] = N_G(v) \cup \{v\}$. Similarly, for a set $S \subseteq V(G)$ we denote its neighborhood $N_G(S) = (\bigcup_{v \in S} N_G(v)) \setminus S$, and its closed neighborhood $N_G[S] = N_G(S) \cup S$. In cases where it is clear which graph is being referred to, the subscript $(_G)$ may be omitted.

Degree For a graph G and a vertex $v \in V(G)$, the *degree* of v is the number of vertices adjacent to v in the graph, $deg_G(v) = |N_G(v)|$. In cases where it is clear which graph is being referred to, the subscript $(_G)$ may be omitted. A vertex of degree 1 is called a *leaf.* For a graph G, the maximum degree of any vertex in G is denoted by $\triangle(G)$.

Subgraph For two graphs G and G', G' is a *subgraph* of G, denoted by $G' \subseteq G$, if both $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. For a set $S \subseteq V(G)$, the maximal subgraph $G' \subseteq G$ such that $V(G') = S$ is called the subgraph *induced* by S, and is denoted $G[S]$. The graph where S and its incident edges are removed, is denoted as $G - S = G[V(G) \setminus S]$. Similarly, the graph $G[V(G) \setminus \{v\}]$ obtained by removing a single vertex $v \in V(G)$ and its incident edges is denoted by $G - v$, and the graph obtained by removing a subgraph $G' \subseteq G$ and its incident edges is denoted $G - G' = G[V(G) \setminus V(G')]$.

Path A *path* in a graph is a sequence of vertices connected by edges. In this thesis we will only consider *simple* paths, i. e. paths that doesn't contain any vertex more than once.

Cycle A *cycle* is a path where there is an edge between the first and the last vertex. In this thesis we will only consider *simple* cycles, i. e. cycles which contain at least three vertices and no vertex is contained more than once. Two cycles are considered non-distinct if they contain the same vertices in the same or opposite order, even if the starting and ending vertices are different.

Connected graph A graph G is said to be *connected* if there exists a path between all pairs of vertices in the graph. A *connected component* of a graph G is a subgraph $G' \subseteq G$ such that G' is connected. When referring to the connected components of a graph G, we refer to the *maximal* connected components of G, i. e. the vertex disjoint connected subgraphs of G such that their union equals G.

Tree A connected graph T is a *tree* if it contains no cycles. A tree is *rooted* if one vertex $r \in V(T)$ has been designated as the root. In rooted trees, all vertices have a natural orientation with respect to the root. For a non-leaf vertex $a \in V(T)$ we denote the set of its children by $C_T(a)$. The subscript $(_T)$ may be omitted if it is clear from the context which tree is being referred to. For two vertices $a, b \in V(T)$, we say that a is an *ancestor* of b if a is on the path from r to b (by this definition, a vertex is always an ancestor of itself).

Subtree For a rooted tree T and a vertex $a \in V(T)$, we let the *subtree* rooted at a be denoted by $T_a = T[\{b \mid a \text{ is an ancestor of } b\}]$. A subtree T_a is a *strict* subtree if $T \neq T_a$, i. e. if a is not the root.

Forest A graph F is a *forest* if every connected component of F is a tree.

Pseudotree A graph P is a *pseudotree* if P is connected and contains at most one distinct cycle.

Pseudoforest A graph F is a *pseudoforest* if every connected component of F is a pseudotree.

Triangle A graph C is a *triangle* if it contains exactly three vertices which are all adjacent.

Graph class A *graph class* is a (possibly infinite) collection of graphs which share a common property or structure.

1.2.3 Graph Properties and Graph Problems

We will here give some properties vertex sets and edge sets of a graph G can have. For each listed vertex/edge set property, there is a corresponding graph property denoting the maximum or minimum size such a set can take in G.

Vertex cover A *vertex cover* (VC) is a vertex set $S \subseteq V(G)$ such that every edge of G is incident to at least one vertex in S. The size of a minimum vertex cover of G is denoted $\textsc{vc}(G)$.

Independent set An *independent set* (IS) is a vertex set $I \subseteq V(G)$ such that no edge of G has both its endpoints in I. The *independence number* $\alpha(G)$ is the largest number of distinct vertices which can constitute an independent set of G. An independent set I of G with cardinality $|I| = \alpha(G)$ is called a *maximum independent set*, abbreviated MIS.

Feedback vertex set A *feedback vertex set* (FVS) is a vertex set $X \subseteq V(G)$ such that every cycle of G has at least one vertex in X. We also say that a feedback vertex set is a *forest modulator*, since $G - X$ is a forest. The size of a minimum feedback vertex set of G is denoted $\textsc{fvs}(G)$.

Degree-d modulator A *degree-d modulator* (DdM) is a vertex set $X \subseteq V(G)$ such that the maximum degree in the graph where X is removed from G is at most d, i. e. $\triangle(G - X) \leq d$. Note that vertex cover is exactly the case when $d = 0$; every degree-0 modulator is a vertex cover, and every vertex cover is a degree-0 modulator. We abbreviate a degree-1 modulator as D1M, and a degree-2 modulator as D2M. The size of a minimum degree-d modulator for G is denoted $\textsc{DdM}(G)$.

Pseudoforest modulator A *pseudoforest modulator* (PFM) is a vertex set $X \subseteq V(G)$ such that $G - X$ is a pseudoforest. The size of a minimum pseudoforest modulator for G is denoted $\textsc{pfm}(G)$.

Matching A *matching* is an edge set $M \subseteq E(G)$ such that no two edges in M share an endpoint. If every vertex of a graph G is in a matching M in G, then M is a *perfect* matching in G. We let $V(M)$ denote the vertices of a matching M. The largest number of distinct edges which can constitute a matching of G is called the maximum matching number, and is denoted $\textsc{mm}(G)$.

For each vertex/edge set property above, there is a corresponding optimization problem finding a maximum or minimum set having the property in question. For instance, the MINIMUM VERTEX COVER problem asks us to find a minimum vertex set $S \subseteq V(G)$ such that S is a vertex cover. We will attempt to use the convention where we refer to graph problems in CAPITAL LETTERS, and sets with the given properties in regular case. It is also possible to define weighted and counting versions of these problems, but we will not encounter any of those in this thesis.

The problems corresponding to the set properties described here can be understood either as *optimization* problems, where the aim is to find a minimum or maximum sized set which satisfy the property, or as a *decision* problems, which asks whether there exists a set of a size k satisfying the property, where k is a non-negative integer provided along with the input. Optimization problems are prefixed by the words "maximum" or "minimum," e. g. the MAXIMUM INDEPENDENT SET problem, whereas the decision versions have no such prefix, e. g. the INDEPENDENT SET problem.

Given a graph decision problem Q and an instance (G, k), we say that (G, k) is a *yes-instance* of Q if there exists a set of size k in G which satisfy the property required by Q, and a *no-instance* otherwise.

Chapter 2

Parameterized Algorithms and Kernelization

In this chapter we will introduce the field of parameterized complexity, with a particular focus on kernelization and how Vertex Cover has been solved in this framework previously. Before we do, we will draw some attention to the broader picture, where we briefly discuss complexity theory and different approaches for coping with NP-hardness.

2.1 Coping with NP-hardness

In the field of computational complexity theory, one classifies problems according to their difficulty, namely how much time and/or space is required to solve the problem with a computer. The most famous open question in this area, and possibly within all of computer science, is whether P is equal to NP or not. Informally we say that a problem Q is in the class P if there exists a deterministic algorithm which solves all instances of Q optimally in polynomial time, i. e. time bounded by a polynomial function of the size of the input description. Assuming that P $\neq$ NP, then there is no such algorithm for problems which are NP-hard. There are written hundredfold books and papers on the subject that can be consulted for a deeper introduction [29].

As decades has passed without anyone proving P = NP, alternative techniques to solve NP-hard problems have emerged [18]. Here are some of them.

Randomized algorithms This approach is relaxing the requirement that the algorithm is *deterministic*, and can thus have either a randomized running time, or produce the correct answer only with some probability strictly less than 1.

Restricted input algorithms This approach is relaxing the requirement that the algorithm solves *all* instances of the problem, but is only concerned with solving problem instances with a certain structure. For instance, an algorithm which solves Vertex Cover only on trees.

Approximation algorithms This approach to NP-hard optimization problems is relaxing the requirement that the algorithm solves the problem *optimally.* Instead, some guarantee is given that the found solution is not too far from the optimum (as opposed to heuristics, which gives no such guarantee).

Fast exponential time algorithms This approach is relaxing the requirement that the problem is solved in *polynomial time*, and is rather concerned with making smart choices which bring down the important constants of the running time, such as the base of the exponential.

Parameterized algorithms This approach is relaxing the requirement that the algorithm solves *all* instances in *polynomial time.* Rather, for some property of the problem quantifiable by a parameter k, the algorithm will have a running time which is polynomial as a function of the input description size n, but is allowed to be super-polynomial as a function of the parameter k. For a class of parameterized problems where the parameter k is bounded by some constant, the algorithm will thus run in polynomial time. Pseudo-polynomial algorithms, such as the $\mathcal{O}(nW)$-algorithm for the Knapsack problem, are special cases of parameterized algorithms.

The topic for this thesis is within parameterized algorithms, which we will introduce more thoroughly in the next section. The approaches above may also be combined in various ways, and in our results we rely on restricted input algorithms for solving Independent Set on certain graph classes, such as trees, cycles and pseudoforests.

2.2 Definitions in Parameterized Complexity

Parameterized complexity is a branch of complexity theory where one attempts to classify parameterized problems according to their difficulty under the assumption that P $\neq$ NP. Some NP-hard problems can be solved in time which is polynomial in the *size* of the input, but require exponential time or worse in terms of some parameter k which describe a quantifiable property of the problem. Thus if k is small, or at most some fixed constant, then the problem can still be considered tractable even though NP-hard problems in general are intractable. This is what the concept of *fixed-parameter tractability* (fpt) captures; however, before we formally define it, we need to know formally what a parameterized problem is.

Definition 2.1 (Parameterized problem [7, Definition 1.1])**.** A parameterized problem Q is a language $\mathrm{Q} \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed, finite alphabet.

This means that every valid instance of a parameterized problem Q is on the form (I, k), where I describes the input and the parameter k is a natural number. For example, an instance of Vertex Cover parameterized by the solution size is a pair (G, k), where G is a graph encoded as a string over Σ, and k is the solution size. We say that (G, k) is in Vertex Cover if and only if G correctly encodes a graph, and there is a vertex cover of size k in the graph encoded by G.

Definition 2.2 (Fixed-parameter tractable [7, Definition 1.2])**.** A parameterized problem $\mathrm{Q} \subseteq \Sigma^* \times \mathbb{N}$ is called *fixed-parameter tractable* (fpt) if there exists an algorithm $\mathcal{A}$, a computable function $f : \mathbb{N} \to \mathbb{N}$, and a constant c such that, given $(I, k) \in \Sigma^* \times \mathbb{N}$, the algorithm $\mathcal{A}$ correctly decides whether $(I, k) \in \mathrm{Q}$ in time bounded by $f(k) \cdot |(I, k)|^c$. The complexity class containing all fixed-parameter tractable problems is called FPT.

So far we have discovered what an fpt-algorithm is, but we have said nothing about kernelization, which is the topic of this thesis. Intuitively, a kernelization algorithm can be viewed as a preprocessing algorithm which throws away parts of the problem which are easy to solve. It then leaves behind only the “hard” part of the problem in the output instance, but includes a guarantee that the size of the output is bounded by a function of the parameter k. More formally, we use the following definition.

Definition 2.3 (Kernel [7, Definition 2.1])**.** A *kernelization algorithm*, for a parameterized problem Q is an algorithm $\mathcal{A}$ that, given an input instance (I, k), works in polynomial time and returns a *kernel*, i. e. an output instance (I', k') such that $(I, k) \in \mathrm{Q}$ if and only if $(I', k') \in \mathrm{Q}$. Moreover, we require that $|I'| + k' \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.

If the upper bound $g(\cdot)$ is a polynomial function of the parameter, then we say that Q admits a *polynomial kernel*.

In this thesis we obtain a kernel by making use of *reduction rules*. A reduction can be viewed as a miniature version of a kernel, but without any guarantee on the output size. A reduction for a problem Q receives as input an instance (I, k), and outputs an equivalent instance of the same problem (I', k'). A reduction is *safe* if $(I, k) \in \mathrm{Q} \iff (I', k') \in \mathrm{Q}$. Typically, a collection of reduction rules applied in some defined order together constitute a kernel. When no reduction rules can be applied to an instance, we call the instance *reduced*.

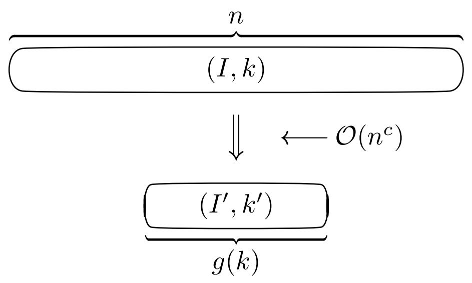


Figure 2.1: Kernelization: A polynomial-time algorithm which transforms an input instance (I, k) of a problem to an equivalent output instance (I', k') with a guarantee $g(k)$ on the size of the output. $\mathcal{O}(n^c)$ represents the runtime of the kernelization algorithm, where c is a constant.

Note that solving a parameterized problem Q by first finding the kernel for Q and then solving the kernel by use of brute force yields a running time of $|(I,k)|^c + f(k)$, where c is a constant, $|(I,k)|^c$ represents the time spent by the kernelization algorithm, and f can be any function representing the time spent by the brute force algorithm working on the kernel. It is then clear that any parameterized problem which admits a kernel is FPT. More surprisingly, the other direction is also true, as witnessed by this proposition [7, Lemma 2.2]:

Proposition 2.4. *If a parameterized problem Q is FPT, then it admits a kernelization algorithm.*

This implies that a problem admits a kernel if and only if it is FPT. We can therefore view kernelization as an alternative definition of fixed-parameter tractability. However, it turns out (under common complexity-theoretic assumptions) that not every problem in FPT admits a *polynomial* kernel. We can think of parameterized problems which admit polynomial kernels as "easier" than those who do not, and this distinction gives rise to a finer hardness classification within FPT.

We will in the next section give some simple examples of parameterized algorithms and polynomial kernels for Vertex Cover.

2.3 Parameterized Algorithms for Vertex Cover

What does a parameterized algorithm look like in practice? In this section we will show some examples of different parameterized algorithms for our problem of choice, Vertex Cover. In this problem, we are given a graph G and a non-negative integer k and are asked if there exists a set S of at most k vertices in G such that every edge in G is adjacent to at least one vertex in S. We will for this section use the size of the solution k as the parameter. All the results of this section are previous work, and can be found in an appropriate textbook on the subject [7].

2.3.1 Edge Branching

In this recursive algorithm we branch on which endpoint of a given edge covers the edge, and obtain a running time of $2^k \cdot n^{\mathcal{O}(1)}$. Observe that this running time shows that Vertex Cover is fixed-parameter tractable.

The key observation of Algorithm 2.1 is that every edge needs to be covered, and for each edge uv, there are two possibilities: it is covered either by u or by v. After an arbitrary edge has been chosen, the algorithm recurse on the two choices of how to cover it. Since k is reduced by one in each call, the depth of the recursion is at most k, and seeing that there are at most two branches in each call, $\mathrm{VC}_{\mathrm{EB}}$ is called $\mathcal{O}(2^k)$ times. The other work done in each call is to make two copies of G, which takes polynomial time.

Algorithm 2.1 VERTEX COVER by Edge Branching, $\text{VC}_{\text{EB}}(G, k)$

```
Input: Graph $G$, non-negative integer $k$.
Output: TRUE if $G$ contains a vertex cover of size $k$, FALSE otherwise.

1: if $k = 0$ and $|E(G)| > 0$ then
2:     return FALSE
3: else if $|E(G)| = 0$ then
4:     return TRUE
5: else
6:     Pick $uv \in E(G)$ arbitrarily
7:     return $\text{VC}_{\text{EB}}(G - u, k - 1)$ or $\text{VC}_{\text{EB}}(G - v, k - 1)$
8: end if
```

2.3.2 Vertex Branching

This is an improved version of the edge branching algorithm, which obtain an improved running time of $3 \cdot 1.466^k \cdot n^{\mathcal{O}(1)}$. This algorithm is combining the approaches of parameterized algorithms, restricted input algorithms, and fast exponential time algorithms.

The first important observation we need to make, is that VERTEX COVER is polynomial time solvable on graphs of maximum degree two, i. e. graphs where every connected component is either a path or a cycle. We can therefore assume that we have an algorithm $\text{VC}_{\triangle 2}$ which solves VERTEX COVER on such graphs in polynomial time. The second key observation, is that for some vertex $v \in V(G)$ with $N(v) \neq \emptyset$, either v is in the optimal vertex cover S, or *all* of $N(v)$ must be.

Algorithm 2.2 VERTEX COVER by Vertex Branching, $\text{VC}_{\text{VB}}(G, k)$

```
Input: Graph $G$, integer $k$.
Output: TRUE if $G$ contains a vertex cover of size $k$, FALSE otherwise.

 1: if $k \geq 0$ and $|E(G)| = 0$ then
 2:     return TRUE
 3: else if $k \leq 0$ then
 4:     return FALSE
 5: else if $\triangle(G) \leq 2$ then
 6:     return $\text{VC}_{\triangle 2}(G, k)$
 7: else
 8:     Pick $v \in V(G)$ such that $deg(v) \geq 3$
 9:     return $\text{VC}_{\text{VB}}(G - v, k - 1)$ or $\text{VC}_{\text{VB}}(G - N[v], k - deg(v))$
10: end if
```

The number of leaves in the recursion tree of Algorithm 2.2 will be at most $3 \cdot 1.466^k$. To see this, we provide the following lemma.

Lemma 2.5 ([7, Theorem 3.2])**.** *Let $T(k)$ be the maximum number of leaves in the recursion tree of Algorithm 2.2, where k is the parameter. Then $T(k) \leq 3 \cdot 1.466^k$ for every $k \geq 0$.*

Proof. We will do a proof by strong induction on k. See that the base case holds for $k \in \{0, 1, 2\}$ (they all have at most 3 leaves). For the inductive case $k \geq 3$, observe that

$$T(k) = T(k-1) + T(k - deg(v))$$

v is here the vertex chosen on line 8 of Algorithm 2.2. Observe that in cases where $k - deg(v) < 0$, then $T(k - deg(v)) = 1$ by inspection of the algorithm. Since $k \geq 3$, we see that $3 \cdot 1.466^{k-3} \geq 1$. Further, because $deg(v) \geq 3$ and $3 \cdot 1.466^{k-3} \geq 3 \cdot 1.466^{k-c}$ for every $c \geq 3$, we have by the induction hypothesis that

$$\begin{aligned} T(k) &\leq 3 \cdot 1.466^{k-1} + 3 \cdot 1.466^{k-3} \\ &\leq 3 \cdot 1.466^{k}(1.466^{-1} + 1.466^{-3}) \end{aligned}$$

Seeing that $1.466^{-1} + 1.466^{-3} \leq 1$, we have now proven the lemma. □

Since the work done in each call of $\mathrm{VC_{VB}}$ is polynomial, this implies that the running time of the algorithm is $3 \cdot 1.446^k \cdot n^{\mathcal{O}(1)}$ as stated.

2.3.3 Quadratic Kernel

So far, we have seen examples of parameterized algorithms which solve Vertex Cover, but we have not yet seen examples of kernels. In this and the next sections we will give such examples. We will begin by giving a kernel for Vertex Cover on at most $2k^2$ vertices. In this classic example by Buss [2], we will make use of three reduction rules which will be applied exhaustively, and when none of them are applicable, we will argue that the number of vertices in the reduced instance is at most $2k^2$.

Each reduction rule is given an input instance (G, k), and will return an equivalent output instance (G', k'). The reduction rules are as follows:

1. If there is a vertex $v \in G$ such that $deg(v) = 0$, then remove v from G. We let $G' := G - v$, and $k' := k$.

2. If there is a vertex $v \in G$ such that $deg(v) > k$, then remove v and its incident edges from G and reduce k by one. We let $G' := G - v$, and $k' := k - 1$.

3. If there are strictly more than k^2 edges in G, let (G', k') be a trivial *no*-instance: Let G' be two vertices connected by an edge, and let $k := 0$.

The first reduction is safe, since isolated vertices are never necessary in a vertex cover. The second reduction is safe, because if v is not in the vertex cover, then all of its neighbors must be; however, there are too many of them. The third reduction is safe, because each vertex can cover at most k

vertices seeing that the second reduction was not applicable. If there are more than k^2 edges, then there are too many to be covered, and it is a *no*-instance.

For the bound on the size of a reduced instance, observe that every vertex has at least one edge incident to it, or else the first reduction would be applicable. Since each edge is incident to at most two vertices, and there are at most k^2 edges, then it must be the case that there are at most $2k^2$ vertices.

2.3.4 Crown Reduction

In this section we will introduce an additional reduction rule which can be combined with those of the previous section as a fourth reduction. We then obtain a kernel for VERTEX COVER on at most $3k$ vertices in this reduction by Chor, Fellows and Juedes [5]. This reduction rule makes use of what is called a *crown decomposition* of a graph G, which is a partition of the vertices $V(G)$ into three sets C, H, R (crown, head, and rest) such that C is a non-empty independent set, there are no edges between C and R, and there exists a matching in the bipartite graph between H and C which saturates H. Such a decomposition is very useful in the pursuit of a minimum vertex cover due to the following lemma.

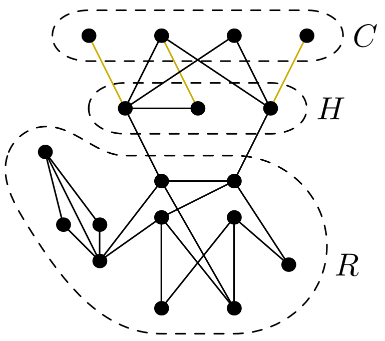


Figure 2.2: A crown decomposition: C is a non-empty independent set, there are no edges between C and R, and there exists a matching in the bipartite graph between H and C which saturates H (e. g. highlighted edges).

Lemma 2.6 ([5, Lemma 9])**.** *Let G be a graph without isolated vertices, and let (C, H, R) be a crown decomposition of G. Then there is a minimum vertex cover of G which contains all of H and none of C.*

Proof. Because there are no isolated vertices, every vertex in C must be incident to at least one vertex in H, so H is not empty. Since there is a matching in the bipartite graph between vertices of C and H which saturate H, then in order to cover the edges of that matching, at least $|H|$ vertices are required. These edges must necessarily be covered by vertices in C or H, but since there are no edges between C and R, the choice of all vertices in H is always at least as good as any choice involving vertices from C. □

If we obtain a crown decomposition of a graph without isolated vertices, then by Lemma 2.6 we can safely select H to be in the vertex cover. Removing H and its incident edges from G will yield all vertices of C isolated in the resulting graph, so C may safely be removed as well. The next proposition gives us an algorithm which either finds a crown decomposition, or a witness that a vertex cover of size k is impossible.

Proposition 2.7 (Crown Lemma [7, Lemma 2.14])**.** *Let G be a graph without isolated vertices and with at least $3k+1$ vertices. Then there is a polynomial time algorithm that either (a) finds a matching of size $k+1$ in G; or (b) finds a crown decomposition (C, H, R) of G.*

This gives rise to the following reduction.

4. If $|V(G)| \geq 3k+1$, then apply the algorithm of Proposition 2.7 to G. If case (a) applies, then let (G', k) be a trivial *no*-instance; let G' be two vertices connected by an edge, and let $k = 0$. If case (b) applies, then "chop off the head": Remove C and H from G, and reduce k by $|H|$. Let $G' := G - (C \cup H)$, and $k' := k - |H|$.

In case (a), then the reduction is safe because at least $k+1$ vertices are required to cover the found matching, which is too much. In case (b), safeness of the reduction follows by Lemma 2.6. Finally, observe that a reduced kernel has at most $3k$ vertices, since the crown reduction is not applicable.

2.3.5 Integer Linear Programming

In this section, we introduce another way of finding a crown decomposition which will enables us to give a kernel for Vertex Cover on at most $2k$ vertices. Rather than using the algorithm of Proposition 2.7 as in the previous section, we utilize the technique of *linear programming* (LP), as originally shown by Chen, Kanj and Jia [3] using the Nemhauser-Trotter theorem [26].

In linear programming, one specifies a number of linear inequality constraints over some variables, and an objective function to be minimized or maximized. Solving a linear program optimally is polynomial time solvable, for instance by the simplex method [9]. For a deeper introduction to linear programming, a textbook in algorithms should be consulted [6, 10].

While the Vertex Cover can not be formulated as a linear program, it can be formulated as an *integer* linear program (ILP), where there is an additional requirement that the variable assignments are integers. Let there be one variable x_i for each vertex v_i of a graph G. We formulate the following ILP for vertex cover:

$$\begin{array}{rll} \text{Objective function:} & \text{Minimize } \displaystyle\sum_{i=1}^{n} x_i & \\ \text{Constraints:} & x_i \in \mathbb{Z} & \text{for every } 1 \leq i \leq n \\ & 0 \leq x_i \leq 1 & \text{for every } 1 \leq i \leq n \\ & x_i + x_j \geq 1 & \text{if } v_i v_j \in E(G) \end{array}$$

We begin by observing that any assignment for the variables $x_1, x_2, \dots x_n$ which satisfy the constraints will yield a corresponding vertex cover in the graph G; we let the vertices corresponding to variables set to 1 be included in the vertex cover. Observe that a variable assignment which minimize the objective function subject to these constraints will correspond to a minimum vertex cover; thus we conclude that solving ILP's is NP-hard.

However, by relaxing the requirement that the variables are integers, we are able to solve the resulting LP optimally in polynomial time. In any optimal solution to the LP, each variable gets assigned a real value between 0 and 1. We can then partition the vertices of the graph according to the value assigned to their respective variable:

$$C = \{v_i \mid x_i < \frac{1}{2}\} \qquad H = \{v_i \mid x_i > \frac{1}{2}\} \qquad R = \{v_i \mid x_i = \frac{1}{2}\}$$

We will now argue that (C, H, R) is in fact a crown decomposition. First, observe that there can be no edge between two vertices $v_i, v_j \in C$, since then there would be an unsatisfied constraint $x_i + x_j \geq 1$. By the same argument, there can be no edge between a vertex of C and a vertex of R. It remains to show that there is a matching in the bipartite graph between C and H which saturates H. In order to do so, we will rely on Hall's marriage theorem.

Proposition 2.8 (Hall's marriage theorem [17]). *Let B be an undirected bipartite graph with bipartition (C, H). The graph B has a matching saturating H if and only if for all $H' \subseteq H$, we have $|N_B(H')| \geq |H'|$.*

Assume for the sake of contradiction that there is not a matching in the bipartite graph (C, H) which saturates H. Then by Proposition 2.8, there exists some set $H' \subseteq H$ such that $|N_G(H') \cap C| < |H'|$. Let ϵ be some strictly positive real value such that for every vertex $v_i \in H'$, it holds that $x_i - \epsilon \geq \frac{1}{2}$. Now observe that subtracting ϵ from every variable corresponding to a vertex in H' and adding ϵ to every variable corresponding to a vertex in $N_G(H') \cap C$ will yield a feasible solution to the LP. Moreover, the value of the objective function is strictly smaller than before, since $|N_G(H') \cap C| < |H'|$. But the LP was already solved optimally, and we have hence reached a contradiction. We thus conclude that there is a matching between C and H which saturates H, and that (C, H, R) is indeed a crown decomposition.

Whenever solving the LP yields non-empty C and H, we can then safely include H in the vertex cover and remove C by Lemma 2.6, and repeat. When at some point the LP yields $C = \emptyset$ and $H = \emptyset$, then observe that every variable has value $\frac{1}{2}$. Because the objective function of the linear program $val(\text{LP})$ is always smaller than or equal to the value of the integer version $val(\text{ILP})$ due to there being less constraints, we get that $\frac{V(G)}{2} = val(\text{LP}) \leq val(\text{ILP}) = \textsc{vc}(G)$. Here, $\textsc{vc}(G)$ is the size of a minimum vertex cover, and we then conclude that if there are strictly more than $2k$ vertices in G, then it is not possible to find a vertex cover of size k.

There are plenty more fpt algorithms solving the Vertex Cover problem, and the currently fastest algorithm by Chen et al. [4] runs in time $\mathcal{O}(1.2738^k + kn)$ and makes use of a combination of different techniques, including several reductions. In this section we saw a kernel for Vertex Cover on $2k$ vertices, and it is believed that this is optimal [3].

However, all the examples we have seen in this section has used the size of the solution k as the parameter; the runtime of the algorithms are given as a function of k, and the bound on the kernel size has been given in terms of the solution size k. But a vertex cover can be very large, so even though linear kernels are great, it may be helpful to consider parameters that are structurally smaller. This is the topic for the next section.

2.4 Ecology of Parameters

So far we have seen parameterized algorithms for Vertex Cover which use the size of the solution as a parameter. However, the definition of parameterized problems and fpt-algorithms are not limited to this; anything quantifiable can be a parameter. While not every quantifiable parameter is helpful in the sense that they yield fpt-algorithms, some are. For the same problem, different parameters can take different numeric values, and everything else being equal, it would be preferable to solve the problem with a running time depending on the smallest parameters available.

For instance, consider the two graph parameters vc, the size of a minimum vertex cover, and d1m, the size of a minimum degree-1 modulator. For a graph G, observe that every vertex cover $S \subseteq V(G)$ is also a degree-1 modulator, since the maximum degree of $G - S$ is 0, which is indeed less than or equal to 1. Thus we know that $\text{vc}(G) \geq \text{d1m}(G)$. The difference can also be arbitrarily large; the graph which consists of n disjoint paths each on two vertices have a minimum vertex cover of size n, but a minimum degree-1 modulator of size 0. We say that a degree-1 modulator is *structurally smaller* than a vertex cover, and is thus a stronger parameter.

Making similar observations for various graph parameters, we find that there exists a partial ordering on the set of all possible graph parameters. For the parameters we consider in this thesis, we get the partial order shown in Figure 2.3. We observe that, as we expect, a degree-2 modulator (D2M) is structurally smaller than a degree-1 modulator, and hence also a vertex cover. The size of a minimum feedback vertex set (FVS) is not comparable to the size of a minimum degree-2 modulator, but both are at least as large as a minimum pseudoforest modulator (PFM).

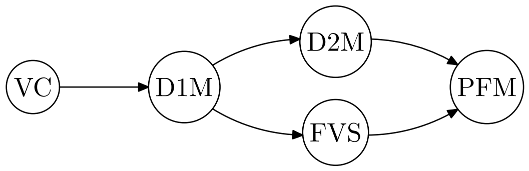


Figure 2.3: The partial order ($\geq$) on the graph parameters considered in this thesis.

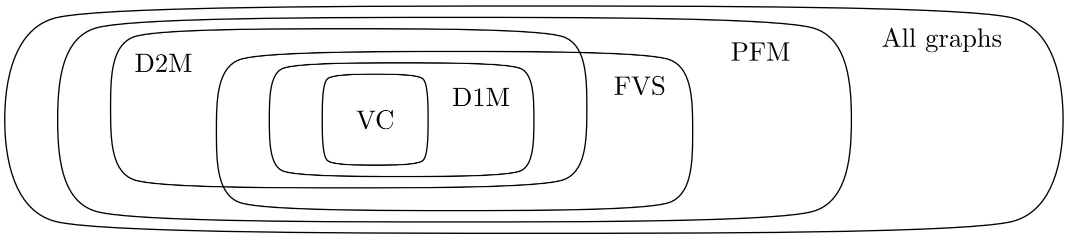


Figure 2.4: A Venn diagram showing where graph classes with bounded parameters may reside.

For a given graph G, it may therefore be beneficial to use an algorithm whose runtime is measured in the strongest parameter available. However, the advantage of selecting a stronger parameter can also be seen in a slightly different light, by considering which graph classes become tractable with the parameter. In Figure 2.4 we see that every graph class which have bounded size on their minimum vertex cover also has a bounded size of their minimum degree-1 modulator. All the parameters are similarly related as before, the only difference being that we now are talking about *graph classes* rather than a single graph. By selecting a stronger parameter which has an fpt-algorithm/kernel, more classes of graphs become tractable.

However, choosing a stronger parameter may also come with a drawback, if the algorithms at hand have a worse dependency on the stronger parameter than on the weaker one. For instance, we saw that Vertex Cover has a *linear* kernel parameterized by the size of the minimum vertex cover, but the best known kernel parameterized by the size of a minimum feedback vertex cover is on a *cubic* number of vertices [20]. Thus, for the kernel to be as small as possible, the practitioner would need to compare exactly how much smaller the parameter is for the problem (or class of problems) in question before blindly deciding to use the stronger parameter. One can also combine different kernelization algorithms in series or in parallel to get the best of all worlds.

It is then useful to have multiple parameters to choose from when deciding how to best approach a problem. How various parameters influence the complexity of different problems is called the *ecology* of parameters [12]. As a part of the parameter ecology program [13, 24], Jansen et al. [22] classify the kernelization complexity of Feedback Vertex Set for a wide range of parameters. In particular, they show that Feedback Vertex Set admits a kernel on $\mathcal{O}(|X|^{10})$ vertices where the parameter $|X|$ is the size of a minimum pseudoforest modulator. In this thesis we do a similar exploration for Vertex Cover by demonstrating a polynomial kernel when parameterized by the size of a minimum degree-2 modulator or by the size of a minimum pseudoforest modulator. As a gentle introduction to the techniques in play, we will first show a cubic kernel for Vertex Cover parameterized by the size of a minimum degree-1 modulator.

2.5 Kernel Parameterized by Degree-1 Modulator

In this section, we will give a cubic kernel for Vertex Cover parameterized by the size of a minimum degree-1 modulator, which is a simplified version of the kernel parameterized by a feedback vertex set by Jansen and Bodlaender [20]. This section contains no new results, but serves as a gentle introduction to the framework we will use in the next chapters. Another polynomial kernel for this problem on $\mathcal{O}(|X|^2)$ vertices was also shown by Majumdar et al. [25]. We define the problem as follows:

Vertex Cover/Degree-1 Modulator (VC/D1M)
Input: A graph G, a degree-1 modulator vertex set $X \subseteq V(G)$ such that $\triangle(G - X) \leq 1$, and an integer k.
Parameter: Size of the degree-1 modulator $|X|$.
Question: Does G contain a vertex cover of size at most k?

Note that we here state the problem with a degree-1 modulator given as part of the input, but this condition can be omitted, seeing that a degree-1 modulator can be constant-factor approximated (see for instance Appendix A).

For ease of presentation, we will first develop a kernel for Independent Set parameterized by the size of a minimum degree-1 modulator, and then by the immediate correspondence between the Vertex Cover and Independent Set problems the kernel for VC/D1M will follow. For the remainder of this section, we will thus focus on the following problem:

Independent Set/Degree-1 Modulator (IS/D1M)
Input: A graph G, a degree-1 modulator vertex set $X \subseteq V(G)$ such that $\triangle(G - X) \leq 1$, and an integer k.
Parameter: Size of the degree-1 modulator $|X|$.
Question: Does G contain an independent set of size at least k?

Throughout the chapter, let $F := G - X$ be the induced subgraph remaining after the modulator X has been removed from G. Note that F has maximum degree 1, and is as such a forest containing only isolated vertices and paths on two vertices.

Definition 2.9 (Conflicts [20, Definition 3])**.** Let (G, X, k) be an instance of IS/D1M where $F' \subseteq F$ is a subgraph of the forest F and $X' \subseteq X$ is a subset of the modulator X. Then the number of *conflicts* induced by X' on F' is defined as $\textsc{Conf}_{F'}(X') := \alpha(F') - \alpha(F' - N_G(X'))$.

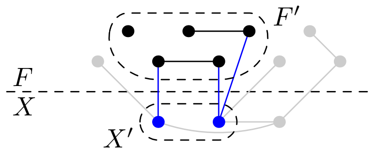


Figure 2.5: Conflicts: In the figure, we observe that $\alpha(F') = 3$, and $\alpha(F' - N_G(X')) = 2$. Hence, we get that $\textsc{Conf}_{F'}(X') = 1$. In other words, the number of conflicts induced by X' on F' is 1.

Choosing X' to be in an independent set I of G may prevent some vertices in F' from being included in same set I. In particular, no vertex $v \in V(F') \cap N_G(X')$ can be chosen to be in I. In light of this, the term $\textsc{Conf}_{F'}(X')$ can be understood as the price one has to pay in F' by choosing to include X' in an independent set. See Figure 2.5.

2.5.1 Reduction Rules

We introduce here the reduction rules. Each reduction receives as input an instance (G, X, k) of Independent Set/Degree-1 Modulator, and outputs an equivalent instance (G', X', k'). Reduction rules will be applied exhaustively starting with lower number rules.

Reduction 2.1 ([20, Reduction 1])**.** If there is a vertex $v \in X$ such that $\textsc{Conf}_F(\{v\}) \geq |X|$, then delete v from the graph G and from the set X. We let $G' := G - v$, $X' := X - v$ and $k' := k$.

Reduction 2.2 ([20, Reduction 2])**.** If there are distinct vertices $u, v \in X$ with $uv \notin E(G)$ for which $\textsc{Conf}_F(\{u, v\}) \geq |X|$, then add the edge uv to G. We let $G' := (V(G), E(G) \cup \{uv\})$, $X' := X$ and $k' := k$.

The correctness of the two above rules can be established by the following lemma.

Lemma 2.10 ([20, Lemma 2])**.** *Let (G, X, k) be an instance of* IS/D1M. *If $X_I \subseteq X$ is a subset of the degree-1 modulator such that $\textsc{Conf}_F(X_I) \geq |X|$, then there is a MIS for G that does not contain all the vertices of X_I.*

Proof. Assume that $I' \subseteq V(G)$ is a maximum independent set which contains X_I. We will show that then there is also another independent set I such that $|I| \geq |I'|$ and which satisfies $X_I \nsubseteq I$.

Because $\textsc{Conf}_F(X_I) \geq |X|$, we have that $\alpha(F) \geq |X| + \alpha(F - N_G(X_I))$. Now consider the independent set I'. Some of its vertices are in X, however no more than $|X|$. The remainder of vertices of I' are in $V(F)$, however no more than $\alpha(F - N_G(X_I))$. Thus $|X| + \alpha(F - N_G(X_I)) \geq |I'|$. But then $\alpha(F) \geq |I'|$, and we see that a MIS of the pseudoforest F satisfies the requirements of the lemma. $\square$

Reduction 2.3 ([20, Reduction 3])**.** If there exists a connected component T in F such that for all independent subsets $X_I \subseteq X$ with $|X_I| \leq 2$ there is no conflict induced by X_I on T, i. e. $\textsc{Conf}_T(X_I) = 0$, then remove T from G and reduce k by 1. We let $G' := G - T$, $X' := X$ and $k' := k - 1$.

In order to prove safeness of the above reduction, we will here provide an auxiliary lemma. The statement of the lemma originates in the paper by Jansen and Bodlaender [20, Lemma 3], however the proof is much simpler in our case. The same goes for the safeness proof itself, which follows afterwards.

Lemma 2.11. *Let (G, X, k) be an instance of* IS/D1M, *where $T \subseteq F$ is a connected component of the forest F. If there is some independent set $X_I \subseteq X$ such that $\textsc{Conf}_T(X_I) > 0$, then there is also an independent set $X'_I \subseteq X_I$ with $|X'_I| \leq 2$ such that $\textsc{Conf}_T(X'_I) > 0$.*

Proof. Assume that there exists some independent set $X_I \subseteq X$ such that $\textsc{Conf}_T(X_I) > 0$. There are two cases. In the first case, T is a single vertex. Then $\textsc{Conf}_T(X_I) > 0$ implies that the vertex in T has at least one neighbor in X_I. Pick an arbitrary vertex $x \in X_I \cap N_G(T)$, and observe that $\textsc{Conf}_T(\{x\}) > 0$. In the second case, T is a path on two vertices. Then $\textsc{Conf}_T(X_I) > 0$ implies that both vertices of T has a neighbor in X_I. We pick two arbitrary (possibly non-distinct) vertices $x_1, x_2 \in X_I$ such that x_1 is neighbor to one of the vertices of T and x_2 is neighbor to the other one. Observe that $\textsc{Conf}_T(\{x_1, x_2\}) > 0$. This concludes the proof. □

Lemma 2.12 ([20, Lemma 4])**.** *Reduction 2.3 is safe. Let (G, X, k) be an instance of* IS/D1M *to which Reduction 2.3 is applicable, and let (G', X', k') be the output instance resulting from the reduction. Then (G, X, k) is a yes-instance if and only if (G', X', k') is a yes-instance.*

Proof. Since X is a degree-1 modulator, T must be either a single vertex or a path on two vertices. If T is a single vertex, then this implies that there is no edge between X and T, since all singleton subsets of X are independent. Thus the vertex of T is isolated in the graph G. It can then be safely included in the independent set and be forgotten.

If T is a path on two vertices, then the forward direction is trivial, since there is an edge between the elements of T. For the backward direction, let I' be an independent set of G' with $|I'| \geq k'$. Note that I' is also independent in G. By the contrapositive of Lemma 2.11, we have that every independent set $X_I \subseteq X$ yields $\textsc{Conf}_T(X_I) = 0$. In particular, it must be the case that $\textsc{Conf}_T(I' \cap X) = 0$. Observe that there is then a vertex of T which has no edge to the elements of I', and thus adding this vertex to I' to obtain I will yield a sufficiently big independent set of G. □

2.5.2 Bound on Size of Reduced Instances

In this section we will prove that the number of vertices in a reduced instance (G, X, k) is at most $\mathcal{O}(|X_0|^3)$ where $|X_0|$ is the size of the modulator in the original problem (G_0, X_0, k_0). Note that none of the reduction rules increase $|X_0|$, so any bound in terms of $|X|$ is a better or equally good bound as a similar bound in $|X_0|$. For our purposes, a much simpler proof than what was done by Jansen and Bodlaender [20] is sufficient. This is due to the small size of every connected component in F when X is a degree-1 modulator.

Lemma 2.13. *Let (G, X, k) be a reduced instance of* IS/D1M. *Let $\mathcal{C}_F$ denote the collection of all connected components $T \subseteq F$. Then $|\mathcal{C}_F| \leq |X|^3 + |X|^2$, i. e. the number of connected components in F is at most $|X|^3 + |X|^2$.*

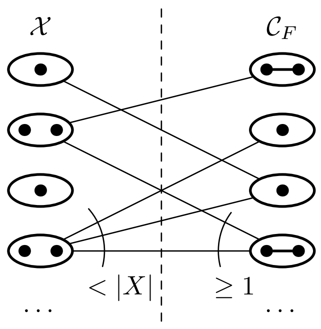


Figure 2.6: The bipartite graph B: There is an edge between $X' \in \mathcal{X}$ and $T \in \mathcal{C}_F$ if and only if X' induce a conflict in T.

Proof. Let $\mathcal{X}$ be the collection of all subsets $X' \subseteq X$ where X' is independent and has $|X'| \leq 2$. Now consider the bipartite graph B between $\mathcal{X}$ and the connected components $\mathcal{C}_F$ where there is an edge between $X' \in \mathcal{X}$ and $T \in \mathcal{C}_F$ if and only if $\textsc{Conf}_T(X') > 0$. Because Reduction 2.3 is not applicable, every connected component $T \in \mathcal{C}_F$ will have at least one edge incident to it in B. Thus any bound on the number of edges in B will also be a bound for $|\mathcal{C}_F|$.

Since every $X' \in \mathcal{X}$ is independent and has size at most two, we know that $\textsc{Conf}_F(X') < |X|$, or else one of Reductions 2.1 or 2.2 would be applicable to X', contradicting that the instance is reduced. Since conflicts induced by X' on F in different connected components of F are distinct, each $X' \in \mathcal{X}$ is incident to less than $|X|$ connected components in B. See Figure 2.6. Since $|\mathcal{X}| \leq |X|^2 + |X|$, we get that $|\mathcal{C}_F| \leq (|X|^2 + |X|) \cdot |X|$. This concludes the proof. □

We are now ready to prove the main theorem of this section. This is a simplified version of the kernelization result by Jansen and Bodlaender for the Independent Set/Feedback Vertex Set problem [20, Theorem 2].

Theorem 2.14. Independent Set/Degree-1 Modulator *admits a kernel on $\mathcal{O}(|X|^3)$ vertices: There is a polynomial time algorithm that transforms an instance (G, X, k) to an equivalent instance (G', X', k') such that*

- $k' \leq k$
- $|X'| \leq |X|$
- $|V(G')| \leq 2|X|^3 + 2|X|^2 + |X|$

Proof. Let (G', X', k') be a reduced instance after applying the reductions of this section to the original instance (G, X, k). First observe that there are no reductions which increase k or $|X|$. It remains to show the bound for $|V(G)|$. By Lemma 2.13, there are at most $|X|^3 + |X|^2$ connected components of F. Since each component has at most two vertices, $|V(F)| \leq 2|X|^3 + 2|X|^2$. Seeing that $|V(G)| = |X| + |V(F)|$, we obtain that $|V(G)| \leq 2|X|^3 + 2|X|^2 + |X|$. □

Corollary 2.15. Vertex Cover/Degree-1 Modulator *has kernel on* $\mathcal{O}(|X|^3)$ *vertices: There is a polynomial time algorithm that transforms an instance* (G, X, k) *to an equivalent instance* (G', X', k') *such that* $|V(G')| \leq 2|X|^3 + 2|X|^2 + |X|$.

Proof. First, we transform the instance (G, X, k) of VC/D1M to an instance $(G, X, |V(G)| - k)$ of IS/D1M. Then we apply Theorem 2.14 to obtain $(G', X', |V(G')| - k')$, which we can transform back to an instance of VC/D1M (G', X', k'). Since the transformations between VC/D1M and IS/D1M preserve both the graph and the modulator, the bounds on the number of vertices in Theorem 2.14 also holds for VC/D1M. $\square$

Chapter 3

Kernel for Vertex Cover Parameterized by Degree-2 Modulator

This chapter gives a kernel for Vertex Cover parameterized by the size of a minimum degree-2 modulator X on $\mathcal{O}(|X|^7)$ vertices. Another polynomial kernel for this problem on $\mathcal{O}(|X|^5)$ vertices was independently shown by Majumdar et al. [25]. We define the problem as follows:

Vertex Cover/Degree-2 Modulator (VC/D2M)
Input: A graph G, a degree-2 modulator vertex set $X \subseteq V(G)$ such that $\triangle(G - X) \leq 2$, and an integer k.
Parameter: Size of the degree-2 modulator $|X|$.
Question: Does G contain a vertex cover of size at most k?

Note that we here state the problem with a degree-2 modulator given as part of the input, but this condition can be omitted, seeing that a degree-2 modulator can be constant-factor approximated (see for instance Appendix A).

For ease of presentation, we will first develop a kernel for Independent Set parameterized by the size of a minimum degree-2 modulator, and then by the immediate correspondence between the Vertex Cover and Independent Set problems the kernel for VC/D2M will follow. For the remainder of this section, we will thus focus on the following problem:

Independent Set/Degree-2 Modulator (IS/D2M)
Input: A graph G, a degree-2 modulator vertex set $X \subseteq V(G)$ such that $\triangle(G - X) \leq 2$, and an integer k.
Parameter: Size of the degree-2 modulator $|X|$.
Question: Does G contain an independent set of size at least k?

3.1 Tools

Throughout the chapter, let $F := G - X$ be the induced subgraph remaining after the modulator X has been removed from G. Note that F has maximum degree two, and is as such a collection of paths and cycles. We will for completeness restate unconventional definitions here.

Definition 3.1 (Conflicts [20, Definition 3])**.** (Conflicts) Let (G, X, k) be an instance of IS/D2M where $F' \subseteq F$ is a subgraph of F and $X' \subseteq X$ is a subset of the modulator X. Then the number of *conflicts* induced by X' on F' is defined as $\textsc{Conf}_{F'}(X') := \alpha(F') - \alpha(F' - N_G(X'))$.

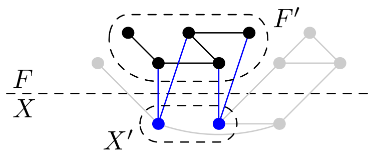


Figure 3.1: Conflicts: In the figure, we observe that $\alpha(F') = 3$, and $\alpha(F' - N_G(X')) = 1$. Hence, we get that $\textsc{Conf}_{F'}(X') = 2$. In other words, the number of conflicts induced by X' on F' is 2.

Choosing X' to be in an independent set I of G may prevent some vertices in F' from being included in same set I. In particular, no vertex $v \in V(F') \cap N_G(X')$ can be chosen to be in I. In light of this, the term $\textsc{Conf}_{F'}(X')$ can be understood as the price one has to pay in F' by choosing to include X' in an independent set. Observe that $\textsc{Conf}_{F'}(X')$ is polynomial time computable when the independence number is.

Next, we will introduce *chunks*. This concept originates in the paper by Jansen and Bodlaender [20, Definition 2], but a is altered for our purposes in two ways: The size of the chunks are here of size at most three (as opposed to two), and we have additionally introduced the requirement that a chunk cannot induce a conflict on F which is of size $|X|$ or larger.

Definition 3.2 (Chunks)**.** Let (G, X, k) be an instance of IS/D2M. A set $X' \subseteq X$ is a *chunk* if the following hold:

- X' is independent in G,
- the size of X' is between 1 and 3, i. e. $1 \leq |X'| \leq 3$, and
- the number of conflicts in F induced by X' is less than $|X|$, i. e. $\textsc{Conf}_F(X') < |X|$.

We let $\mathcal{X}$ be the collection of all chunks in X.

The collection of chunks $\mathcal{X}$ can be seen as all suitable candidate subsets of size at most three from X to be included in a maximum independent set I of G. The idea is that I may contain a chunk as a subset, but it is never necessary to include a subset $X' \subseteq X$ of size at most three which is *not* a chunk. This will allow us to discard potential solutions containing non-chunk subsets of X with size at most three. In order for this intuition to hold, we provide the following lemma, originally by Jansen and Bodlaender though slightly altered to fit our purposes. For completeness, we also provide a proof.

Lemma 3.3 ([20, Lemma 2])**.** *Let (G, X, k) be an instance of* IS/D2M *where $\alpha(G) \geq k$. Then there exists an independent set I of G such that $|I| \geq k$ and for all subsets $X' \subseteq X \cap I$ it holds that* $\textsc{Conf}_F(X') < |X|$.

Proof. Assume that $I' \subseteq V(G)$ is an independent set for which there exists some $X' \subseteq I' \cap X$ such that $\textsc{Conf}_F(X') \geq |X|$. We will show that then there exists another independent set I such that $|I| \geq |I'|$ and for all subsets $X'' \subseteq X$ with $\textsc{Conf}_F(X'') \geq |X|$, it holds that X'' is not a subset of I.

Because $\textsc{Conf}_F(X') \geq |X|$, we have that $\alpha(F) \geq |X| + \alpha(F - N_G(X'))$. Now consider the independent set I'. Some of its vertices are in X, however no more than $|X|$. The remainder of vertices of I' are in $V(F)$, however no more than $\alpha(F - N_G(X'))$. Thus $|X| + \alpha(F - N_G(X')) \geq |I'|$. But then $\alpha(F) \geq |I'|$, and we see that a maximum independent set of the pseudoforest F satisfies the requirements of the lemma. $\square$

Definition 3.4. (Anchor triangle) Let (G, X, k) be an instance of IS/D2M. Let P be a triangle in F with $V(P) = \{p_1, p_2, p_3\}$. Then P is an *anchor triangle* if there exists three distinct vertices $x_1, x_2, x_3 \in X$ such that:

- $N_G(p_1) = \{p_2, p_3, x_1\}$
- $N_G(p_2) = \{p_1, p_3, x_2\}$
- $N_G(p_3) = \{p_1, p_2, x_3\}$

An anchor triangle is *redundant* if there is also another anchor triangle with the same open neighborhood in G, and it is *non-redundant* if there is not.

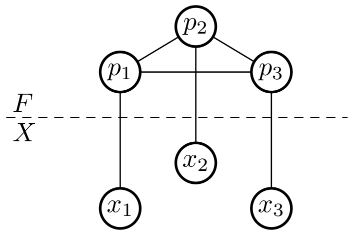


Figure 3.2: The connected component $P \subseteq F$ with vertices $V(P) = \{p_1, p_2, p_3\}$ is an anchor triangle for the triple $\{x_1, x_2, x_3\} \subseteq X$.

Definition 3.5. (Unnecessary triple) Let (G, X, k) be an instance of IS/D2M, and let $X_3 \subseteq X$ be a triple, i. e. a set containing three distinct vertices. Then X_3 is an *unnecessary triple* if there exists an anchor triangle P in F such that $N_G(P) = X_3$.

The fact that a triple $X_3 \subseteq X$ is unnecessary as defined above should intuitively be understood with respect to constructing an independent set. If a triple X_3 is unnecessary then there exists a maximum independent set for G which does not contain all of X_3. This intuition is supported by the next lemma.

Lemma 3.6. *Let (G, X, k) be an instance of* IS/D2M*. If there exists an independent set of size at least k in G, then there exists an independent set I of G with $|I| \geq k$ containing no unnecessary triple $X_3 \subseteq X$.*

Proof. We will prove the lemma by constructing the required independent set I satisfying the properties, assuming we have an arbitrary independent set I' of size at least k. By definition, for each unnecessary triple $X_3 \subseteq X \cap I'$, there is at least one anchor triangle $P \subseteq F$ such that $N_G(P) = X_3$. Since $X_3 \subseteq I'$, we have that no vertex of P is in I'. Let $p_u \in V(P)$ be an arbitrary vertex of P, and let $u \in X_3$ be the neighbor of p_u in X_3. Observe that we can replace u by p_u in I', and still have I' be an independent set of the same size. We do this for every unnecessary triple $X_3 \subseteq X \cap I'$ to obtain I, which then satisfies the requirement of the lemma. $\square$

3.2 Reduction Rules

We introduce here the reduction rules. Each reduction receives as input an instance (G, X, k) of Independent Set/Degree-2 Modulator, and outputs an equivalent instance (G', X', k'). Reductions 3.1, 3.2, 3.4 and 3.5 originates in the result by Jansen and Bodlaender [20], though Reduction 3.4 is altered to fit the context of a graph with maximum degree 2, which required significant changes to the proof. Reduction rules will be applied exhaustively, and a lower number rule is always applied before a higher number rule if at any point they are both applicable.

Reduction 3.1 ([20, Reduction 1])**.** If there is a vertex $v \in X$ such that $\textsc{Conf}_F(\{v\}) \geq |X|$, then delete v from the graph G and from the set X. We let $G' := G - v$, $X' := X - v$ and $k' := k$.

Reduction 3.1 is safe due to Lemma 3.3.

Reduction 3.2 ([20, Reduction 2])**.** If there are distinct vertices $u, v \in X$ with $uv \notin E(G)$ for which $\textsc{Conf}_F(\{u, v\}) \geq |X|$, then add edge uv to G. We let $G' := (V(G), E(G) \cup \{uv\})$, $X' := X$ and $k' := k$.

Reduction 3.2 is safe, because there always exists a maximum independent set I in G leaving at least one of $\{u, v\}$ outside I by Lemma 3.3. Adding uv to $E(G)$ thus does not reduce the size of the maximum independent set. The backward direction is trivial, since removing an edge does not reduce the maximum size of an independent set.

Reduction 3.3. If there are distinct $u, v, w \in X$ such that $\textsc{Conf}_F(\{u, v, w\}) \geq |X|$, the set $\{u, v, w\}$ is independent in G, and for which there is no anchor triangle P with $N_G(P) = \{u, v, w\}$, then add an anchor triangle $P' = \{p_u, p_v, p_w\}$ to the graph such that $N_G(P') = \{u, v, w\}$, and increase k by one. Let $V(G') := V(G) \cup \{p_u, p_v, p_w\}$ and let $E(G') := E(G) \cup \{p_u p_v, p_u p_w, p_v p_w, p_u u, p_v v, p_w w\})$. Further, let $X' := X$ and let $k' := k + 1$.

Note that Reduction 3.3 makes the triple $\{u, v, w\} \subseteq X$ unnecessary as defined in Definition 3.5.

Lemma 3.7. *Reduction 3.3 is safe. Let (G, X, k) be an instance of* IS/D2M *to which Reduction 3.3 is applicable, and let (G', X', k') be the output instance resulting from the reduction. Then (G, X, k) is a yes-instance if and only if (G', X', k') is a yes-instance.*

Proof. For the forward direction of the proof, assume (G, X, k) is a *yes*-instance, let $u, v, w \in X$ be the elements which triggers the reduction, and let $V(P') = \{p_u, p_v, p_w\}$ be the vertices of the added anchor triangle in the output instance. Let I be an independent set of G with $|I| \geq k$. By Lemma 3.3, we assume that at least one of u, v, w is not in I. Without loss of generality (by symmetry), let $u \notin I$. For the output instance, observe that $I' := I \cup \{p_u\}$ is an independent set of G' with size $|I'| = |I| + 1 \geq k + 1 = k'$, which makes (G', X', k') a *yes*-instance.

For the backward direction, assume that (G', X', k') is a *yes*-instance, and let P' be the added anchor triangle in the output instance. Let I' be an independent set of G' with $|I'| \geq k'$. Because P' induces a triangle in G', at most one vertex of P' is in I'. Thus, $I := I' \setminus V(P')$ is an independent set of G with $|I| \geq |I'| - 1 \geq k' - 1 = k$, which makes (G, X, k) a *yes*-instance. □

The next reduction rule originates in the paper by Jansen and Bodlaender [20, Reduction 3], but it has been altered to fit our purposes. Because chunks are defined differently, and because F now contains cycles, the proof of safeness of this reduction is in some aspects very different.

Reduction 3.4. If there exists a connected component P in F which is not a non-redundant anchor triangle, and for every chunk $X' \in \mathcal{X}$ there is no conflicts induced by X' on P, i. e. $\textsc{Conf}_P(X') = 0$, then remove P from G and reduce k by $\alpha(P)$. We let $G' := G - P$, $X' := X$ and $k' := k - \alpha(P)$.

To prove that Reduction 3.4 is safe, we will rely on the following lemma, which states that any connected component with maximum degree at most two (paths and cycles), has a small (at most size three) obstruction in terms of obtaining a maximum independent set.

Lemma 3.8. *Let P be a connected component with maximum degree two, and let Z be a set of vertices such that $\alpha(P) > \alpha(P - Z)$. Then there exist three (possibly non-distinct) vertices $u, v, w \in Z \cap V(P)$ such that $\alpha(P) > \alpha(P - \{u, v, w\})$.*

Proof. Since P has maximum degree 2, it is either a path or a cycle. Moreover, the number of vertices in P is either even or odd. We consider each case separately:

- P is an odd path. Then there is a unique maximum independent set for P containing every odd-indexed vertex of P. Since $\alpha(P) > \alpha(P - Z)$, at least one odd-indexed vertex $u \in V(P)$ is in Z. See that also $\alpha(P) > \alpha(P - u)$.
- P is an even path. Then there are exactly two distinct maximum independent sets; one containing all the odd-indexed vertices of P and one containing all the even-indexed vertices of P. Since $\alpha(P) > \alpha(P - Z)$, at least one odd-indexed vertex $u \in V(P)$ and one even-indexed vertex $v \in V(P)$ are also in Z. See that also $\alpha(P) > \alpha(P - \{u, v\})$.
- P is an even cycle. Then there are exactly two distinct maximum independent sets, and by an analogous argument to the previous case there exists $u, v \in Z \cap V(P)$ such that $\alpha(P) > \alpha(P - \{u, v\})$.

- P is an odd cycle. Pick an arbitrary vertex $u \in Z \cap V(P)$. Observe that $\alpha(P) = \alpha(P - u)$. Then we must have that $\alpha(P - u) > \alpha((P - u) - Z)$. But since $P - u$ is an even path, we have already shown in a previous case that there exist two vertices $v, w \in (Z \cap V(P - u))$ such that $\alpha(P - u) > \alpha(P - \{u, v, w\})$. Then it follows that $\alpha(P) > \alpha(P - \{u, v, w\})$. □

We are now able to make the following observation:

Observation 3.9. Let $P \subseteq F$ be a connected component in F and let $X' \subseteq X$ be an independent set such that $\textsc{Conf}_P(X') > 0$. Then there exists some $X'' \subseteq X'$ with $1 \leq |X''| \leq 3$ such that $\textsc{Conf}_P(X'') > 0$.

Proof. We see that the observation is true, since by Lemma 3.8 there exist $u, v, w \in N_G(X') \cap V(P)$ such that $\alpha(P) > \alpha(P - \{u, v, w\})$. Then for each element u, v, w, pick an arbitrary neighbor $x_u, x_v, x_w \in X'$ (they need not be distinct) to form the set $X'' := \{x_u, x_v, x_w\}$. See that then $\textsc{Conf}_P(X'') > 0$. □

We are now equipped to prove safeness of Reduction 3.4.

Lemma 3.10. *Reduction 3.4 is safe. Let (G, X, k) be an instance of* IS/D2M *to which Reduction 3.4 is applicable, and let (G', X', k') be the output instance resulting from the reduction. Then (G, X, k) is a yes-instance if and only if (G', X', k') is a yes-instance.*

Proof. Let $P \subseteq F$ be the connected component which triggered the reduction. For the forward direction of the proof, assume (G, X, k) is a *yes*-instance and let I be an independent set of G with size at least k. Let $I' := I \setminus V(P)$. Clearly I' is an independent set of G'. Now observe that $|I \cap V(P)| \leq \alpha(P)$, and thus $|I'| = |I| - |I \cap V(P)| \geq k - \alpha(P) = k'$. Hence (G', X', k') is a *yes*-instance.

For the backward direction, we assume that (G', X', k') is a *yes*-instance, and has an independent set I' of size at least k'. Because of Lemma 3.6 we can assume that I' contains no unnecessary triples. We want to show that we can always pick some independent set $I_P \subseteq V(P)$ with $|I_P| = \alpha(P)$ such that $I := I' \cup I_P$ is an independent set with size at least $k' + \alpha(P) = k$. Since I' and $V(P)$ are disjoint by construction, it will suffice to show that $\alpha(P - N_G(I')) \geq \alpha(P)$.

Assume for the sake of contradiction that $\alpha(P - N_G(I')) < \alpha(P)$. Since P was a connected component in F, all its neighbors $N_G(P)$ are in X. Thus we have that $\textsc{Conf}_P(X \cap I') > 0$. By Observation 3.9, we further have that there exists some $X'' \subseteq X \cap I'$ such that $1 \leq |X''| \leq 3$ and $\textsc{Conf}_P(X'') > 0$.

For any such X'', there are two cases. In the first case, $\textsc{Conf}_F(X'') < |X|$. Because X'' is also independent and has size at most three, it is a chunk of X in the input instance. This contradicts the preconditions for Reduction 3.4, so this case can not happen.

In the second case, $\textsc{Conf}_F(X'') \geq |X|$. But then one of Reductions 3.1, 3.2 or 3.3 would have previously been applied to X'', yielding it either unfeasible for an independent set or making it an unnecessary triple in the input instance. Because non-redundant anchor triangles are not chosen for removal by Reduction 3.4, X'' is also an unnecessary triple in the output instance, which contradicts that I' contains no unnecessary triples. This concludes the proof. □

The final reduction is originally by Jansen and Bodlaender, though slightly altered to make sure it does not destroy any triangles of F. This change is also required for the safeness of the reduction in the context of a degree-2 modulator as opposed to a feedback vertex set. The proof of safeness itself is not altered, except to account for the new definition of chunks, but is included for completeness.

Reduction 3.5 ([20, Reduction 4])**.** If there are distinct vertices $u, v \in V(F)$ which are adjacent in G, are not part of a triangle in F, and for which there is no chunk which induce a conflict on $\{u, v\}$, i. e. for all chunks $X' \in \mathcal{X}$, $\textsc{Conf}_{\{u,v\}}(X') = 0$, then reduce the graph as follows:

- Delete vertices u, v with their incident edges and decrease k by one.
- If u has a neighbor t in F which is not v, make it adjacent to $N_G(v) \cap X$.
- If v has a neighbor w in F which is not u, make it adjacent to $N_G(u) \cap X$.
- If the vertices t, w both exist and $tw \notin E(G)$, add the edge tw to the graph.

Formally, let the output instance be defined by the following:

$$\begin{aligned}
V(G') &:= V(G) \setminus \{u, v\} \\
E(G') &:= (E(G) \setminus \{xy \mid x \in \{u, v\}, y \in N_G[u] \cup N_G[v]\}) \\
&\qquad \cup \{xy \mid x \in N_F(u) \setminus \{v\}, y \in N_G(v) \cap X\} \\
&\qquad \cup \{xy \mid x \in N_F(v) \setminus \{u\}, y \in N_G(u) \cap X\} \\
&\qquad \cup \{xy \mid x \in N_F(v) \setminus \{u\}, y \in N_F(u) \setminus \{v\}\} \\
X' &:= X \\
k' &:= k - 1
\end{aligned}$$

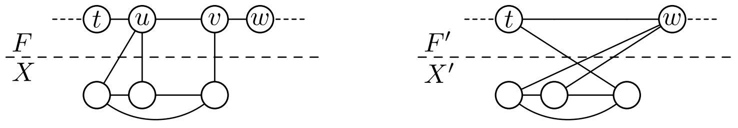


Figure 3.3: Reduction 3.5: Shrinking 2-paths with no conflict-inducing chunks ($k' = k - 1$).

Lemma 3.11 ([20, Lemma 5])**.** *Reduction 3.5 is safe. Let (G, X, k) be an instance to which rule 3.5 is applicable at vertices u, v, and let (G', X', k') be the instance resulting from the reduction. Then it holds that (G, X, k) is a yes-instance if and only if (G', X', k') is a yes-instance.*

Proof. Observe that for Reduction 3.5 to be applicable, $N_G(u) \cap X$ and $N_G(v) \cap X$ form a complete bipartite graph in G, i. e. $N_G(u) \cap N_G(v) \cap X = \emptyset$ and every vertex in $N_G(u) \cap X$ is adjacent to every vertex in $N_G(v) \cap X$. If this was not the case, then two non-adjacent (possibly non-distinct) vertices $x_u \in N_G(u) \cap X$ and $x_v \in N_G(v) \cap X$ would induce a conflict on $\{u, v\}$, i. e. $\textsc{Conf}_{\{u,v\}}(\{x_u, x_v\}) > 0$. Since none of the reductions 3.1 and 3.2 were applicable to $\{x_u, x_v\}$, it must be the case that $\textsc{Conf}_{\{u,v\}}(\{x_u, x_v\}) < |X|$, and $\{x_u, x_v\}$ is thus a chunk. This contradicts the preconditions for the reduction. Thus, for every independent set I of G, we can assume at least one of $N_G(v) \cap X \cap I$ and $N_G(v) \cap X \cap I$ is the empty set. The same will hold for the output graph G', since no edges between $N_G(u) \cap X$ and $N_G(v) \cap X$ were removed by the reduction. Notice also that since the reduction is not applied to triangles, t and w are distinct if they both exist. We now prove the two directions of the proof separately.

For the forward direction, let I be a maximal independent set of G with $|I| \geq k$. We first prove that one can assign exactly one of u, v to be in I. Observe that no independent set in G can contain both $\{u, v\}$ since they are adjacent. If I does not contain any of $\{u, v\}$, assume without loss of generality (by symmetry) that $N_G(u) \cap X \cap I = \emptyset$. Then I must contain t, and t must exists — if not, then $N_G[u] \cap I = \emptyset$, contradicting that I is maximal. But if t is in I, we can easily replace t by u in I, not changing the number of elements in I while still maintaining an independent set. This way, we ensure one of $\{u, v\}$ is in I.

We next show how to obtain an independent set I' of G' with $|I'| \geq k - 1 = k'$. Without loss of generality (by symmetry) we assume that u is in I. Let $I' := I \setminus \{u\}$. Then $|I'| \geq k - 1$. It remains to show that I' is an independent set of G'. It will suffice to show that Reduction 3.5 does not add any edge between vertices in I'. Observe that any edge added is either between t and $N_G(v) \cap X$ or it is between w and $N_G(u) \cap X$. Because $t \in N_G(u)$ and $N_G(u) \cap X \subseteq N_G(u)$, all edges added by the transformation have at least one endpoint in $N_G(u)$; this endpoint is outside of both I and I'.

For the backward direction of the proof, let I' be an independent set of G' with $|I'| \geq k' = k - 1$. We show how to obtain an independent set I for G with size at least k. It is not hard to see that I' is also an independent set of G. Depending on how I' interacts with G, we will build I in different ways.

- If t exists and $t \in I'$, then let $I := I' \cup \{v\}$. To prove that I is an independent set, we show that $N_G(v) \cap I' = \emptyset$ by showing that $\{u, w\} \cap I' = \emptyset$ and $N_G(v) \cap I' \cap X = \emptyset$. Clearly, $u \notin I'$, because $u \notin V(G')$. Because $t \in I$, it must hold that $w \notin I'$ as t and w are neighbors in G' if w exists. Finally, every vertex in $N_G(v) \cap X$ was a neighbor of t in G', so they can not be in I' either.
- If w exists and $w \in I'$, then let $I := I' \cup \{u\}$. The correctness argument is symmetric to that of the previous case.

In the remaining cases, we know that $\{t, w\} \cap I' = \emptyset$. Since at least one of $N_G(v) \cap X \cap I'$ and $N_G(v) \cap X \cap I'$ are empty, there are two cases left.

- If $N_G(u) \cap X \cap I' = \emptyset$, let $I := I' \cup \{u\}$. Since $t \notin I'$ and $v \notin I'$, we have that $N_G(u) \cap I' = \emptyset$, and thus I is still an independent set.
- If $N_G(v) \cap X \cap I' = \emptyset$, let $I := I' \cup \{v\}$. The correctness argument is symmetric to that of the previous case. □

3.3 Bound on Size of Reduced Instances

In this section we will prove that the number of vertices in a reduced instance (G', X', k') is at most $\mathcal{O}(|X|^7)$ where $|X|$ is the size of the modulator in the original problem (G, X, k). Note that none of the reduction rules increase $|X|$, so any bound in terms of $|X'|$ is a better or equally good bound as a similar bound in $|X|$. The bound on the reduced instances we give here is done differently than how Jansen and Bodlaender [20] show their bound, though some elements are similar, such as the use of König's theorem on a perfect matching in a bipartite graph.

In order to prove this bound, we will first find a maximum matching M for F, and let the unmatched vertices constitute the set $\hat{X} := V(F) \setminus V(M)$. Clearly, $|V(G)| = |X| + |V(F)|$ and $|V(F)| = |V(M)| + |\hat{X}|$. The two main parts of our analysis will be (a) to show that $|\hat{X}| \leq \mathcal{O}(|X|^4)$, and then (b) to show that $|M| \leq |\hat{X}| \cdot \mathcal{O}(|X|^3)$.

Notice that Reduction 3.4 will remove connected components from F. When the reduction is not applicable, we should then be able to give some bound on the number of connected components in F. The next lemma gives such a bound:

Lemma 3.12. *Let (G, X, k) be a reduced instance of* IS/D2M. *Let $\mathcal{C}_F$ denote the set of all connected components $P \subseteq F$. Then $|\mathcal{C}_F| \leq |\mathcal{X}| \cdot |X| + |X|^3$, i. e. the number of connected components in F is at most $|\mathcal{X}| \cdot |X| + |X|^3$.*

Proof. Let $\mathcal{A}_F$ denote the set of all non-redundant anchor triangles in F. Consider the bipartite graph B between the chunks $\mathcal{X}$ and connected components $\mathcal{C}_F \setminus \mathcal{A}_F$ where there is an edge between $X' \in \mathcal{X}$ and $P \in \mathcal{C}_F \setminus \mathcal{A}_F$ if and only if $\textsc{Conf}_P(X') > 0$. Because Reduction 3.4 is not applicable, every connected component $P \in \mathcal{C}_F \setminus \mathcal{A}_F$ will have at least one edge incident to it in B. Thus any bound on the number of edges in B will also be a bound for $|\mathcal{C}_F \setminus \mathcal{A}_F|$.

For every chunk $X' \in \mathcal{X}$, we have by definition that $\textsc{Conf}_F(X') < |X|$. Since conflicts induced by a chunk X' on F in different connected components of F are distinct, each $X' \in \mathcal{X}$ is incident to less than $|X|$ connected components in B. We then get that $|\mathcal{C}_F \setminus \mathcal{A}_F| \leq |\mathcal{X}| \cdot |X|$. It remains to show that $|\mathcal{A}_F| \leq |X|^3$ to conclude the proof. This can be verified by observing that every anchor triangle has a neighborhood in X of size exactly 3. If two anchor triangles had the same neighborhood in G (and hence in X), they would be redundant, so there is at most one non-redundant anchor triangle for each distinct triple of X. Observe that the number of such distinct triples is less than $|X|^3$. □

Observation 3.13. Let (G, X, k) be a reduced instance of IS/D2M. Let M be a maximum matching in F, and let $\hat{X} := V(F) \setminus V(M)$ be the unmatched vertices. Then it holds that $|\hat{X}| \leq |\mathcal{X}| \cdot |X| + |X|^3$.

We see that the observation is true since F is a collection of paths and cycles. Then a maximum matching M of F will leave at most one vertex out of the matching for each connected component, and thus by Lemma 3.12 the observation follows immediately.

Next, we will do part (b) of our analysis, namely to give a bound for $|V(M)|$. In this endeavor we will make use of the following lemma:

Lemma 3.14. *Let (G, X, k) be a reduced instance of* IS/D2M *and let M be a maximum matching in F yielding a set of unmatched vertices $\hat{X} := V(F) \setminus V(M)$. Let $\hat{F} := F - \hat{X}$ be the graph induced by the maximum matching. Then for all chunks $X' \in \mathcal{X}$ it holds that* $\textsc{Conf}_{\hat{F}}(X') \leq |X| + |\hat{X}|$.

Proof. Consider an arbitrary chunk X'. By the definition of chunks we have

$$\textsc{Conf}_F(X') \leq |X| \tag{3.1}$$

By applying the definition of conflicts we then get

$$\alpha(F) \leq |X| + \alpha(F - N_G(X')) \tag{3.2}$$

Since removing vertices never will increase the independence number, we have

$$\alpha(F - \hat{X}) \leq \alpha(F) \tag{3.3}$$

Recall that $\hat{F} = F - \hat{X}$. Combining equations 3.2 and 3.3, we then get

$$\alpha(\hat{F}) \leq |X| + \alpha(F - N_G(X')) \tag{3.4}$$

By the definition of conflicts, we know that

$$\alpha(\hat{F}) = \textsc{Conf}_{\hat{F}}(X') + \alpha(\hat{F} - N_G(X')) \tag{3.5}$$

Combining equations 3.4 and 3.5, we get

$$\textsc{Conf}_{\hat{F}}(X') \leq |X| + \alpha(F - N_G(X')) - \alpha(\hat{F} - N_G(X')) \tag{3.6}$$

Since removing $|\hat{X}|$ vertices will reduce the independence number by at most $|\hat{X}|$, we observe

$$\alpha(F - N_G(X')) \leq \alpha(F - N_G(X') - \hat{X}) + |\hat{X}| \tag{3.7}$$

Recall that $\hat{F} = F - \hat{X}$. Combining equations 3.6 and 3.7, we get

$$\textsc{Conf}_{\hat{F}}(X') \leq |X| + |\hat{X}| \tag{3.8}$$

□

We will here state König's theorem, which we will refer to whilst proving the bound on M in the lemma that follows.

Proposition 3.15 (König's theorem [28, Theorem 16.2])**.** *For every bipartite graph G, the size of a minimum vertex cover equals the number of edges in a maximum matching.*

Lemma 3.16. *Let (G, X, k) be a reduced instance of* Independent Set/Degree-2 Modulator *and let M be a maximum matching in F yielding a set of unmatched vertices $\hat{X} := V(F) \setminus V(M)$. Then it holds that $|M| \leq (|\hat{X}| + |X|) \cdot |\mathcal{X}| + |\mathcal{C}_F|$.*

Proof. Let $M_T \subseteq M$ denote the edges which are a part of a triangle connected component in F. Now consider the bipartite graph B between $\mathcal{X}$ and $M \setminus M_T$ where there is an edge between $X' \in \mathcal{X}$ and the edge $uv \in M \setminus M_T$ if and only if $\textsc{Conf}_{\{u,v\}}(X') > 0$. Because Reduction 3.5 has been applied exhaustively, every matched pair in $M \setminus M_T$ has at least one edge incident to it in the bipartite graph. Thus, any bound on the number of edges in B will also be a bound for $|M \setminus M_T|$.

Observe that M is a perfect matching for $\hat{F}$, and that $\hat{F}$ is also bipartite (all odd cycles were broken when removing $\hat{X}$). Then by König's Theorem (Proposition 3.15), every minimum vertex cover of $\hat{F}$ contains exactly one endpoint from each edge in M. Furthermore, since M is perfect, any maximum independent set for $\hat{F}$ also contains exactly one endpoint from each edge in M. See that when a chunk $X' \in \mathcal{X}$ induces a conflict with some number p of pairs in M, then $\textsc{Conf}_{\hat{F}}(X') \geq p$, since each pair is independently contributing to a conflict induced on $\hat{F}$.

By Lemma 3.14 we have that $\textsc{Conf}_{\hat{F}}(X') \leq |X| + |\hat{X}|$ for every $X' \in \mathcal{X}$, which then limits the number of edges in B, and we get $|M \setminus M_T| \leq |\mathcal{X}| \cdot (|X| + |\hat{X}|)$. In order to complete the proof, it remains to show that $|M_T| \leq |\mathcal{C}_F|$. But this must be true, since there is at most one edge in M_T for each connected component. □

Theorem 3.17. Independent Set/Degree-2 Modulator *admits a kernel on $\mathcal{O}(|X|^7)$ vertices: There is a polynomial time algorithm that transforms an instance (G, X, k) to an equivalent instance (G', X', k') such that*

- $k' \leq k + |X|^3$
- $|X'| \leq |X|$
- $|V(G')| \leq 2|X|^7 + 6|X|^6 + 8|X|^5 + 11|X|^4 + 10|X|^3 + 5|X|^2 + |X|$

Proof. We will begin with the proof that $k' \leq k + |X|^3$. The only transformation which increase k is Reduction 3.3, which rise k by 1 each time it is applied. Note that none of the other reductions will ever remove or destroy a non-redundant anchor triangle in such a way that it will trigger Reduction 3.3 for the same triple twice. Thus the transformation will de done at most once for each triple in X, and k is increased by less than $|X|^3$.

Next, observe that no reduction rule ever increase X. The only reduction which change X is Reduction 3.1, which removes a vertex from X. Thus $X' \subseteq X$, and we have $|X'| \leq |X|$.

For the bound on $|V(G')|$, consider a maximum matching M of F' which yield a set of unmatched vertices $\hat{X} := F' - V(M)$. Let $\mathcal{X}'$ be the collection of chunks in the reduced instance, and $\mathcal{X}$ the collection of chunks in the original instance.

We start by observing that

$$|V(G')| = |X'| + |V(F')| = |X'| + 2|M| + |\hat{X}|$$

By Observation 3.13 we have

$$|\hat{X}| \leq |\mathcal{X}'| \cdot |X'| + |X'|^3 \leq |\mathcal{X}| \cdot |X| + |X|^3$$

By Lemmata 3.16 and 3.12 we have

$$|M'| \leq (|\hat{X}| + |X'|) \cdot |\mathcal{X}'| + |\mathcal{X}'| \cdot |X'| + |X'|^3 \leq |\hat{X}| \cdot |\mathcal{X}| + 2 \cdot |X| \cdot |\mathcal{X}| + |X|^3$$

Combining the above, we get

$$|M'| \leq |\mathcal{X}|^2 \cdot |X| + |X|^3 \cdot |\mathcal{X}| + 2 \cdot |X| \cdot |\mathcal{X}| + |X|^3$$

For the bound on $|V(G)|$ we then get

$$|V(G')| \leq |X| + 2|\mathcal{X}|^2 \cdot |X| + 2|X|^3 \cdot |\mathcal{X}| + 4|X| \cdot |\mathcal{X}| + 2|X|^3 + |\mathcal{X}| \cdot |X| + |X|^3$$

Observing that $|\mathcal{X}| \leq |X|^3 + |X|^2 + |X|$ and expanding the expression, we get the following bound

$$|V(G')| \leq 2|X|^7 + 6|X|^6 + 8|X|^5 + 11|X|^4 + 10|X|^3 + 5|X|^2 + |X|$$

Finally, observe that each reduction runs in polynomial time. □

Corollary 3.18. Vertex Cover/Degree-2 Modulator *admits a kernel on* $\mathcal{O}(|X|^7)$ *vertices: There is a polynomial time algorithm that transforms an instance* (G, X, k) *to an equivalent instance* (G', X', k') *such that* $|V(G')| \leq 2|X|^7 + 6|X|^6 + 8|X|^5 + 11|X|^4 + 10|X|^3 + 5|X|^2 + |X|$.

Proof. First, we transform the instance (G, X, k) of VC/D2M to an instance $(G, X, |V(G)| - k)$ of IS/D2M. Then we apply Theorem 3.17 to obtain $(G', X', |V(G')| - k')$, which we can transform back to an instance of VC/D2M (G', X', k'). Since the transformations between VC/D2M and IS/D2M preserve both the graph and the modulator, the bounds on the number of vertices in Theorem 3.17 also holds for VC/D2M. □

Chapter 4

Kernel for Vertex Cover Parameterized by Pseudoforest Modulator

This chapter gives a kernel for Vertex Cover on $\mathcal{O}(|X|^{12})$ vertices, where the parameter $|X|$ is the size of a minimum pseudoforest modulator of G. We define the problem as follows:

Vertex Cover/Pseudoforest Modulator (VC/PFM)
Input: A graph G, a pseudoforest modulator set $X \subseteq V(G)$ such that $G - X$ is a pseudoforest, and an integer k.
Parameter: Size of the pseudoforest modulator $|X|$.
Question: Does G contain a vertex cover of size at most k?

Note that we here state the problem with a pseudoforest modulator given as part of the input, but this condition can be omitted, seeing that a pseudoforest modulator can be approximated. For example, computing a modulator to a pseudoforest is a special case of the $\mathcal{F}$-Deletion problem, thus there exists a randomized constant factor approximation algorithm of running time $\mathcal{O}(nm)$ [14].

For ease of presentation, we will first develop a kernel for Independent Set parameterized by the size of a minimum pseudoforest modulator, and then by the immediate correspondence between the Vertex Cover and Independent Set problems the kernel for VC/PFM will follow. For the remainder of this section, we will thus focus on the following problem:

Independent Set/Pseudoforest Modulator (IS/PFM)
Input: A graph G, a pseudoforest modulator set $X \subseteq V(G)$ such that $G - X$ is a pseudoforest, and an integer k.
Parameter: Size of the pseudoforest modulator $|X|$.
Question: Does G contain an independent set of size at least k?

4.1 Tools

Throughout the chapter, let $F := G - X$ be the induced subgraph remaining after the modulator X has been removed from G. Note that F is a pseudoforest. We will for completeness restate unconventional definitions here, even if they have appeared elsewhere in the thesis.

Definition 4.1 (Conflicts [20, Definition 3])**.** Let (G, X, k) be an instance of IS/PFM where $F' \subseteq F$ is a subgraph of the pseudoforest F and $X' \subseteq X$ is a subset of the modulator X. Then the number of *conflicts* induced by X' on F' is defined as $\textsc{Conf}_{F'}(X') := \alpha(F') - \alpha(F' - N_G(X'))$.

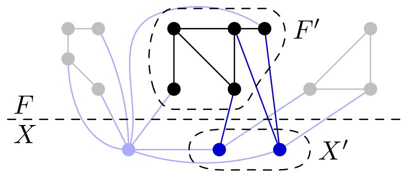


Figure 4.1: Conflicts: In the figure, we observe that $\alpha(F') = 3$, and $\alpha(F' - N_G(X')) = 1$. Hence, we get that $\textsc{Conf}_{F'}(X') = 2$. In other words, the number of conflicts induced by X' on F' is 2.

Choosing X' to be in the independent set I of G may prevent some vertices in F' from being included in same set I. In particular, no vertex $v \in V(F') \cap N_G(X')$ can be chosen to be in I. In light of this, the term $\textsc{Conf}_{F'}(X')$ can be understood as the price one has to pay in F' by choosing to include X' in the independent set. Observe that $\textsc{Conf}_{F'}(X')$ is polynomial time computable when the independence number is.

Next, we will introduce *chunks*. This concept originates in the paper by Jansen and Bodlaender [20, Definition 2], but a is altered for our purposes in two ways: The size of the chunks are here of size at most three (as opposed to two), and we have additionally introduced the requirement that a chunk cannot induce a conflict on F which is of size $|X|$ or larger.

Definition 4.2 (Chunks)**.** Let (G, X, k) be an instance of IS/PFM. A set $X' \subseteq X$ is a *chunk* if the following hold:

- X' is independent in G,
- the size of X' is between 1 and 3, i. e. $1 \leq |X'| \leq 3$, and
- the number of conflicts in F induced by X' is less than $|X|$, i. e. $\textsc{Conf}_F(X') < |X|$

We let $\mathcal{X}$ be the collection of all chunks of X.

The collection of chunks $\mathcal{X}$ can be seen as all suitable candidate subsets of size at most three from X to be included in a maximum independent set I of G. The idea is that I may contain a chunk as a subset, but it is never necessary to include a subset $X' \subseteq X$ of size at most three which is *not*

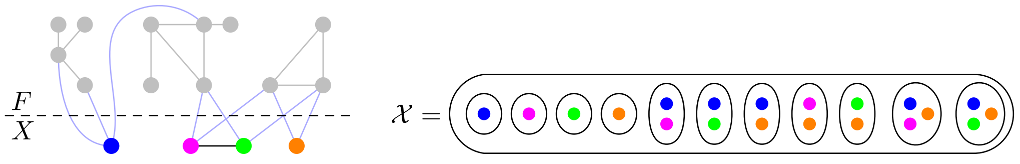


Figure 4.2: Chunks: Independent subsets of X with size ≤ 3 which induce a small conflict on F. There is no chunk which includes both the pink and the green vertex, since there is an edge between them in G.

a chunk. This will allow us to discard potential solutions containing non-chunk subsets of X with size at most three. In order for this intuition to hold, we provide the following lemma, originally by Jansen and Bodlaender though slightly altered to fit our purposes. For completeness, we also provide a proof.

Lemma 4.3 ([20, Lemma 2]). *Let (G, X, k) be an instance of* IS/PFM *where $\alpha(G) \geq k$. Then there exists an independent set I of G such that $|I| \geq k$ and for all subsets $X' \subseteq X \cap I$ it holds that* $\textsc{Conf}_F(X') < |X|$.

Proof. Assume that $I' \subseteq V(G)$ is an independent set with $|I'| \geq k$ for which there exists some $X' \subseteq I' \cap X$ such that $\textsc{Conf}_F(X') \geq |X|$. We will show that then there exists another independent set I such that $|I| \geq |I'|$ and for all subsets $X'' \subseteq X$ with $\textsc{Conf}_F(X'') \geq |X|$, it holds that X'' is not a subset of I.

Because $\textsc{Conf}_F(X') \geq |X|$, we have that $\alpha(F) \geq |X| + \alpha(F - N_G(X'))$. Now consider the independent set I'. Some of its vertices are in X, however no more than $|X|$. The remainder of vertices of I' are in $V(F)$, however no more than $\alpha(F - N_G(X'))$. Thus $|X| + \alpha(F - N_G(X')) \geq |I'|$. But then $\alpha(F) \geq |I'|$, and we see that a MIS of the pseudoforest F satisfies the requirements of the lemma. $\square$

Definition 4.4. (Anchor triangle) Let (G, X, k) be an instance of IS/PFM. Let P be a triangle connected component in F with $V(P) = \{p_1, p_2, p_3\}$. Then P is an *anchor triangle* if there exists three distinct vertices $x_1, x_2, x_3 \in X$ such that:

- $N_G(p_1) = \{p_2, p_3, x_1\}$
- $N_G(p_2) = \{p_1, p_3, x_2\}$
- $N_G(p_3) = \{p_1, p_2, x_3\}$

An anchor triangle is *redundant* if there is also another anchor triangle with the same open neighborhood in G, and it is *non-redundant* if there is not.

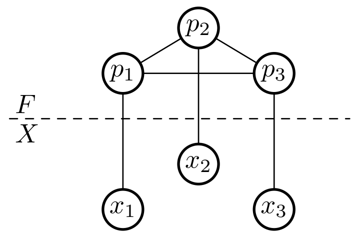


Figure 4.3: The connected component $P \subseteq F$ with vertices $V(P) = \{p_1, p_2, p_3\}$ is an anchor triangle for the triple $\{x_1, x_2, x_3\} \subseteq X$.

Definition 4.5. (Unnecessary triple) Let (G, X, k) be an instance of IS/PFM, and let $X_3 \subseteq X$ be a triple, i. e. a set containing three distinct vertices. Then X_3 is an *unnecessary triple* if there exists an anchor triangle P in F such that $N_G(P) = X_3$.

The fact that a triple $X_3 \subseteq X$ is unnecessary as defined above should intuitively be understood with respect to constructing an independent set. If a triple X_3 is unnecessary then there exists a MIS which does not contain all of X_3. This intuition is supported by the next lemma.

Lemma 4.6. *Let (G, X, k) be an instance of* IS/PFM. *If there exists an independent set of size at least k in G, then there exists an independent set I of G with $|I| \geq k$ containing no unnecessary triple $X_3 \subseteq X$.*

Proof. We will prove the lemma by constructing the required independent set I satisfying the properties, assuming we have an arbitrary independent set I' of size at least k. By definition, for each unnecessary triple $X_3 \subseteq X \cap I'$, there is at least one anchor triangle $P \subseteq F$ such that $N_G(P) = X_3$. Since $X_3 \subseteq I'$, we have that no vertex of P is in I'. Let $p_u \in V(P)$ be an arbitrary vertex of P, and let $u \in X_3$ be the neighbor of p_u in X_3. Observe that we can here replace u by p_u in I', and still have I' be an independent set of the same size. We do this for every unnecessary triple $X_3 \subseteq X \cap I'$ to obtain I, which then satisfies the requirement of the lemma. $\square$

4.2 Reduction Rules

We introduce here the reduction rules. Each reduction receives as input an instance (G, X, k) of Independent Set/Pseudoforest Modulator, and outputs an equivalent instance (G', X', k'). Reductions 4.1, 4.2 and 4.4 originates in [20], though Reduction 4.4 is altered to fit the context of a pseudoforest, which also required changes to the proof.

Reduction rules will be applied exhaustively starting with lower number rules, until Reduction 4.4 is no longer applicable. During this process, a lower number rule is always applied before a higher number rule if at any point they are both applicable. Then Reductions 4.5 and 4.6 will be applied once each to obtain the final reduced instance. Note that each reduction is computable in polynomial time.

Reduction 4.1 ([21, Reduction 1])**.** If there is a vertex $v \in X$ such that $\textsc{Conf}_F(\{v\}) \geq |X|$, then delete v from the graph G and from the set X. We let $G' := G - v$, $X' := X - v$ and $k' := k$.

Reduction 4.1 is safe due to Lemma 4.3.

Reduction 4.2 ([21, Reduction 2])**.** If there are distinct vertices $u, v \in X$ with $uv \notin E(G)$ for which $\textsc{Conf}_F(\{u, v\}) \geq |X|$, then add edge uv to G. We let $G' := (V(G), E(G) \cup \{uv\})$, $X' := X$ and $k' := k$.

Reduction 4.2 is safe, because there always exists a maximum independent set I in G leaving at least one of $\{u, v\}$ outside I by Lemma 4.3. Adding uv to $E(G)$ thus does not reduce the size of the maximum independent set. The backward direction is trivial, since removing an edge does not reduce the maximum size of an independent set.

Reduction 4.3. If there are distinct $u, v, w \in X$ such that $\textsc{Conf}_F(\{u, v, w\}) \geq |X|$, the set $\{u, v, w\}$ is independent in G, and for which there is no anchor triangle P with $N_G(P) = \{u, v, w\}$, then add an anchor triangle $P' = \{p_u, p_v, p_w\}$ to the graph such that $N_G(P') = \{u, v, w\}$, and increase k by one. Let $V(G') := V(G) \cup \{p_u, p_v, p_w\}$ and let $E(G') := E(G) \cup \{p_up_v, p_up_w, p_vp_w, p_uu, p_vv, p_ww\})$. Further, let $X' := X$ and let $k' := k + 1$.

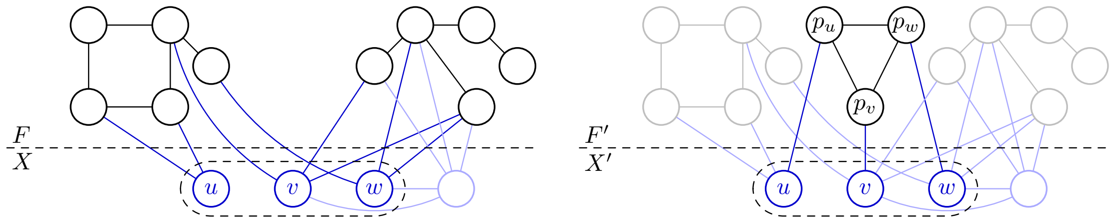


Figure 4.4: Reduction 4.3: Adding an anchor triangle to the independent triple $\{u, v, w\}$ ($k' = k+1$). This makes $\{u, v, w\}$ an unnecessary triple in the output instance.

Note that Reduction 4.3 makes the triple $\{u, v, w\} \subseteq X$ unnecessary in the output instance as defined in Definition 4.5.

Lemma 4.7. *Reduction 4.3 is safe. Let (G, X, k) be an instance of* IS/PFM *to which Reduction 4.3 is applicable, and let (G', X', k') be the output instance resulting from the reduction. Then (G, X, k) is a yes-instance if and only if (G', X', k') is a yes-instance.*

Proof. For the forward direction of the proof, assume (G, X, k) is a *yes*-instance, let $u, v, w \in X$ be the elements which triggers the reduction, and let $P' = \{p_u, p_v, p_w\}$ be the vertices of the added anchor triangle in the output instance. Let I be an independent set of G with $|I| \geq k$. By Lemma 4.3, we assume that at least one of u, v, w is not in I. Without loss of generality, let $u \notin I$. For the output instance, observe that $I' := I \cup \{p_u\}$ is an independent set of G' with size $|I'| = |I| + 1 \geq k + 1 = k'$, which makes (G', X', k') a *yes*-instance.

For the backward direction, assume that (G', X', k') is a *yes*-instance, and let P' be the added anchor triangle in the output instance. Let I' be an independent set of G' with $|I'| \geq k'$. Because P' induces a triangle in G', at most one vertex of P' is in I'. Thus, $I := I' \setminus P'$ is an independent set of G with $|I| \geq |I'| - 1 \geq k' - 1 = k$, which makes (G, X, k) a *yes*-instance. □

The next reduction rule originates in the paper by Jansen and Bodlaender [20, Reduction 3], but it has been altered to fit our purposes. Because chunks are defined differently, and because F now contains cycles, the proof of safeness of this reduction is in some aspects very different.

Reduction 4.4. If there exists a connected component P in F which is not a non-redundant anchor triangle, and for every chunk $X' \in \mathcal{X}$ there is no conflict induced by X' on P, i. e. $\textsc{Conf}_P(X') = 0$, then remove P from G and reduce k by $\alpha(P)$. We let $G' := G - P$, $X' := X$ and $k' := k - \alpha(P)$.

To prove that Reduction 4.4 is safe, we will rely on the following lemma, which states that any pseudotree has a small (at most size three) obstruction in terms of obtaining a maximum independent set.

Lemma 4.8. *Let P be a pseudotree and let Z be a set of vertices such that $\alpha(P) > \alpha(P - Z)$. Then there exist three (possibly non-distinct) vertices $u, v, w \in Z \cap V(P)$ such that $\alpha(P) > \alpha(P - \{u, v, w\})$.*

The proof of the above lemma is quite technical, and is postponed till Section 4.4 in order to preserve the flow of the kernelization algorithm. Taking Lemma 4.8 as a black box, we are able to make the following observation:

Observation 4.9. Let $P \subseteq F$ be a connected component in the pseudoforest F and let $X' \subseteq X$ be an independent set such that $\textsc{Conf}_P(X') > 0$. Then there exists some $X'' \subseteq X'$ with $1 \leq |X''| \leq 3$ such that $\textsc{Conf}_P(X'') > 0$.

Proof. We see that the observation is true, since by Lemma 4.8 there exist $u, v, w \in N_G(X') \cap V(P)$ such that $\alpha(P) > \alpha(P - \{u, v, w\})$. Then for each element u, v, w, pick an arbitrary neighbor $x_u, x_v, x_w \in X'$ (they need not be distinct) to form the set $X'' := \{x_u, x_v, x_w\}$. See that then $\textsc{Conf}_P(X'') > 0$. □

We are now equipped to prove safeness of Reduction 4.4.

Lemma 4.10. *Reduction 4.4 is safe. Let (G, X, k) be an instance of* IS/PFM *to which Reduction 4.4 is applicable, and let (G', X', k') be the output instance resulting from the reduction. Then (G, X, k) is a yes-instance if and only if (G', X', k') is a yes-instance.*

Proof. Let $P \subseteq F$ be the connected component which triggered the reduction.

For the forward direction of the proof, assume (G, X, k) is a *yes*-instance and let I be an independent set of G with size at least k. Let $I' := I \setminus V(P)$. Clearly I' is an independent set of G'. Now observe that $|I \cap V(P)| \leq \alpha(P)$, and thus $|I'| = |I| - |I \cap V(P)| \geq k - \alpha(P) = k'$. Hence (G', X', k') is a *yes*-instance.

For the backward direction, we assume that (G', X', k') is a *yes*-instance, and has an independent set I' of size at least k'. Because of Lemma 4.6 we can assume that I' contains no unnecessary triples. We want to show that we can always pick some independent set $I_P \subseteq V(P)$ with $|I_P| = \alpha(P)$ such that $I := I' \cup I_P$ is an independent set with size at least $k' + \alpha(P) = k$. Since I' and $V(P)$ are disjoint in G by construction, it will suffice to show that $\alpha(P - N_G(I')) \geq \alpha(P)$.

Assume for the sake of contradiction that $\alpha(P - N_G(I')) < \alpha(P)$. Since P was a connected component in F, all its neighbors $N_G(P)$ are in X. Thus we have that $\textsc{Conf}_P(X' \cap I') > 0$. By

Observation 4.9, we further have that there exists some $X'' \subseteq X' \cap I'$ such that $1 \leq |X''| \leq 3$ and $\textsc{Conf}_P(X'') > 0$.

For any such X'', there are two cases. In the first case, $\textsc{Conf}_F(X'') < |X|$. Because X'' is also independent and has size at most 3, it is a chunk of X in the input instance. This contradicts the preconditions for Reduction 4.4, so this case can not happen.

In the second case, $\textsc{Conf}_F(X'') \geq |X|$. But then one of Reductions 4.1, 4.2 or 4.3 would have previously been applied to X'', yielding it either unfeasible for an independent set or making it an unnecessary triple in the input instance. Because non-redundant anchor triangles are not chosen for removal by Reduction 4.4, X'' is also an unnecessary triple in the output instance, which contradicts that I' contains no unnecessary triples. This concludes the proof. $\square$

Notice that Reduction 4.4 will remove connected components from F. When the reduction is not applicable, we should then be able to give some bound on the number of connected components in F. The next lemma gives such a bound:

Lemma 4.11. *Let (G, X, k) be an instance of* IS/PFM *which is irreducible with respect to Reductions 4.1, 4.2, 4.3 and 4.4. Let $\mathcal{C}_F$ denote the set of all connected components $P \subseteq F$. Then $|\mathcal{C}_F| \leq |\mathcal{X}| \cdot |X| + |X|^3$, i. e. the number of connected components in F is at most $|\mathcal{X}| \cdot |X| + |X|^3$.*

Proof. Let $\mathcal{A}_F$ denote the set of all non-redundant anchor triangles in F. Consider the bipartite graph B between the chunks $\mathcal{X}$ and connected components $\mathcal{C}_F$ where there is an edge between $X' \in \mathcal{X}$ and $P \in \mathcal{C}_F$ if and only if $\textsc{Conf}_P(X') > 0$. Because Reduction 4.4 is not applicable, every connected component $P \in \mathcal{C}_F$ will have at least one edge incident to it in B, unless the component is a non-redundant anchor triangle. Thus any bound on the number of edges in B will also be a bound for $|\mathcal{C}_F \setminus \mathcal{A}_F|$.

For every chunk $X' \in \mathcal{X}$, we have by definition that $\textsc{Conf}_F(X') < |X|$. Since conflicts induced by a chunk X' on F in different connected components of F are distinct, each chunk $X' \in \mathcal{X}$ is incident to less than $|X|$ connected components in B. We then get that $|\mathcal{C}_F \setminus \mathcal{A}_F| \leq |\mathcal{X}| \cdot |X|$. It remains to show that $|\mathcal{A}_F| \leq |X|^3$ to conclude the proof. This can be verified by observing that every anchor triangle has a neighborhood in X of size exactly 3. If two anchor triangles had the same neighborhood in G (and hence in X), they would be redundant, so there is at most one non-redundant anchor triangle for each distinct triple of X. The number of such distinct triples is less than $|X|^3$. $\square$

When the above reduction rules have been exhaustively applied, the next two reductions will be executed exactly once each.

Reduction 4.5. Let $\hat{X} \subseteq V(F)$ be a set such that $\hat{X}$ contains exactly one vertex of each cycle in F. For the output instance, let $G' := G$, $X' := X \cup \hat{X}$, and $k' := k$.

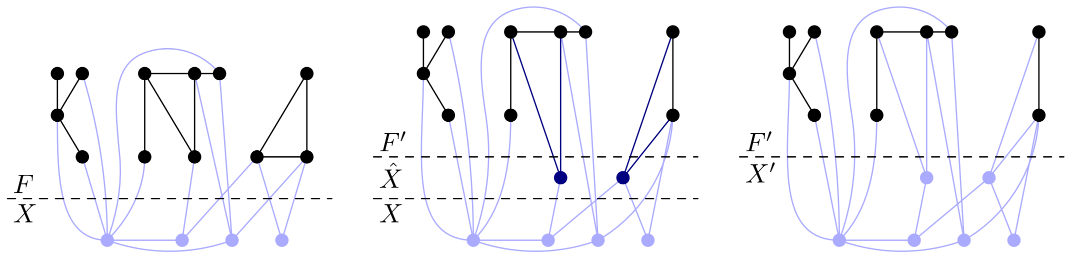


Figure 4.5: Reduction 4.5: One vertex from each cycle in F is moved to X.

The reduction is safe because neither G nor k was changed. Observe that X is now a feedback vertex set (which is fine, since every feedback vertex set is also a modulator to pseudoforest). This reduction may increase the size of X dramatically. This is why the Reduction is applied only once, such that we can give guarantees for the size of the reduced instance.

Observation 4.12. Let (G', X', k') be an instance of IS/PFM after Reduction 4.5 have been applied to (G, X, k). Then $X' \leq |X|^4 + 2|X|^3 + |X|^2 + |X|$.

Proof. Since the input instance (G, X, k) was irreducible with respect to Reductions 4.1, 4.2, 4.3 and 4.4, we have by Lemma 4.11 that the number of connected components in F is at most $|\mathcal{X}| \cdot |X| + |X|^3$, where $\mathcal{X}$ is the chunks in the input instance. Since $|\mathcal{X}| < |X|^3 + |X|^2 + |X|$, then the number of connected components is less than $|X|^4 + 2|X|^3 + |X|^2$. Further, because F is a pseudoforest, there is at most one vertex from each connected component in $\hat{X}$. We thus have that $|\hat{X}| \leq |X|^4 + 2|X|^3 + |X|^2$, which in turn yields $|X \cup \hat{X}| \leq |X| + |X|^4 + 2|X|^3 + |X|^2$. □

After Reduction 4.5 has been applied once, the returned instance (G, X, k) is ready for the final reduction step. Note that since X is now a feedback vertex set, (G, X, k) is now an instance of Independent Set/Feedback Vertex Set as well.

Independent Set/Feedback Vertex Set (IS/FVS)
Input: A simple undirected graph G, a feedback vertex set $X \subseteq V(G)$, and an integer k.
Parameter: Size of the feedback vertex set $|X|$.
Question: Does G contain an independence set of size at least k?

The next step relies on the following result of Jansen and Bodlaender [20, Theorem 2]:

Proposition 4.13. Independent Set/Feedback Vertex Set *has a kernel with a cubic number of vertices: There is a polynomial-time algorithm that transforms an instance* (G, X, k) *into an equivalent instance* (G', X', k') *such that* $|X'| \leq 2|X|$, *and* $|V(G')| \leq 56|X|^3 + 28|X|^2 + 2|X|$.

Reduction 4.6. Let the output instance (G', X', k') be the kernel after applying Proposition 4.13 to (G, X, k). This reduction is applied once only.

4.3 Bound on Size of Reduced Instances

In this section we will prove that the number of vertices in a reduced instance (G', X', k') is at most $\mathcal{O}(|X|^{12})$ where $|X|$ is the size of the modulator in the original problem (G, X, k).

Theorem 4.14. Independent Set/Pseudoforest Modulator *has a kernel with $\mathcal{O}(|X|^{12})$ vertices: There is a polynomial time algorithm that transforms an instance (G, X, k) to an equivalent instance (G', X', k') such that*

- $|V(G')| \leq 56(|X|^4 + 2|X|^3 + |X|^2 + |X|)^3 + 28(|X|^4 + 2|X|^3 + |X|^2 + |X|)^2 + 2(|X|^4 + 2|X|^3 + |X|^2 + |X|)$,
- $|X'| \leq 2|X|^4 + 4|X|^3 + 2|X|^2 + 2|X|$, *and*
- $k' \leq k + |X|^3$.

Proof. We will begin with the proof that $k' \leq k + |X|^3$. The only transformation which increase k is Reduction 4.3, which rise k by 1 each time it is applied. However, this transformation will be done less than $|X|^3$ times, since the rule will be applied at most once for each distinct triple of X.

Next, we focus on the bound $|X'| \leq 2|X|^4 + 4|X|^3 + 2|X|^2 + 2|X|$. Let (G'', X'', k'') be the instance of IS/PFM after Reduction 4.5 has been applied. Observe that none of Reductions 4.1, 4.2, 4.3 or 4.4 increase X. Then by Observation 4.12 we have that $|X''| \leq |X|^4 + 2|X|^3 + |X|^2 + |X|$. By Proposition 4.13 we have that the size is at most doubled after applying Reduction 4.6. Thus the bound holds.

For the bound on $V(G)$, let us consider the instance of IS/FVS (G'', X'', k'') to which Reduction 4.6 was applied in order to obtain the final reduced instance (G', X', k'). We have already established that $|X''| \leq |X|^4 + 2|X|^3 + |X|^2 + |X|$. It follows from Proposition 4.13 that in the final reduced instance, $|V(G')| \leq 56|X''|^3 + 28|X''|^2 + 2|X''|$, which in terms of $|X|$ yields $|V(G')| \leq 56(|X|^4+2|X|^3+|X|^2+|X|)^3+28(|X|^4+2|X|^3+|X|^2+|X|)^2+2(|X|^4+2|X|^3+|X|^2+|X|)$.

Finally, observe that each reduction can be done in polynomial time. □

Corollary 4.15. Vertex Cover/Pseudoforest Modulator *has kernel with $\mathcal{O}(|X|^{12})$ vertices: There is a polynomial time algorithm that transforms an instance (G, X, k) to an equivalent instance (G', X', k') such that $|V(G')| \leq 56(|X|^4 + 2|X|^3 + |X|^2 + |X|)^3 + 28(|X|^4 + 2|X|^3 + |X|^2 + |X|)^2 + 2(|X|^4 + 2|X|^3 + |X|^2 + |X|)$.*

Proof. First, we transform the instance (G, X, k) of VC/PFM to an instance $(G, X, |V(G)| - k)$ of IS/PFM. Then we apply Theorem 4.14 to obtain $(G', X', |V(G')| - k')$, which we can transform back to an instance of VC/PFM (G', X', k'). Since the transformations between VC/PFM and IS/PFM preserve both the graph and the modulator, the bounds on the number of vertices in Theorem 4.14 also holds for VC/PFM. □

4.4 Proof of Lemma 4.8

In this section we prove Lemma 4.8. Our starting point will be the following result by Jansen and Bodlaender [19, Lemma 4], rephrased here in terms of a vertex set Z:

Proposition 4.16. *Let T be a tree, and let Z be a set of vertices. If $\alpha(T) > \alpha(T - Z)$, then there exist two (possibly non-distinct) vertices $u, v \in Z \cap V(T)$ such that $\alpha(T) > \alpha(T - \{u, v\})$.*

We also need to establish a framework for reasoning about independent sets in trees which rely on whether a vertex is α-critical or not. We will use the following definition:

Definition 4.17 (α-critical)**.** Let G be a graph, and let v be a vertex. If v is in every maximum independent set of G, i. e. $\alpha(G) = 1 + \alpha(G - v)$, then v is *α-critical* in G.

Observation 4.18. Let G be a graph and let v be a vertex in G. Then v is an α-critical vertex of G if and only if $\alpha(G - v) = \alpha(G - N[v])$.

Lemma 4.19. *Let T be a tree rooted at r. Then r is α-critical in T if and only if a is not α-critical in T_a for all children a of r.*

Proof. For the forward direction of the proof, assume r is α-critical in T. Since T is a tree rooted at r, we have that $\alpha(T - r) = \sum_{a \in C(r)} \alpha(T_a)$ and $\alpha(T - N[r]) = \sum_{a \in C(r)} \alpha(T_a - a)$.

Assume for the sake of contradiction that there is some $a' \in C(r)$ such that a' is α-critical in $T_{a'}$. Then we have $\alpha(T_{a'}) = 1 + \alpha(T_{a'} - a')$, which lead us to conclude that $\sum_{a \in C(r)} \alpha(T_a) \geq 1 + \sum_{a \in C(r)} \alpha(T_a - a)$. This contradicts that r is α-critical in T by Observation 4.18.

For the backward direction of the proof, assume that for all children a of r, a is not α-critical for T_a. Seeing that T is a tree, we recall that $\alpha(T - r) = \sum_{a \in N(r)} \alpha(T_a)$. Adding the single vertex r back will increase the independence number by at most one. It remains to show that it also increase by at least one to conclude the proof. But this can be done by picking a maximum independent set in $T - r$ which avoids all of $C(r)$, and then include r. Note that such a set exists by the initial assumption that no child a of r is α-critical for T_a. ☐

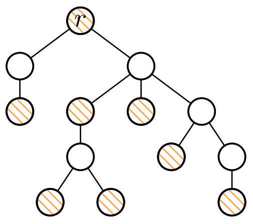


Figure 4.6: Lemma 4.19: Let T be a tree rooted at r. A vertex $a \in V(T)$ is marked with orange stripes if a it is α-critical in the subtree T_a. Then a is α-critical if and only if, for all its children $b \in C(a)$, b is not α-critical in T_b.

Observation 4.20. Let G be a graph and let a be an α-critical vertex of G. If there is a set of vertices Z such that a is not α-critical in $G - Z$, then $\alpha(G) > \alpha(G - Z)$.

Observation 4.21. Let T be a tree rooted at r, and let Z be a set of vertices. Then the following holds:

- If r is α-critical in $T - Z$, then $\alpha(T - Z) = 1 + \sum_{a \in C(r)} \alpha(T_a - Z)$. In particular, if r is α-critical in T then $\alpha(T) = 1 + \sum_{a \in C(r)} \alpha(T_a)$.
- If r is not α-critical in $T - Z$, then $\alpha(T - Z) = \sum_{a \in C(r)} \alpha(T_a - Z)$. In particular, if r is not α-critical in T then $\alpha(T) = \sum_{a \in C(r)} \alpha(T_a)$.

Armed with Proposition 4.16 and the framework for reasoning about independent sets in trees presented above, we are now prepared to show the next lemma:

Lemma 4.22. *Let T be a tree rooted at r, and let Z be a set of vertices. Then the following holds:*

(a) If $\alpha(T) > \alpha(T - Z)$ and r is α-critical in T, then either

(i) There exist two (possibly non-distinct) vertices $u, v \in Z \cap V(T)$ such that $\alpha(T) > \alpha(T - \{u, v\})$ and r is still α-critical in $T - \{u, v\}$.

(ii) There exists a vertex $u \in Z \cap V(T)$ such that $\alpha(T) > \alpha(T - u)$ and r is not α-critical in $T - u$.

(b) If $\alpha(T) = \alpha(T - Z)$ and r is α-critical in $T - Z$ but not in T, then there exists a vertex $u \in Z \cap V(T)$ such that r is α-critical in $T - u$.

Proof. The proof is done by strong induction on the height of the tree, where we will go two levels deep. For the base case we consider trees of height 0 and 1, i. e. the single vertex and stars rooted at the center vertex. In both cases, it is easy to verify that the lemma holds. For the inductive step, we let T be a tree rooted in r. By the induction hypothesis we assume the lemma holds for all strict subtrees of T, as these have all strictly smaller height than T. We will now show each part of the lemma separately.

Part (a). We assume that $\alpha(T) > \alpha(T - Z)$, and that r is α-critical in T. First consider the case when $r \in Z$. Then just pick $u := r$ to meet the requirement for (ii). Thus, for the remainder of the case we can assume $r \notin Z$. By Lemma 4.19, we can also assume that there is no child a of r such that a is α-critical in T_a. We will now consider two cases.

Case 1. This case applies if there is no child a of r such that $\alpha(T_a) > \alpha(T_a - Z)$. Then there must exist at least one child a' of r such that a' is α-critical in $T_{a'} - Z$, or else there is a contradiction with the assumption that $\alpha(T) > \alpha(T - Z)$. Hence we can apply the induction hypothesis (b) to $T_{a'}$ and find that there exists a vertex $u \in Z \cap V(T_{a'})$ such that a' is α-critical in $T_{a'} - u$. Observe that by Lemma 4.19, r is not α-critical in $T-u$. Further applying Observation 4.21 to find that $\alpha(T) > \alpha(T-u)$, we see that we have met the requirement of (ii).

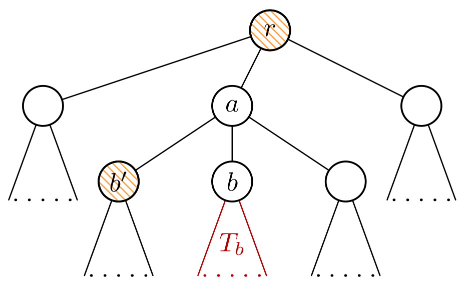
Case 2. This case applies if there exists a child a of r such that $\alpha(T_a) > \alpha(T_a - Z)$. Before we proceed further, we want to establish that there must exist some child b' of a such that $\alpha(T_{b'}) > \alpha(T_{b'} - Z)$. For the sake of contradiction, assume there is not, i. e. $\sum_{b\in C(a)} \alpha(T_b) = \sum_{b\in C(a)} \alpha(T_b - Z)$. Because a is not α-critical in T_a by Lemma 4.19, we have that $\alpha(T_a) = \sum_{b\in C(a)} \alpha(T_b)$ by Observation 4.21. Now note that $\sum_{b\in C(a)} \alpha(T_b - Z) \leq \alpha(T_a - Z)$. Combining the above, we obtain $\alpha(T_a) \leq \alpha(T_a - Z)$, which contradicts the initial assumption of this case. Thus for the remainder of the case we can assume there is at least one child b' of a such that $\alpha(T_{b'}) > \alpha(T_{b'} - Z)$. By Lemma 4.19, we also have that there exists at least one child b'' of a which is α-critical in $T_{b''}$. Now we will again distinguish between two different cases:

Subcase 2.1. This subcase applies if there exist two distinct children b and b' of a such that $\alpha(T_b) > \alpha(T_b - Z)$ and b' is α-critical in $T_{b'}$ (see Figure 4.7). By Lemma 4.16, there exist $u, v \in Z \cap V(T_b)$ such that $\alpha(T_b) > \alpha(T_b - \{u, v\})$. Note that a is not α-critical in $T_a - \{u, v\}$ by Lemma 4.19 because b' is α-critical in $T_{b'} - \{u, v\}$ ($T_{b'}$ is untouched by u, v). In order to meet the requirements of (i), we will proceed to show that $\alpha(T_a) > \alpha(T_a - \{u, v\})$.

To show this, we use Observation 4.21 to find that $\alpha(T_a) = \sum_{b''\in C(a)} \alpha(T_{b''})$ and $\alpha(T_a - \{u, v\}) = \sum_{b''\in C(a)} \alpha(T_{b''} - \{u, v\})$. Because there exists a child b of a such that $\alpha(T_b) > \alpha(T_b - \{u, v\})$, it then follows that $\alpha(T_a) > \alpha(T_a - \{u, v\})$. Because a is not α-critical in $T_a - \{u, v\}$ and the siblings of a are untouched by u, v, it follows that r is α-critical by Lemma 4.19. By symmetric reasoning as above but using T and r rather than T_a and a, we conclude that also $\alpha(T) > \alpha(T - \{u, v\})$, thus obtaining the requirements for (i).

Subcase 2.2. This subcase applies if there is exactly one child b of a which is α-critical in T_b, and this is the only child of a for which $\alpha(T_b) > \alpha(T_b - Z)$ (see Figure 4.7). The vertex a may have other children besides b, but for all of these children $b' \in C(a) \setminus \{b\}$, we have that b' is not α-critical in $T_{b'}$ and $\alpha(T_{b'}) = \alpha(T_{b'} - Z)$. By the induction hypothesis (a) there are two cases:

- There exist two (possibly non-distinct) vertices $u, v \in Z \cap V(T_b)$ such that $\alpha(T_b) > \alpha(T_b - \{u, v\})$ and b is α-critical in $T_b - \{u, v\}$. This case leaves a not α-critical in $T_a - \{u, v\}$, and yields $\alpha(T_a) > \alpha(T_a - \{u, v\})$. By an analogous argument to that of the previous subcase, we obtain the requirements for (i).
- There exists $u \in Z \cap V(T_b)$ such that $\alpha(T_b) > \alpha(T_b - u)$, and such that b is not α-critical in $T_b - u$. In this case, a is α-critical in $T_a - u$. This, however, in turn yields r not α-critical in $T - u$, which must reduce the independence number of T by Observation 4.20. Thus, we have obtained the requirement for (ii).



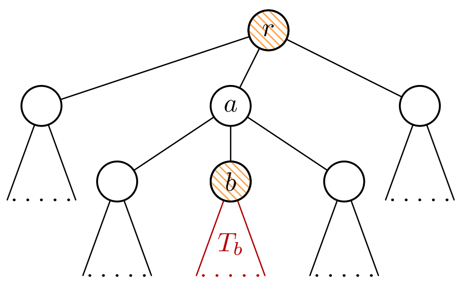


Figure 4.7: Proof of Lemma 4.22 (a), Case 2: Orange stripes indicates that the vertex is α-critical in the subtree rooted at that vertex. Subtrees where $\alpha(T_a) > \alpha(T_a - Z)$ are drawn with a red color.

We have now exhausted all possibilities, and in each case obtained the requirements for either (i) or (ii). This concludes the proof for part (a).

Part (b). We assume that $\alpha(T) = \alpha(T - Z)$ and that r is α-critical in $T - Z$, but not in T. By Lemma 4.19, we know that r is α-critical in $T - Z$ if and only if no child a of r is α-critical in $T_a - Z$. If there are two or more children which are α-critical in their respective subtrees before forbidding Z, we note that $\alpha(T) > \alpha(T - Z)$ by Observations 4.20 and 4.21, contradicting the premise of part (b) in the lemma. Thus there is exactly one child a of r which is α-critical in T_a, and for all other children a' of r it holds that a' is not α-critical in neither $T_{a'}$ nor $T_{a'} - Z$, as this would contradict either that $\alpha(T) = \alpha(T - Z)$ or that r is α-critical in $T - Z$.

By Observation 4.20, we know that $\alpha(T_a) > \alpha(T_a - Z)$. By the induction hypothesis (a), there are two possibilities; either (i) is true for T_a, and there exist vertices $u, v \in Z$ such that $\alpha(T_a) > \alpha(T_a - \{u, v\})$ and so that a is α-critical in $T_a - \{u, v\}$. That, however, would contradict the premise that $\alpha(T) = \alpha(T - Z)$. We may then assume that only (ii) is true, and that there exists a vertex $u \in Z \cap V(T_a)$ such that $\alpha(T_a) > \alpha(T_a - u)$ and which leave a not α-critical in $T_a - u$. See that this choice of u will also leave r α-critical in $T - u$, concluding the proof for part (b). □

We will now move on to pseudotrees, i. e. graphs which contains at most one cycle, and study how maximum independent sets behave in them when a set Z of vertices forbidden to be included in any independent set is introduced. We can view a pseudotree P as a cycle C with rooted trees being attached to vertices of C (if there is no cycle, P is a tree and behaves accordingly). A vertex in C may have zero, one or several trees being attached to it. Every non-cycle vertex with a neighbor in C is the root r of a tree T attached to the cycle. Formally, we use this definition:

Definition 4.23 (Attached tree)**.** Let P be a pseudotree which contains a cycle $C \subseteq P$. Each connected component T of $P - C$ is then an *attached tree* of C. Further, let $c \in V(C)$ and $r \in V(T)$ be the two vertices such that $cr \in E(P)$ (there is a unique pair with this property since P is a pseudotree). Then r is designated as the root of T, and c is the *attachment point* vertex for T. We say that c has an attached tree T, and that T is *attached* to C at c.

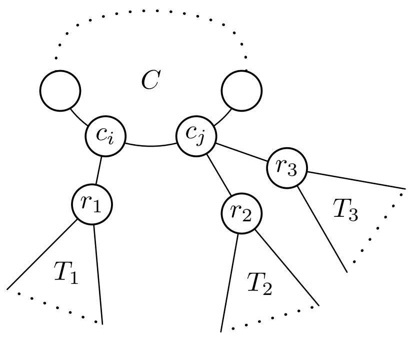


Figure 4.8: Attached tree: A pseudotree consists of a cycle C and attached subtrees. We say that the attached tree T_1 is rooted at r_1, and is attached to C at its attachment point vertex c_i. The cycle vertex c_j has two attached trees, T_2 and T_3.

In building a maximum independent set I for a pseudotree P, each attached tree is faced with two possibilities; either their attachment point vertex $c \in C$ is in I, or it is not. In some cases, c being in I may reduce the number of vertices available to pick in a tree T attached to c, since the root r of T is forbidden in I due to it being a neighbor of c. In cases like this, we observe that it is always at least as good to let c be left outside I and rather pick the larger independent set from T. This motivates distinguishing between trees where the root is α-critical in the tree from those where it is not.

With this in mind, we observe that the following greedy strategy for finding a MIS in a pseudotree P is correct: First, pick a MIS in every tree T attached to C which avoids the root r of T unless r is α-critical in T. Next, find a MIS for the cycle C which avoids any $c \in C$ which has an attached tree T where the root r is α-critical in T. The union of found independent sets will be a MIS for P. We can easily extend this algorithm to also avoid a set Z of vertices:

Algorithm 4.24 (Greedy MIS for pseudotree avoiding Z)**.** Input: A pseudotree P and a set of vertices Z. Output: A maximum independent set I of $P - Z$.

1. If P contain no cycle C, then return the MIS of P avoiding Z.
2. Let $Z' := \emptyset$. This set is for cycle vertices which will be marked as forbidden due to α-criticalness of roots in attached subtrees.
3. For every tree $T_0, T_1, \cdots, T_t$ attached to the cycle C, let r_i be the root of T_i and let $c_i \in C$ be the attachment point vertex for T_i. Then for $i \in \{1, 2, \cdots, t\}$, do the following:
   (a) Let I_{T_i} be a MIS for $T_i - Z$ which avoids r_i unless r_i is α-critical in $T_i - Z$.
   (b) If r_i is α-critical in $T_i - Z$, mark c_i as forbidden: Let $Z' := Z' \cup \{c_i\}$.
4. Let I_C be a MIS for the cycle vertices $C - (Z \cup Z')$.
5. Let the final solution be the union of the found sets: Return $(\bigcup_{i=1}^{t} I_{T_i}) \cup I_C$.

When Algorithm 4.24 is called on input (P, Z), we refer to sets of this solution with a superscript $^{(P,Z)}$, e. g. the set $I_{T_i}^{(P,Z)}$ refers to the set I_{T_i} when the algorithm is called on input (P, Z). However, note that we will never actually call this algorithm during the kernelization of IS/PFM, it is used only for analysis. Seeing that the above algorithm is correct, we are now ready to prove the final lemma of this section:

Lemma 4.8. *Let P be a pseudotree and let Z be a set of vertices such that $\alpha(P) > \alpha(P - Z)$. Then there exist three (possibly non-distinct) vertices $u, v, w \in Z \cap V(P)$ such that $\alpha(P) > \alpha(P - \{u, v, w\})$.*

Proof. Assume the condition of the lemma holds. If P has no cycle, then by Lemma 4.16 there exist $u, v \in Z \cap V(P)$ such that $\alpha(P) > \alpha(P - \{u, v\})$, and we have obtained the requirement of the lemma (we simply let one of the elements repeat, e.g. let $w := v$). For the remainder of the proof, we may therefore assume that P has a cycle C.

If there is a vertex w of the cycle C in P which is also in Z, then pick it as one of the three elements. Unless $\alpha(P) > \alpha(P - w)$, we have by Lemma 4.16 that there are two vertices $u, v \in Z \cap V(P)$ such that $\alpha(P - w) > \alpha(P - \{u, v, w\})$. In either case we have obtained the requirement of the lemma. For the remainder of the proof we may therefore assume that there is no vertex of the cycle $C \subseteq P$ which is also in Z.

Let us next consider the trees attached to C. If there exists a tree T rooted at r which is of one of the following types, then we can find $u, v, w \in Z$ which satisfy the requirement of the lemma:

- *r is not α-critical in T and $\alpha(T) > \alpha(T - Z)$.* Then by Lemma 4.16, there exist $u, v \in Z \cap V(T)$ such that $\alpha(T) > \alpha(T - \{u, v\})$. Observe that Algorithm 4.24 will produce sets $|I_T^{(P,\emptyset)}| > |I_T^{(P,\{u,v\})}|$ and $|I_C^{(P,\emptyset)}| \geq |I_C^{(P,\{u,v\})}|$, so by the correctness of the algorithm we conclude that $\alpha(P) > \alpha(P - \{u, v\})$. The requirement of the lemma is thus satisfied.
- *r is α-critical in T, $\alpha(T) > \alpha(T - Z)$, and case (i) of Lemma 4.22 (a) holds for T, r, Z.* Then there exist $u, v \in Z \cap V(T)$ such that $\alpha(T) > \alpha(T - \{u, v\})$ and r is α-critical in $T - \{u, v\}$. In order to show that $\alpha(P) > \alpha(P - \{u, v\})$, consider what Algorithm 4.24 will do: Whether called on $(P, \emptyset)$ or $(P, \{u, v\})$, the attachment point vertex $c \in C$ for T will still be marked as forbidden. Thus the only difference occurs in I_T, which is strictly smaller in the latter case. Hence, the requirement of the lemma is satisfied.
- *r is α-critical in T, and $\alpha(T) = \alpha(T - Z)$.* Let c be the attachment point vertex for T. Observe that $\alpha((P - T) - c) > \alpha((P - T) - (\{c\} \cup Z))$, or else there is a contradiction with the preconditions of the lemma. Since $(P - T) - c$ is a tree, we then have by Lemma 4.16 that there exist $u, v \in Z \cap V((P - T) - c)$ such that $\alpha((P - T) - c) > \alpha((P - T) - \{c, u, v\})$. Observe that also $\alpha(P) > \alpha(P - \{u, v\})$, and the lemma is satisfied.

From now on, we can assume that there are no trees attached to C of the above types. We observe that every tree T which is attached to C with root r must then be one of these three types instead:

(a) r is α-critical in T, and $\alpha(T) > \alpha(T - Z)$. Case (i) of Lemma 4.22 (a) does not apply, so by case (ii), there exists a singleton $u \in Z \cap V(T)$ such that $\alpha(T) > \alpha(T - u)$ and r is not α-critical in $T - u$.

(b) r is α-critical in $T - Z$, but not in T. We know that $\alpha(T) = \alpha(T - Z)$, and by Lemma 4.22 (b) that there exists a singleton $u \in Z \cap V(T)$ such that r is α-critical in $T - u$.

(c) r is α-critical in neither T nor $T - Z$. We know that $\alpha(T) = \alpha(T - Z)$.

If there is a vertex of the cycle $c \in C$ such that two trees T_a and T'_a both of type (a) are attached to c with respective roots r and r', then we have that there exists $u \in Z \cap V(T_a)$ such that $\alpha(T_a) > \alpha(T_a - u)$. Note that u is sufficient to yield $\alpha(P) > \alpha(P - u)$. To see this, consider what happens in Algorithm 4.24: Regardless of whether it was called with parameters $(P, \emptyset)$ or $(P, \{u\})$, c will be marked as forbidden since r' is still α-critical in $T'_a - u$ (u is not in T'_a). The only difference between the two runs of the algorithm is that I_{T_a} will be strictly smaller in the latter case. Hence the requirement of the lemma holds, and going forward we can assume no such cycle vertex exists.

If there is a cycle vertex c with two attached trees T_a and T_b of types (a) and (b) respectively, then there are elements $u \in Z \cap V(T_a)$ and $v \in Z \cap V(T_b)$ such that $\alpha(T_a) > \alpha(T_a - u)$ and r_b is α-critical in $T_b - v$. Note that then $\alpha(P) > \alpha(P - \{u, v\})$. To see this, consider what happens in Algorithm 4.24: Regardless of whether it was called with input $(P, \emptyset)$ or $(P, \{u, v\})$, c will be marked as forbidden because r_a is α-critical in T, and r_b is α-critical in $T_b - v$. Since $I_{T_a}^{(P,\{u,v\})}$ is strictly smaller than $I_{T_a}^{(P,\emptyset)}$, we have reached the requirement of the lemma. From now on we assume there are no such cycle vertices.

We can now partition all the cycle vertices $c \in C$ into the following three categories:

Redeemable A cycle vertex $c \in C$ is *redeemable* if it has exactly one attached tree T of type (a), and all other attached trees are of type (c). We know there exists a singleton $u \in Z \cap V(T)$ such that c is *redeemed*, i. e. such that T will have its root r not α-critical in $T - u$. Informally we may note that the price payed in T for redeeming c is at least one by Observation 4.20, i. e. Algorithm 4.24 yields $|I_T^{(P,\emptyset)}| > I_T^{(P,\{u\})}$.

Blockable A cycle vertex $c \in C$ is *blockable* if it has at least one attached tree T of type (b). It may also have any number of attached trees of type (c). We know there exists a singleton $u \in Z \cap V(T)$ such that c is *blocked*, i. e. such that the root r of T will be α-critical in $T - u$.

Free A cycle vertex $c \in C$ is *free* if all attached trees are of type (c).

If there are no redeemable cycle vertices in C, then consider some blockable cycle vertex c_u. It exists, or else we have that $\alpha(P) = \alpha(P - Z)$. Because there are no trees where Z cause a drop in the independence number of the tree, the difference must occur in $|I_C|$ when using Algorithm 4.24 on the inputs $(P, \emptyset)$ and (P, Z). By Lemma 4.16, we can extrapolate that there exist two (possibly non-distinct) blockable cycle vertices $c_v, c_w \in C - c_u$ such that $\alpha(C - c_u) > \alpha(C - \{c_u, c_v, c_w\})$. If we now let u, v, w be elements that block their respective cycle vertex c_u, c_v, c_w, then we have that $\alpha(P) > \alpha(P - \{u, v, w\})$, and the lemma is satisfied. Going forth, we will assume there is at least one redeemable cycle vertex.

If there is exactly one redeemable cycle vertex c, observe that Algorithm 4.24 will pick $\alpha(C)$ vertices of C when finding the MIS for input $(P, \emptyset)$. Let T be the tree of type (a) attached to c. See that there is no way to compensate the cost occurring in I_T when u redeems c. Thus, $\alpha(P) > \alpha(P - u)$, and the lemma holds. Assume then, there are at least two redeemable cycle vertices.

If there are exactly two redeemable cycle vertices c_1 and c_2, then similarly to the previous case, Algorithm 4.24 will pick at least $\alpha(C) - 1$ vertices of C on input $(P, \emptyset)$. If we pick u, v such that they redeem c_1 and c_2 respectively, then the price payed is at least two, out of which at most one can be compensated in C. Thus, $\alpha(P) > \alpha(P - \{u, v\})$, and the lemma holds. Assume then, there are at least three redeemable cycle vertices.

Pick two redeemable vertices c_i and c_j and a maximal length path along the cycle $Q = \{c_1, c_2, \cdots, c_{i-1}, c_i, c_{i+1}, \cdots, c_{j-1}, c_j, c_{j+1}, \cdots, c_q\}$ such that c_i and c_j are the only redeemable vertices in Q. Since there are at least three redeemable vertices in C, such a path exists, and c_1 is not a neighbor of c_q. Let $\alpha(Q)$ be the maximum independent set of Q after c_i and c_j have been both redeemed. It remains to observe that $\alpha(Q) \leq 1 + \alpha(Q - \{c_i, c_j\})$ to conclude that redeeming c_i by u and c_j by v will cause $\alpha(P) > \alpha(P - \{u, v\})$. This concludes the proof of the lemma. $\square$

Chapter 5

Conclusions

In this section we will give a summery of the thesis, give some comments on the progress of writing the thesis, and provide some open questions for future work.

5.1 Summary

In this thesis we have studied parameterized kernelization algorithms for the Vertex Cover problem, which is an approach for dealing with the NP-hardness of the problem by relaxing that all instances are solved in polynomial time. We have examined some parameterized algorithms for solving Vertex Cover, and seen examples of how to obtain a linear kernel when parameterizing by the natural parameter, the size of the solution.

We have discovered how some parameters are structurally related to each other, which motivates the search of kernels parameterized by structurally stronger parameters. As an introductory example, we showed that Vertex Cover yields a cubic kernel when parameterized by the size of a minimum degree-1 modulator. This is a simplified version of the cubic kernel for Vertex Cover parameterized by a the size of a minimum feedback vertex set given by Jansen and Bodlaender [20].

We have then presented two new results, namely a kernel for Vertex Cover/Degree-2 Modulator on $\mathcal{O}(|X|^7)$ vertices, and a kernel for Vertex Cover/Pseudoforest Modulator on $\mathcal{O}(|X|^{12})$ vertices. Independently from our work, Majumdar et al. showed another polynomial kernel for VC/D2M on $\mathcal{O}(|X|^5)$ vertices. Combining this with related kernelization results, we get a parameter ecology for Vertex Cover shown in Table 5.1.

Parameter k	Kernel upper bound (number of vertices)
Minimum vertex cover	$2k$ [3]
Minimum degree-1 modulator	$\mathcal{O}(k^2)$ [25]
Minimum feedback vertex vet	$\mathcal{O}(k^3)$ [20]
Minimum degree-2 modulator	$\mathcal{O}(k^5)$ [25]
Minimum pseudoforest modulator	$\mathcal{O}(k^{12})$
Minimum treewidth-2 modulator	No polynomial kernel unless NP $\subseteq$ coNP/poly [8]

Table 5.1: Parameter ecology for Vertex Cover

We have also given a $(d+2)$–approximation algorithm for Degree-d Modulator intended for use by the practitioner, which is found in the appendix.

5.2 Notes on the Progress

We started the work by reading and understanding the paper by Jansen and Bodlaender [20] which gave a cubic kernel for Vertex Cover parameterized by a feedback vertex set. It follows from this result that there is also a polynomial kernel when parameterizing by the size of a degree-1 modulator. However, it remained open if there also was a polynomial kernel when parameterizing by a degree-2 modulator, which is another parameter structurally stronger than D1M.

The first step was to simplify the kernelization of [20] to accommodate only a degree-1 modulator. This is shown is section 2.5. Our next task was then to come up with a solution to the degree-2 modulator version.

The first idea that appeared, was the one of breaking apart the cycles in the analysis of the reduced instances by introducing the set $\hat{X}$ containing one vertex of every cycle. However, the possibility of odd cycles in F posed a problem even before we could apply such a rule, since the reduction which originally removed connected components (Reduction 2.3) was unsafe.

For a starter, it was not sufficient to consider only chunks of size at most two, since the independence number of odd cycles will not drop if only two of its vertices are forbidden to be picked. Thus, the collection of chunks were expanded to include subsets of X of size three. This in turn posed challenges in the analysis, which were circumvented by adding another restriction to the definition of chunks, namely that they had a conflict smaller than $|X|$. At first we didn't find any problems with this change.

But then, whilst confidently working out the finer details of our newfound kernel, we suddenly discovered that removing a connected component could potentially turn a non-chunk into a chunk in the reduced instance, which previously was not possible. This invalidated the safeness of the reduction, and a fix was required.

We began by observing that the kernel could still work for *another* problem, namely the variation of Independent Set where also a set of forbidden triples is provided with the input. If we could find a reduction from this problem to our problem (which we knew existed by the Cook-Levin theorem, seeing that Independent Set is NP-complete), then we could still say something interesting. It turned out that there is very a nice reduction using what we call "anchor triangles," which we could also incorporate directly into our kernelization procedure. The kernel was luckily saved, and is found in Chapter 3.

Our next step was asking ourselves whether we could apply the same techniques in order to obtain a kernel for Vertex Cover/Pseudoforest Modulator. The answer turned out to be yes, but with a twist. In the analysis of reduced instances, it is important that we can argue that the number of vertices in the pseudoforest is polynomially bounded by the number of chunks, and in order to do so the argument by Jansen and Bodlaender for the FVS case require that each structure of a certain type in F has triggered a certain reduction. The claim is unfortunately not true for pseudoforests, even when we introduce the set $\hat{X}$ in the analysis. The solution was to introduce the set *before* the analysis, at the cost of some factor in the exponential. This kernel is found in Chapter 4.

Pseudotrees also required a much more elaborate proof to show that chunks of size 3 were sufficient to witness the existence of obstructions in X to picking a MIS for the pseudotree.

The journey certainly as has had its ups and downs, but overcoming the setbacks were in the end what made it such an exciting experience.

5.3 Open Questions

In working with this thesis, we have encountered some problems which beseech further research:

- *Upper bounds.* Can the upper bound for the kernels demonstrated in this thesis be improved? In particular, does there exist a kernel on at most $\mathcal{O}(|X|^7)$ vertices for Vertex Cover/Pseudoforest Modulator?
- *Closing the gap.* It was recently showed that VC/D1M does not admit a kernel on $\mathcal{O}(|X|^{2-\epsilon})$ vertices, and that VC/D2M does not admit a kernel on $\mathcal{O}(|X|^{3-\epsilon})$ vertices, for any $\epsilon > 0$ unless NP $\subseteq$ coNP/poly [25]. This also implies that VC/FVS is unlikely to have a kernel on $\mathcal{O}(|X|^{2-\epsilon})$, and VC/PFM is unlikely to have a kernel on $\mathcal{O}(|X|^{3-\epsilon})$ for any strictly positive ϵ. This yields an open gap for all four problems except VC/D1M.
- *Obstruction to kernelization.* What characterizes a structural parameter which yields a polynomial kernel to Vertex Cover? Why does a modulator to pseudoforest yield a polynomial kernel, whereas a modulator to treewidth 2 do not? Does Vertex Cover admit a polynomial kernel when parameterized by the size of a minimum modulator to cactus graphs?
- *Technique application.* Can the techniques for finding polynomial kernels for Vertex Cover and Independent Set demonstrated in this thesis be applied to other problems and/or parameters? Does the Degree-1 Modulator problem admit a polynomial kernel parameterized by the size of a minimum degree-2 modulator?

5.4 Afterthought

There are many complex problems in the world, but most of them has some inherent structure, or is not very far from one. As decision makers we are all algorithm designers, so we should not be discouraged when faced with a seemingly impossible task; it might just be tractable if we find the right parameter.

Appendix A

Approximation for Degree-d Modulator

We will here give a $(d+2)$–approximation algorithm for the DEGREE-D MODULATOR problem, which can be implement to run in time $\mathcal{O}(n+m)$ on a graph with n vertices and m edges. The algorithm is a generalization of the well-known 2-approximation to VERTEX COVER by Gavril [15, page 134], and for the case when $d=0$ the algorithms are identical.

Algorithm A.1 $(d+2)$–approximation for DEGREE-D MODULATOR

```
Input: Graph $G$, non-negative integer $d$.
Output: A vertex set $X \subseteq V(G)$ such that $\triangle(G-X) \leq d$, and such that $|X| \leq (d+2) \cdot |X^\star|$
where $X^\star$ is a minimum degree-d modulator for $G$.

1: $X := \emptyset$
2: $G' := G$
3: while $\triangle(G') > d$ do
4:     Pick $v \in V(G')$ arbitrarily such that $deg_{G'}(v) > d$.
5:     Pick $d+1$ distinct neighbors of $v$ arbitrarily, $u_1, u_2, \ldots, u_{d+1} \in N_{G'}(v)$.
6:     $X := X \cup \{v, u_1, u_2, \ldots, u_{d+1}\}$
7:     $G' := G - X$
8: end while
9: return $X$
```

Lemma A.2. Algorithm A.1 is correct: Given as input a graph G and an integer d, the algorithm will output a degree-d modulator X to G which is contains at most $(d+2) \cdot |X^\star|$ vertices, where $X^\star$ is a minimum degree-d modulator to G.

Proof. Let the number of iterations of the while-loop be denoted by k. Observe that $|X| = (d+2) \cdot k$. It remains to show that $X^\star \geq k$ to conclude the proof.

For each iteration i of the loop, let X_i be the vertices added to X, and let v_i be the vertex that was chosen at line 4 at that particular iteration. We will now show that $X^\star$ contains at least one element from X_i for every $i \in \{1, 2, \ldots, k\}$.

Assume for the sake of contradiction that there is some X_i such that $X_i \cap X^\star = \emptyset$. But then v_i has at least $d+1$ neighbors in $G - X^\star$, which contradicts that $X^\star$ is a degree-d modulator. Since every X_i is disjoint, we can thus conclude that $|X^\star| \geq k$, and the lemma holds. $\square$

Lemma A.3. Algorithm A.1 can be implemented to run in time $\mathcal{O}(n+m)$ on an arbitrary graph on n vertices and m edges.

Proof. Given an adjacency list representation of G, one possible implementation is to iterate once over all the vertices of G, and then for each vertex $v \in V(G)$, iterate over its neighbors to see if there are more than $d+1$ of them which are not yet marked to be in X. Whenever such a vertex is found where v itself is also unmarked, it and its first $d+1$ unmarked neighbors are marked to be in X. After this loop is finished, do a single iteration of the vertices and include the marked ones in the set X. This implementation will visit each vertex twice and follow each edge at most twice, thus obtaining a runtime of $\mathcal{O}(n+m)$. $\square$

Bibliography

[1] R. Balasubramanian, Michael R. Fellows, and Venkatesh Raman. An improved fixed-parameter algorithm for vertex cover. *Information Processing Letters*, 65(3):163–168, 1998.

[2] Jonathan F. Buss and Judy Goldsmith. Nondeterminism within P. *SIAM J. Computing*, 22(3):560–572, 1993.

[3] Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: further observations and further improvements. *J. Algorithms*, 41(2):280–301, 2001.

[4] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.

[5] Benny Chor, Michael R. Fellows, and David W. Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In *Proceedings of the 30th Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 3353 of *Lecture Notes in Comput. Sci.*, pages 257–269. Springer, 2004.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

[7] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Dániel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2016.

[8] Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. On the hardness of losing width. *Theory Comput. Syst.*, 54(1):73–82, 2014.

[9] George B Dantzig and Mukund N Thapa. *Linear programming 1: introduction*. Springer Science &amp; Business Media, 2006.

[10] Sanjoy Dasgupta, Christos H Papadimitriou, and Umesh Vazirani. *Algorithms*. McGraw-Hill, Inc., 2006.

[11] Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 3rd edition, 2005.

[12] Michael R. Fellows, Bart M. P. Jansen, and Frances A. Rosamond. Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *Eur. J. Comb.*, 34(3):541–566, 2013.

[13] Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Matthias Mnich, Frances A. Rosamond, and Saket Saurabh. The complexity ecology of parameters: An illustration using bounded max leaf number. *Theory of Computing Systems*, 45(4):822–848, 2009.

[14] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar F-deletion: Approximation, kernelization and optimal FPT algorithms. In *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 470–479. IEEE, 2012.

[15] Michael R. Garey and David S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979.

[16] Jiong Guo and Rolf Niedermeier. Linear problem kernels for NP-hard problems on planar graphs. In *Proceedings of the 34th International Colloquium of Automata, Languages and Programming (ICALP)*, volume 4596 of *Lecture Notes in Comput. Sci.*, pages 375–386. Springer, 2007.

[17] Philip Hall. On representatives of subsets. *J. London Math. Soc.*, 10:26–30, 1935.

[18] Juraj Hromkovič. *Algorithmics for hard problems: introduction to combinatorial optimization, randomization, approximation, and heuristics.* Springer Science &amp; Business Media, 2013.

[19] Bart M. P. Jansen and Hans L. Bodlaender. Vertex cover kernelization revisited: Upper and lower bounds for a refined parameter. In *Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 9 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 177–188. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[20] Bart M. P. Jansen and Hans L. Bodlaender. Vertex cover kernelization revisited - upper and lower bounds for a refined parameter. *Theory Comput. Syst.*, 53(2):263–299, 2013.

[21] Bart M. P. Jansen and Stefan Kratsch. Data reduction for graph coloring problems. *Inf. Comput.*, 231:70–88, 2013.

[22] Bart M.P. Jansen, Venkatesh Raman, and Martin Vatshelle. Parameter ecology for feedback vertex set. *Tsinghua Science and Technology*, 19(4):387–409, Aug 2014.

[23] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Plenum Press, New York, 1972.

[24] Christian Komusiewicz and Rolf Niedermeier. New races in parameterized algorithmics. In *Mathematical Foundations of Computer Science 2012*, pages 19–30. Springer, 2012.

[25] Diptapriyo Majumdar, Venkatesh Raman, and Saket Saurabh. Kernels for structural parameterizations of vertex cover - case of small degree modulators. in press, 2015.

[26] George L Nemhauser and Leslie E Trotter Jr. Vertex packings: structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, 1975.

[27] Kenneth Rosen. *Discrete Mathematics and its applications.* McGraw-Hill, 1999.

[28] Alexander Schrijver. *Combinatorial optimization. Polyhedra and efficiency. Vol. A.* Springer-Verlag, Berlin, 2003.

[29] Michael Sipser. *Introduction to the Theory of Computation.* International Thomson Publishing, 1st edition, 1996.

[30] Robert Roth Stoll. *Set theory and logic.* Courier Dover Publications, 1979.