Polynomial Kernels for Graph and Hypergraph Optimisation Problems

Gabriele Muciaccia

A thesis submitted to Royal Holloway, University of London for the degree of Doctor of Philosophy in the Faculty of Science

Department of Computer Science
August 2014

Declaration

I Gabriele Muciaccia hereby declare that this thesis and the work presented in it is entirely my own. Where the work is a result of collaborative research, or the work of others has been used, this is clearly stated.

Abstract

When dealing with hard problems, problems which we are not able to solve in polynomial time, it is common practice to preprocess an instance to reduce its size and make the task of solving it easier. Parameterized Complexity offers a theoretical framework to prove the efficiency of preprocessing algorithms, which are called kernelizations. The underlying idea is that for every problem we identify a parameter which represents the part of the input which is 'difficult' to solve, then we try to reduce the input size and we measure the result in terms of the parameter.

Especially successful kernelizations are the ones that compute a kernel whose size is bounded by a polynomial in the parameter. In this thesis we consider some combinatorial optimisation problems on graphs and hypergraphs, and we study the existence (or non-existence) of polynomial kernels for these problems. In particular, we describe a generic kernelization for a theoretical class of graph problems, which can be used to derive the existence of a polynomial kernel for many graph problems of interest.

Acknowledgements

First of all, I want to thank my supervisors, Gregory Gutin and Anders Yeo, who helped me immensely during these past three years. In particular, I would like to thank Gregory for his availability and for his continuous support, both for academic and personal issues, and I would like to thank Anders for always trying to patiently answer my many questions.

Furthermore, I would like to thank all the persons I worked with for the many inspiring discussions and fruitful hours spent together: Robert Crowston, Gregory Gutin, Mark Jones, Fahad Panolan, Geevarghese Philip, Ashutosh Rai, M.S. Ramanujan, Saket Saurabh, and Anders Yeo. In particular, I would like to thank my senior PhD colleagues Robert Crowston and Mark Jones, whom it has been a pleasure working with. General thanks go to everyone I met at Royal Holloway during these years, and I would like to especially thank Janet Hales for all her help.

My thanks also go to my former teachers, who were a source of inspiration and encouraged me to persist in this path. Among the others, I would like to mention René Schoof, Ilaria Damiani, Francesca Tovena, Lorenzo Tortora de Falco, Lionel Vaux, Paul Ruet, Frédéric Havet.

Obviously I thank all of my friends and flatmates, present and past, far and close, whom I owe so much. Thanks to Giulio and Gug for sharing many difficult and stressful moments before the exams. Thanks to Martina for the long chats whenever I grew nostalgic. Thanks to Ele, Jer, Max and Ale for having made a home of this house. And also thanks to Spatto for having inspired so many of my goals.

My deepest gratitude goes to my family, my parents and my brother. I know that come what may, I can always count on a bowl of minestra when I need. And special thanks to my grandmother, who does not grow tired of adding flags on the map.

This thesis is dedicated to the two persons without whom I would have never even suspected the beauty of Maths, my aunt Baba and my friend Natalia.

Contents

Chapter 1

Introduction

1.1 What is Parameterized Complexity

In the course of the twentieth century humankind became able to create machines that could perform calculations at a speed that would have been inconceivable before. The appearance of the first prototypes of these machines was preceded by the formulation of a theoretical model of computation, known as Turing machine, which was introduced in 1936 by British mathematician Alan Turing.

This model was intended to formalize the notion of functions that are 'effectively calculable'. The Church-Turing hypothesis, in fact, states that such functions correspond precisely to the functions that can be computed by a (universal) Turing machine, the so-called *computable* functions.

During the same period, other definitions were given for the class of computable functions and, despite their apparent dissimilarity, they were proven to be equivalent. Hence, the Church-Turing hypothesis became generally accepted and it has kept its status practically unchallenged until now.

Nonetheless, it was soon clear that not all computable functions were equivalent from a practical point of view; in particular, though effectively calculable in theory, there were some that appeared to be somewhat more difficult to compute in practice.

The problem is that the definition of computable function would not take into account the physical resources needed to perform the calculation. In particular, the amount of required time and memory is a key aspect: once it is clear that a problem can be solved using a computable function, it is of primary importance to optimise it as much as possible. In other words, even when a problem can be solved using a mechanical procedure which requires a finite amount of resources, we may not have a sufficient amount of them available, and this could be a limitation as serious as not having a procedure at all.

In order to be able to compare different problems according to how efficiently they can be solved, in the 1960s it was suggested to relate the time and memory needed to the length of the input, in a paper by Hartmanis and Stearns which is considered the starting point of modern Computational Complexity theory [53]. It was proposed, and later became generally accepted, that feasible problems corresponded (at least to a certain extent) to the ones that could be solved using an amount of time bounded by a polynomial in the input size [15], that is, the problems that are in the complexity class P. This is known as the Cobham's thesis.

The realm of infeasibility instead became associated with the complexity class NP (and complexity classes beyond). Strictly speaking, there was and there still is no proof that $P \neq NP$. Problems in NP are such that, given a candidate solution, it is easy (polynomial time solvable) to check whether it is a correct solution. For some of them, though, despite this we do not know an efficient way to solve them, neither do we believe there exists one. Examples of these problems are the NP-complete problems: it is well-known that finding an efficient solution to one of them would enable us to find an efficient solution to every other and, in particular, would imply that $P = NP$.

In the 1970s, Karp showed that many problems, considered interesting because of their practical applications, are in fact NP-complete [62]. This gave rise to the increasing interest in the study of Computational Complexity, which led to the appearance of many other complexity classes and theoretical methods to classify problems.

Nonetheless, the relation between P and NP remained the central question. If P is a proper subclass of NP as it is generally believed, to solve problems which are NP-complete a superpolynomial (in the input size) amount of time is required, no matter which method is used. Unfortunately, from the point of view of feasibility this represents a big difficulty, as even small input sizes entails astronomic running times and even small increases in the input size entails huge variations in the required computation time.

However, since many problems of interest, starting from the ones contained in Karp's list, have been shown to be NP-complete over the years, different methods were devised to obtain a solution with a practical amount of resources. These methods include approximation, probabilistic and heuristic approaches.

The main idea of approximation is to sacrifice the optimality of the solution in order to obtain a better running time; at the same time, some bound on the size of the computed solution in terms of the optimal one is required. Probabilistic approaches aim to reduce the running time too, allowing a certain degree of randomness that may cause the answer to be wrong, or not optimal, with some (small) probability. Finally, heuristic approaches are methods that perform well in practice, but without proof of their correctness or without a theoretical bound on their running time.

The common feature of these methods is that they generally do not produce optimal solutions, in the sense that they often produce just an approximate solution within practical amount of time.

In this respect, Parameterized Complexity can be considered as a natural counterpart to these approaches. As a matter of fact, in Parameterized Complexity one looks for *exact* solutions for hard problems, where by hard it is generally intended NP-complete or worse. Of course, assuming $P \neq NP$, to achieve this one has to sacrifice the polynomial bound on the running time. For a long time, this was considered a synonym of infeasibility. Instead, the core idea behind Parameterized Complexity is that it is possible in some cases to harness the superpolynomial factor in the running time, making it dependent only on a *parameter* of the input, which is (hopefully) small compared to the total size. In this way, at least in theory even large instances can be efficiently solved when the associated parameter is small enough.

Compared to approximation, probabilistic or heuristic approaches, Parameterized Complexity is a fairly new area. Its late appearance in the field of Computational Complexity is probably partly due to the wariness that superpolynomial running times always arouse. Nevertheless, Parameterized Complexity proved useful in a way that goes beyond the one already mentioned. In particular, the notion of *kernel* has proved to be one of the finest contributions to the difficult art of solving resource-demanding problems.

1.2 Basic definitions

We will assume the reader familiar with classical Computational Complexity theory, in particular with the notion of model of computation (Turing machine, Random Access Machine...), with the asymptotic notation (big $\mathcal{O}$, small o, big Ω) and with the most known complexity classes like P and NP.

We will denote by Σ an *alphabet* of finite size, that is, a set of symbols which will form the 'letters' which languages we consider are written with. For an alphabet Σ, the set $\Sigma^* = \{l_1 \dots l_n : n \in \mathbb{N} \; and \; l_i \in \Sigma \; for \; 1 \leq i \leq n\}$ is the set of all possible words of finite length that can be written using symbols in Σ. The length, or size, of an element $x = l_1 \dots l_n \in \Sigma^*$ is n and it is denoted by $|x|$. Similarly, the size of an integer $n \in \mathbb{N}$ is the number of digits that are used in a unary representation of n, that is, precisely n. Finally, the size of a string $(a_1, \dots, a_n) \in A_1 \times \dots \times A_n$ is defined as $|(a_1, \dots, a_n)| = \sum_{i=1}^{n} |a_i|$.

A *formal language* L, also called just *language*, is a subset of Σ^*. A *problem* can be seen as a language L on a finite alphabet Σ: in fact, the two words will be to all purposes considered interchangeable.

In classical Computation Complexity, the objective is to find 'recipes to solve problems', or, more precisely, to find *algorithms to recognise languages*. Such algorithms could be described as mechanical procedures that, given an element $x \in \Sigma^*$, called the *input*, check whether $x \in L$ and answer `YES` or `NO` accordingly. When an algorithm exists, the next question is whether it can be 'improved', where a better algorithm is an algorithm that requires less physical resources (time and memory, usually) to be computed.

The drawback of the classical approach is that once a problem is shown to be NP-hard little can be done to further classify its complexity. Parameterized Complexity instead offers the tools to distinguish between 'hard' problems and 'even harder' ones, and in doing so also helps to understand where the hard core of the problem lies and to find feasible ways of dealing with it [76].

Parameterized Complexity has sometimes been described as a two-dimensional complexity analysis [75]. The input of a parameterized problem is composed of two parts: the first one is the input of the problem in the classical sense and the second one is the *parameter*, a part of the input which is deemed to account for the hard nature of the problem. The parameter, in this sense, could literally be anything, from the size of the solution to the treewidth of a graph: in fact, it can be defined as an element of Σ^* [30]. Nevertheless, for the purposes of this thesis, it is enough to restrict the parameter to be an integer, but the reader should bear in mind that different definitions are possible [1].

Definition 1.1. A *parameterized problem* Q is a subset of $\Sigma^* \times \mathbb{N}$. An *instance* of a parameterized problem is an element $(x, k) \in \Sigma^* \times \mathbb{N}$, where k is the *parameter*. An instance which is in Q is a `YES`-instance and every other instance is a `NO`-instance.

[1] In particular, sometimes more than one parameter is considered for a problem, in which case it is possible to define the parameter as an element of $\mathbb{N}^t$, with $t \in \mathbb{N}^+$.

When it is clear from the context, a parameterized problem will be just called a *problem*.

The fact that for every instance of a parameterized problem a parameter is specified does not mean, generally speaking, that any additional information is given: the parameter is often implicitly contained in the classical version of the input (though it is not always the case). The parameter should be thought of as a measure of the complexity of an instance.

Definition 1.2. Let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. The problem is said to be *fixed-parameter tractable*, or FPT, if there exists an algorithm $\mathcal{A}$ which for every $(x, k) \in \Sigma^* \times \mathbb{N}$ checks whether $(x, k) \in Q$ and whose running time is bounded by $f(k)|x|^{\mathcal{O}(1)}$, where f is a computable function from $\mathbb{N}$ to $\mathbb{N}$.

The algorithm $\mathcal{A}$ is called an FPT-algorithm.

The complexity class that contains all the parameterized problems that can be solved by an FPT-algorithm is denoted FPT. It is clear from the definition that a fixed-parameter tractable problem Q can be solved in polynomial time for every fixed value of the parameter. In other words, the language $Q_k = \{(x, k') \in Q : k' = k\}$, where k is fixed, also known as the *k-th slice* of Q, lies in P for every $k \in \mathbb{N}$.

However, this is not the defining property of FPT problems, as even an algorithm with running time $|x|^{g(k)}$, $g : \mathbb{N} \to \mathbb{N}$, satisfies this property but not the requirements of Definition 1.2. As a matter of fact, this property describes the complexity class XP, which the class FPT is a proper subclass of [30].

In Parameterized Complexity the role of the class FPT is the counterpart of the role of the class P in classical Computational Complexity. Algorithms with running time bounded by $f(k)|x|^{\mathcal{O}(1)}$, though impractical in the classical sense, proved to be feasible in cases of interest [1, 57, 16], as opposed to algorithms of running time of the order of $|x|^{g(k)}$.

The parameterized counterpart of the class NP is more difficult to describe. In Section 1.4 we will give a brief introduction on this subject.

Before concluding this section, we define the *unparameterized version* of a parameterized problem.

Definition 1.3. Let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. The unparameterized version $\widetilde{Q}$ of Q is the language $\{x\#1^k : (x, k) \in Q\}$, where $\#$ is a new character and 1 is an arbitrary letter in Σ.

1.3 Kernels

The notion of kernel is closely related to the notion of preprocessing. When a problem must be solved on large instances, or when the algorithm to find a solution is not particularly efficient, it is often convenient to simplify the instance, firstly tackling the easier part of the problem. In the framework of Parameterized Complexity it is possible to make this idea rigorous.

Definition 1.4. Let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. The problem is said to admit a *kernel* if there exists an algorithm $\mathcal{K}$ which takes as input an element $(x, k) \in \Sigma^* \times \mathbb{N}$ and in time polynomial in $|x|$ and k outputs an element $(x', k') \in \Sigma^* \times \mathbb{N}$ such that

- $(x, k) \in Q$ if and only if $(x', k') \in Q$,
- $|x'|$ is bounded by $g(k)$ for some function $g : \mathbb{N} \to \mathbb{N}$,
- $k' \leq k$.

The algorithm $\mathcal{K}$ is a *kernelization* for Q and the function g is the *size* of the kernel.

The analysis of polynomial preprocessing techniques was generally overlooked before the introduction of the formalism of Parameterized Complexity. This is due to the fact that being able to unconditionally reduce in polynomial time the size of an instance for an NP-hard problem would mean that $P = NP$: in fact, applying multiple times the reduction it would be possible to reduce the problem to an instance of constant size in polynomial time, thus solving it [71]. Parameterized Complexity resolves this dilemma because the size reduction can be defined *with respect to the parameter*, negating the possibility of reducing it by an arbitrary amount.

Polynomial preprocessing has been widely used to deal with computationally hard problems, not only for heuristic algorithms but also for approximation and probabilistic ones. Using the notion of kernel it is possible to prove theoretical bounds for the efficiency of such preprocessing. Its usefulness is not limited to this, though: for instance, it has applications in cryptography, or as a technique to store and transmit large instances of hard problems [52].

Even limiting the attention to the field of Parameterized Complexity only, it is generally believed that "kernelization is the way of understanding fixed-parameter tractability" [71]. The next theorem partially explains why.

Theorem 1.5. *A parameterized problem $Q \subseteq \Sigma^* \times \mathbb{N}$ is fixed-parameter tractable if and only if it is decidable and admits a kernel.*

Proof. Assume first that Q is fixed-parameter tractable. In this case Q admits an FPT-algorithm $\mathcal{A}$ that solves it (hence the problem is decidable) and whose running time on an instance $(x,k) \in \Sigma^* \times \mathbb{N}$ is bounded by $f(k)|x|^c$, for a function $f : \mathbb{N} \to \mathbb{N}$ and a constant $c \in \mathbb{N}$. Then we can define a kernelization $\mathcal{K}$ of size f in the following way. Run $\mathcal{A}$ for at most $|x|^{c+1}$ steps: if this solves the problem, $\mathcal{K}$ outputs a trivial `YES` or `NO`-instance of constant size accordingly; otherwise, this means that $|x|^{c+1} < f(k)|x|^c$, that is, $|x| < f(k)$, then $\mathcal{K}$ just outputs the original instance.

Assume now that Q is decidable and admits a kernel. An algorithm which first applies the kernelization $\mathcal{K}$ and then solves the problem using a brute-force search is an FPT-algorithm, hence Q is fixed-parameter tractable. □

Theorem 1.5 is not useful in practice, as the size of the kernel it produces is usually too large, but it shows the central role that kernelization plays in the study of parameterized problems.

The efficiency of a kernelization depends on its running time and on its size. In this thesis we will consider mainly the latter; in particular, we will be often interested in *polynomial kernels*, that is kernels for which the size is bounded by a polynomial.

Sometimes, it is easier to devise a kernelization that produces an equivalent instance of *a different problem*. In the literature, this is sometimes called a *bikernel* [2].

Definition 1.6. Let $Q, Q' \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems. The problem Q is said to admit a *bikernel* if there exists an algorithm $\mathcal{K}$ which takes as input an element $(x,k) \in \Sigma^* \times \mathbb{N}$ and in time polynomial in $|x|$ and k outputs an element $(x',k') \in \Sigma^* \times \mathbb{N}$ such that

- $(x,k) \in Q$ if and only if $(x',k') \in Q'$
- $|(x',k')|$ is bounded by $g(k)$ for some function $g : \mathbb{N} \to \mathbb{N}$.

The algorithm $\mathcal{K}$ is a *bikernelization* for Q and the function g is the *size* of the bikernel.

A kernel for a problem Q is obviously also a bikernel. On the other hand, note that Theorem 1.5 could be rewritten as "Q is FPT if and only if it admits a bikernel where the target problem is decidable": the proof is essentially the same. Hence, if we assume we are dealing with decidable problems, a problem admits a kernel if and only if it admits a bikernel, but the size of the kernel obtained via this reasoning is superpolynomial in the size of the bikernel, which makes it uninteresting from a practical point of view.

Being able to prove the existence of a small size kernel for a problem is indeed an important achievement, but there are problems for which we do not believe that such kernels can be found. Nonetheless, this does not necessarily mean that we have to lose hope, there are notions more general than the one of kernel, which may still be useful in practice. In particular, though not directly related to the subject of this thesis, we give the definition of Turing kernel [69], which we believe will have a growing importance, particularly because of its ties with the theory of parallel algorithms.

Definition 1.7. Let Q be a parameterized problem. A t-oracle for Q is an oracle that takes as input (x, k) with $|x| \leq t$, $k \leq t$ and decides whether $(x, k) \in Q$ in constant time.

Definition 1.8. Let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem and let $g : \mathbb{N} \to \mathbb{N}$ be any function. The problem Q is said to admit a *Turing kernel* if there exists an algorithm $\mathcal{T}$ which takes as input an element $(x, k) \in \Sigma^* \times \mathbb{N}$ together with a $g(k)$-oracle for Q and decides whether $(x, k) \in Q$ in time polynomial in $|x|$ and k. The function g is the *size* of the Turing kernel.

A problem that admits a kernel admits a Turing kernel. In addition, a problem that admits a polynomial number of independent kernels admits a Turing kernel too: such a situation may still be practically feasible if parallel computation is employed and this is in fact the original motivation for the definition of Turing kernel [8, 69]. Note that there exist problems which admit a Turing kernel of this kind but they are unlikely to admit a kernel in the classical sense [38, 80]. Nevertheless, Turing kernels may be even more involved, as for some problems it is not clear whether they admit a polynomial number of independent kernels, though they do admit a Turing kernel [82].

1.3.1 Practical tools for kernelization

Roughly speaking, there are two different methods to show that a problem admits a kernel. The first one is to devise rules that, when applied to an instance, transform it into an equivalent instance of smaller size. The second one is to show that large instances can be easily solved. In most cases, a kernelization results from the combination of these two methods.

The basic step of an approach of the first type is the application of a *reduction rule*:

Definition 1.9. [*informal*] Let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. A Reduction Rule $\mathcal{R}$ for Q is an algorithm formed by one or more `if` statements, that takes $(x, k) \in \Sigma^* \times \mathbb{N}$ as input and outputs a possibly different instance (x', k') with $k' \leq k$.

In other words, a reduction rule checks whether some conditions apply and, if so, applies some transformations to an instance of a problem. A reduction rule can be applied *in polynomial time* when the algorithm runs in time polynomial in the size of the input. An instance (x, k) is *reduced under Reduction Rule* $\mathcal{R}$ when (x, k) does not satisfy the conditions for the application. We are only interested in reduction rules which are *valid*.

Definition 1.10. A Reduction Rule $\mathcal{R}$ is valid if it can be applied in polynomial time and if for every instance (x, k) it outputs an instance (x', k'), with $k' \leq k$, which is in Q if and only if (x, k) is.

As for the second method, it often depends upon devising structural properties of the input of a problem that can be checked in polynomial time and enables to immediately answer `YES` or `NO`. Most of the times, these properties are related to the size of the input; for instance, when dealing with graph problems, one may try to show that the problem can be easily solved if the size of the graph is large, compared to the parameter.

Observe that when in the description of a kernelization it is stated that a certain property characterizes the instance as a `YES` or `NO`-instance, it is implicitly intended that the kernelization returns a trivial equivalent instance of constant size.

1.4 Fixed-parameter intractability

To prove that a parameterized problem is fixed-parameter tractable is not necessary to use involved theoretical tools, but the situation changes when one tries to show that a problem is fixed-parameter *intractable*. An elaborate theory has been developed with the objective of obtaining a parameterized counterpart of the NP class, but it turned out that the world of parameterized intractability is generally more complex and varied than its unparameterized counterpart.

This section is by no means a thorough account of the subject, only very basic notions are presented and only in so far as it is needed for the rest of the thesis. See the following monographs for more information on the subject [30, 39, 75].

Before any other consideration, a clarification is needed about the definition of parameterized problem. Definition 1.1 is fine for general purposes, but in this context it is better to restrict the study to decision problems for which the parameter is a function of the input.

Definition 1.11. A parameterized problem Q is a subset of $\Sigma^* \times \mathbb{N}$ such that there is no $x \in \Sigma^*$ with $(x, k) \in Q$ and $(x, k') \in Q$ for $k \neq k'$.

In classical Computational Complexity the notion of *polynomial time many-one reduction* can be used to show that a problem is at least as difficult as another problem. A similar tool can be defined for parameterized problems.

Definition 1.12. Let $Q, Q' \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. Q is said to reduce to Q' by a *parameterized (many-one) reduction* if there exists an algorithm $\mathcal{A}$ which takes as input an element $(x, k) \in \Sigma^* \times \mathbb{N}$ and outputs an element $(x', k') \in \Sigma^* \times \mathbb{N}$, satisfying the following conditions:

- $(x, k) \in Q$ if and only if $(x', k') \in Q'$,
- $k' \leq f(k)$ for some computable function $f : \mathbb{N} \to \mathbb{N}$,
- the running time of $\mathcal{A}$ is bounded by $g(k)|(x, k)|^{\mathcal{O}(1)}$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.

The algorithm $\mathcal{A}$ is called a *parameterized reduction*.

If there is a parameterized reduction from Q to Q' and Q' is fixed-parameter tractable, then Q is too. At the same time, a parameterized problem which is fixed-parameter tractable admits a parameterized reduction to any other problem [39]. Hence, this is a sound tool to compare parameterized problems.

In the complexity classes we will consider here, the notions of hardness and completeness will be always defined using parameterized reductions, in the same way it is done in classical Computational Complexity with polynomial time many-one reductions.

At the end of Section 1.2 it was mentioned that an algorithm whose running time on input (x, k) is of the order of $|x|^{g(k)}$, $g : \mathbb{N} \to \mathbb{N}$, is considered intractable from a parameterized point of view.

Definition 1.13. The class XP contains all the parameterized problems Q for which there exists a computable function $g : \mathbb{N} \to \mathbb{N}$ such that it can be verified in time $|x|^{g(k)}$ whether $(x, k) \in Q$.

Recall that FPT is a proper subclass of XP.

Another class which contains hard problems (from a parameterized point of view) is the following one:

Definition 1.14. The class *para-NP* contains all the parameterized problems Q for which there exists a non-deterministic Turing machine that verifies whether $(x, k) \in Q$ in at most $f(k)|x|^{\mathcal{O}(1)}$ steps, for some function $f : \mathbb{N} \to \mathbb{N}$.

The class *para-NP* is the analogous of the class NP. It contains FPT as a subclass, and $FPT =$ *para-NP* if and only if $P = NP$.

The following theorem provides a useful description of *para-NP*-complete problems. A parameterized problem Q is *nontrivial* if there exist $x \in \Sigma^*$ and $k \in \mathbb{N}$ with $(x, k) \in Q$, and there exists $x' \in \Sigma^*$ such that $(x', k') \notin Q$ for every $k' \in \mathbb{N}$. Also, recall that a slice of a parameterized problem Q is the language Q_k that contains the instances (x, k) in Q for a fixed k (see also Section 1.2).

Theorem 1.15. *[39] Let Q be a nontrivial parameterized problem in para-NP. Then the following statements are equivalent:*

- *Q is para-NP-complete,*
- *The union of finitely many slices of Q is NP-complete. That is, there are $l, m_1, \dots, m_l \in \mathbb{N}$ such that $Q_{m_1} \cup \dots \cup Q_{m_l}$ is NP-complete.*

Nonetheless, not all the problems that do not appear to admit an FPT-algorithm are XP-hard or *para-NP*-hard. In order to prove useful intractability results, it is necessary to refine the analysis. For this reason it was defined a sequence of classes, known as the *W-hierarchy*, which satisfies the following inclusions:

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \dots \subseteq W[SAT] \subseteq W[P] \subseteq XP \cap para - NP$$

It is not known whether the inclusions are all strict, but it is generally believed this is the case. Hence, under the reasonable assumption that $FPT \neq W[1]$, the boundaries between fixed-parameter tractability and fixed-parameter intractability lie between these two classes. Note that so far there are no results ensuring that a $W[1]$-complete parameterized problem can be solved faster than a $W[2]$-complete one, or even than an XP-complete one: in practice, the running times to solve $W[1]$-hard problems which lie in XP are of the order of $|x|^{g(k)}$, $g : \mathbb{N} \to \mathbb{N}$, for an instance $(x, k) \in \Sigma^* \times \mathbb{N}$.

The rigorous definition of the classes in the W-hierarchy requires some care and is beyond the scope of this thesis. Hence, here we will give only the definitions for the classes $W[1]$ and $W[2]$, since they are the only one which will be useful later.

Consider the following two parameterized problems:

Weighted 2-CNF-Satisfiability

Input: A Boolean formula in conjunctive normal form whose clauses have size at most 2 and an integer k.

Parameter: k

Question: Is there a satisfying truth assignment which has weight exactly k?

Weighted CNF-Satisfiability

Input: A Boolean formula in conjunctive normal form and an integer k.

Parameter: k

Question: Is there a satisfying truth assignment which has weight exactly k?

A Boolean formula is in conjunctive normal form if it is a conjunction of clauses, where each clause consists of a disjunction of literals, which are negated or non-negated Boolean variables. The weight of a satisfying truth assignment is the number of variables which are set to 'true'.

For both problems there is no known FPT-algorithm and it is widely believed that one cannot be found. These problems can be used to define the complexity classes $W[1]$ and $W[2]$ [75].

Definition 1.16. The class $W[1]$ contains all the parameterized problems that can be reduced to Weighted 2-CNF-Satisfiability by a parameterized reduction. A parameterized problem is $W[1]$-hard if Weighted 2-CNF-Satisfiability can be reduced to it. A parameterized problem in $W[1]$ which is $W[1]$-hard is $W[1]$-complete.

The class $W[2]$ is defined in the same way replacing Weighted 2-CNF-Satisfiability with Weighted CNF-Satisfiability.

Note that since Weighted 2-CNF-Satisfiability is a special case of Weighted CNF-Satisfiability, it immediately follows that $W[1]$ is contained in $W[2]$.

1.5 Kernel Lower Bounds

It is clear at this point that for a decidable problem admitting a kernel is equivalent to being fixed-parameter tractable. Nonetheless, for a kernel to be practically useful it is mandatory to have its size reduced as much as possible, hence all the effort in proving better and better kernel size upper bounds. In particular, a kernel whose size is bounded by a polynomial in the parameter seems particularly attractive.

However, it was soon clear to researchers that not all problems were likely to admit such polynomial kernels, but, despite this, for some time no theoretical tools were available to show lower bounds on kernel sizes.

The first results in this direction relied on known lower bounds for the approximation version of a problem, or on the notion of duality [75]. In the former case, if there exists a lower bound on the polynomial time approximability of a problem, and the parameter under consideration is the solution size, then there exists a similar bound on the size of a (possible) linear kernel: this is motivated by the observation that such a linear kernel would enable to produce a polynomial time approximation [75].

The notion of duality was introduced by Chen *et al.* [13] as a tool to show kernel lower bounds. The core idea is that if both a problem and one of its dual versions are fixed-parameter tractable, then a two-sided attack can be performed to efficiently solve an instance, thus providing lower bounds as too much 'efficiency' would mean an algorithm to solve an NP-hard problem running in polynomial time.

The definition of dual is as follows (see also section 4.1.3, where the notion of dual is used to prove kernel lower bounds in a different context):

Definition 1.17. Let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem and let $s : \Sigma^* \to \mathbb{N}$ be a mapping such that $0 \leq k \leq s(x)$ for every $(x, k) \in Q$ and $s(x) \leq |x|$ for every $x \in \Sigma^*$. The *s-dual* Q_s of Q is the parameterized problem corresponding to the language $Q_s = \{(x, s(x) - k) : (x, k) \in Q\}$.

The definition is slightly different from the one of Chen *et al.*, but the differences are not substantial and are intended only to make clearer that the dual of a problem is not unique, but depends on the size function s. Chen *et al.* used this notion to prove the following theorem:

Theorem 1.18 ([13])**.** *Let $Q \subseteq \Sigma^* \times \mathbb{N}$ be an NP-hard parameterized problem and s be a size function for it. Suppose that Q admits a kernel of size αk and Q_s admits a kernel of size $\alpha_s k_s$, where $\alpha, \alpha_s \geq 1$. If $(\alpha - 1)(\alpha_s - 1) < 1$ then $P = NP$.*

Proof (sketch). It is possible to write an algorithm that according to the value of the parameter uses the kernelization for Q or for Q_s, obtaining in both cases an instance whose size is strictly less than the size of the original instance. Clearly, a linear number of applications of such an algorithm produces an instance with constant size, thus effectively solving the problem in polynomial time. $\square$

It is clear though that both these approaches only provide lower bounds for kernels of linear sizes and, in addition, only work in a limited amount of cases. Unfortunately, in Computational Complexity theory it is often harder to prove lower bounds than to prove upper bounds. Only in 2009 it was produced a technique, using the concept of *compositional parameterized problem*, that could be used to prove the nonexistence of polynomial kernels for certain problems [9], under a widely believed computational complexity assumption [2].

This was a major breakthrough and eventually gave the possibility to further classify the complexity of fixed-parameter tractable problems, according to the size of the kernels they are expected to admit. Later, this technique was complemented with the tool of *polynomial parameter transformation* [10], and finally both notions were unified using the concept of *cross-composition* [6].

The rest of this section will be devoted to the description of this method and of some of its more recent generalizations.

1.5.1 Cross-composition

Cross-composition is the technique of encoding multiple instances of an NP-hard problem L into a single instance of a parameterized problem Q: 'composition' refers to the encoding of many into one, while 'cross' is due to the fact that L and Q may be different problems (while originally they had to be the same [9]). When a cross-composition is possible and, in addition, the problem Q admits a polynomial kernel, then this ensures the existence of a *distillation* for SAT (see Appendix A, Definition 1), which is deemed to be unlikely.

There are two different types of cross-composition, depending on whether the composition algorithm works as an `OR` gate or an `AND` gate. Nonetheless, traditionally cross-composition refers to the `OR`-cross-composition, as theoretical evidence against the existence of a polynomial kernel for parameterized problems which admit an `OR`-cross-composition was provided before its `AND` equivalent [40]. Hence, here cross-composition will implicitly refer to the notion of `OR`-cross-composition.

The following definition is a useful practical tool in the proofs of existence of a cross-composition:

[2]Note that one should expect complexity theoretical assumptions when proving lower bounds for the kernel size, as if $P = NP$ every parameterized problem admits a kernel of constant size, hence any such proof should at least assume $P \neq NP$.

Definition 1.19. An equivalence relation $\mathcal{R}$ on Σ^* is a *polynomial equivalence relation* if the following two conditions hold:

- There exists an algorithm that given two strings $x, y \in \Sigma^*$ decides whether they belong to the same equivalence class in $(|x| + |y|)^{\mathcal{O}(1)}$ time.
- For any finite set $S \subseteq \Sigma^*$, the equivalence relation $\mathcal{R}$ partitions the elements of S into at most $(\max_{x \in S} |x|)^{\mathcal{O}(1)}$ classes.

We are now ready to give the formal definition of cross-composition.

Definition 1.20. Let $L \subseteq \Sigma^*$ be a language and let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. We say that L *cross-composes* into Q if there exists a polynomial equivalence relation $\mathcal{R}$ and an algorithm $\mathcal{C}$ which, given t strings $x_1, \ldots, x_t$ belonging to the same equivalence class of $\mathcal{R}$, computes an instance $(x^*, k^*) \in \Sigma^* \times \mathbb{N}$ in time polynomial in $\sum_{i=1}^{t} |x_i|$, such that:

- $(x^*, k^*) \in Q$ if and only if $x_i \in L$ for some $1 \leq i \leq t$,
- k^* is bounded by a polynomial in $\max_{i=1}^{t} |x_i| + \log t$.

A parameterized problem Q admits a cross-composition if there exists an NP-hard language L which cross-composes into Q. The algorithm $\mathcal{C}$ is called a cross-composition algorithm.

It is clear from the definition that a cross-composition is a particular kind of polynomial time many-one reduction from $\texttt{OR}(L)$ to Q, where $\texttt{OR}(L)$ is the language that contains all the tuples $(x_1, \ldots, x_t)$ such that at least one of the x_i's is in L. Note that the requirement for a polynomial equivalence relation is only a tool intended to make easier to regroup the input of a cross-composition: indeed, the trivial relation where the only class is the entire Σ^* is a well-defined polynomial equivalence relation.

The notion of cross-composition is closely related to the notion of *weak distillation* [3]:

Definition 1.21. Let $L, L' \subseteq \Sigma^*$ be languages. A *weak* $\texttt{OR}$*-distillation of* L *into* L' (or, in short, a distillation) is an algorithm $\mathcal{D}$ that, given t strings $x_1, \ldots, x_t \in \Sigma^*$, computes a string $y \in \Sigma^*$ in time polynomial in $\sum_{i=1}^{t} |x_i|$, such that:

- $y \in L'$ if and only if $x_i \in L$ for some $1 \leq i \leq t$,
- the length of y is polynomially bounded in $\max_{i=1}^{t} |x_i|$.

[3] In the original definition of distillation $L = L'$: hence the 'weak' in this definition.

A language L is said to admit a distillation when there exists a distillation from L to some language L'.

While the notion of distillation may appear similar to the notion of cross-composition, it is considered less likely for an NP-hard problem to admit a distillation than for an NP-hard parameterized problem to admit a cross-composition: indeed, the existence of cross-compositions was shown for many parameterized problems, while the existence of a distillation is deemed unlikely due to the next theorem. Note that for any language $L \subseteq \Sigma^*$, $\overline{L}$ denotes the language $\Sigma^* \setminus L$. Also, $coNP = \{L \subseteq \Sigma^* : \overline{L} \in NP\}$.

Theorem 1.22. *If there exists a distillation for an NP-hard problem L, then $coNP \subseteq NP/poly$.*

Proof (sketch). By definition, $NP/poly$ is the set of languages which can be decided by a non-deterministic Turing machine with the help of a polynomial advice, where a polynomial advice is a function $f : \mathbb{N} \to \Sigma^*$ such that $|f(n)|$ is bounded by a polynomial in n.

It follows from the definition that the existence of a distillation for L ensures the existence of a distillation for SAT. Therefore, the objective is to show that $\overline{SAT}$ is in $NP/poly$. To do that, the distillation $\mathcal{D}$ of SAT (to a language L) is used as a mapping from $(\overline{SAT}_n)^t$, which is the set of tuples of unsatisfiable formulae of size at most n, to $\overline{L}_{n^c}$, which is the set of strings in $\overline{L}$ of size at most n^c (where c is a constant depending on the distillation algorithm).

By a purely combinatorial argument, it is possible to show that if n and t are big enough (but, at the same time, t is polynomial in n) there exists a subset $S_n \subseteq \overline{L}_{n^c}$ whose size is bounded by a polynomial in n, such that the following two conditions hold:

- if $x \in \overline{SAT}_n$, there exist strings $x_1, \ldots, x_t$ of size at most n such that $x_i = x$ for some i, $1 \leq i \leq t$, and $\mathcal{D}(x_1, \ldots, x_t) \in S_n$,
- if $x \notin \overline{SAT}_n$, for all strings $x_1, \ldots, x_t$ of size at most n and such that $x_i = x$ for some i, $1 \leq i \leq t$, then $\mathcal{D}(x_1, \ldots, x_t) \notin S_n$.

At this point, it is easy to design an algorithm for a non-deterministic Turing machine which runs in polynomial time and, using $f(n) = S_n$ as polynomial advice, decides $\overline{SAT}$. In fact, given an input x, a non-deterministic Turing machine can guess t strings $x_1, \ldots, x_t$ of size at most $|x|$; then, if one of these strings is x, it computes $\mathcal{D}(x_1, \ldots, x_t)$ and accept if and only if $\mathcal{D}(x_1, \ldots, x_t) \in S_n$. □

The reason why $coNP \subseteq NP/poly$ is considered unlikely is that by Yap's theorem [85] it would imply the collapse of the polynomial hierarchy to the third level (the result has since been improved [12]), whereas it is a common conjecture that all the complexity classes in the polynomial time hierarchy are distinct. Recall, incidentally, that the polynomial time hierarchy conjecture is a generalization of well-believed conjectures in Computational Complexity; for instance, if $P = NP$ then the polynomial time hierarchy collapses to its zeroth level, and if $NP = coNP$ then it collapses to its first level.

As mentioned earlier, there exist parameterized problems which admit a cross-composition, but if any of these problems also admits a polynomial kernel, or even just a polynomial bikernel, then we can produce a distillation, as the next theorem shows.

Theorem 1.23. *Let $L \subseteq \Sigma^*$ be a language and let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. If L cross-composes into Q and Q admits a polynomial bikernel into a parameterized problem Q', then there exists a distillation of L into $\mathtt{OR}(\widetilde{Q}')$.*

Proof. Recall that $\mathtt{OR}(\widetilde{Q}')$ is the language that contains tuples $(x_1, \ldots, x_t)$ where at least one of the x_i's is in $\widetilde{Q}'$, the unparameterized version of Q'. Now, let $(x_1, \ldots, x_t)$ be the input of the distillation, for some $t \in \mathbb{N}$, and define $n = \max_{i=1}^{t} |x_i|$. If $t > (|\Sigma| + 1)^n$ then there must be a redundancy in the input, that is $x_i = x_j$ for some $1 \leq i < j \leq t$: assume all multiple copies of a string are removed, then we may assume that $\log t \in \mathcal{O}(n)$).

Using the polynomial equivalence relation $\mathcal{R}$ associated to the cross-composition, it is possible to partition (in polynomial time) the set of strings into r subsets $Y_1, \ldots, Y_r$, such that all the strings in one of the subsets belong to the same equivalence class. In addition, r is bounded by a polynomial in n.

Then, the cross-composition algorithm is applied to each of these sets, producing r instances (z_i, k_i) of Q. Note that the whole computation takes time polynomial in the total input size, and each of the resulting k_i's is bounded by a polynomial in n, as $\log t \in \mathcal{O}(n)$.

At this stage, the kernelization is applied to every instance (z_i, k_i) of Q, which gives r instances (z'_i, k'_i) of Q' such that $|z'_i|$ and k'_i are bounded by a polynomial in k_i and, hence, by a polynomial in n.

Finally, (z'_i, k'_i) is converted to the unparameterized version $\widetilde{z}_i = z'_i\#1^{k'_i}$ and these are all combined together into one tuple $(\widetilde{z}_1, \ldots, \widetilde{z}_r)$.

It is straightforward to verify that $(x_1, \ldots, x_t) \in \mathtt{OR}(L)$ if and only if $(\widetilde{z}_1, \ldots, \widetilde{z}_r) \in \mathtt{OR}(\widetilde{Q}')$. Hence, the first property in the definition of a distillation is satisfied. As for the second one, as $|\widetilde{z}_i|$ for $1 \leq i \leq r$ and r itself are bounded by a polynomial in n, then $|(\widetilde{z}_1, \dots, \widetilde{z}_r)|$ is bounded by a polynomial in n.

In conclusion, the described algorithm is a distillation of L into $\texttt{OR}(\widetilde{Q}')$. $\square$

We have now all the tools needed to prove lower bounds on the kernel size, it is only left to combine the results of Theorem 1.22 and Theorem 1.23:

Corollary 1.24. *Let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem which admits a cross-composition from an NP-hard language $L \subseteq \Sigma^*$. Then Q admits no polynomial bikernel, unless $coNP \subseteq NP/poly$.*

The previous result has probably been the most widely used tool to prove lower bounds for kernel sizes so far, and indeed this will be the only method needed for our purposes. Nonetheless, many refinements appeared later and some are worth mentioning here for completeness.

Dell and van Melkebeek [28] generalized Theorem 1.22, producing the *complementary witness lemma* which made possible to prove lower bounds for kernel sizes of problems which do admit a polynomial kernel. Building on their result, Bodlaender *et al.* [7] introduced the notion of cross-composition *of bounded cost*, in which the parameter k^* of the resulting instance is bounded by $\mathcal{O}(f(t)(\max_{x=1}^{t} |x_i|)^c)$, where c is a constant independent of t. The function $f(t)$ is the *cost* of the cross-composition.

When a parameterized problem Q admits a cross-composition of cost $f(t) = t^{\frac{1}{d}+o(1)}$, $d \in \mathbb{N}$, from an NP-hard language L, then Q admits no bikernel of size bounded by $\mathcal{O}(k^{d-\varepsilon})$ for any $\varepsilon > 0$, unless $coNP \subseteq NP/poly$.

The complementary witness lemma of Dell and van Melkebeek also enables to rule out the existence of polynomial kernels using a *co-nondeterministic* version of the cross-composition [65, 66].

Finally, the notions of cross-composition and distillation can be defined as a particular case of polynomial time many-one reduction from the `AND` of a language instead of the `OR`. It is not difficult to see that a similar version of Theorem 1.23 still holds when replacing `OR` with `AND`, which had already been pointed out by Bodlaender *et al.* [9]. Unfortunately, no equivalent of Theorem 1.22 was known until recently.

Eventually, Drucker managed to prove kernel lower bounds for problems that admit an `AND`-cross-composition under the same theoretical complexity assumption, namely $coNP \nsubseteq NP/poly$ [31]. Additionally, he also strengthened previous results in multiple ways, producing a theoretical setting able to rule out "high-quality probabilistic or quantum polynomial time compression”, where ‘high-quality’ refers to the relation between reliability and compression amount (allowing for trade-off) under theoretical assumptions which are even stronger than the $coNP \nsubseteq NP/poly$ hypothesis.

Chapter 2

Notation and Problem presentation

A *multiset* is a set which can contain multiple copies of the same element. The set of non-negative integers is denoted by $\mathbb{N}$ and the set of positive integers is denoted by $\mathbb{N}^+$. The set $\{1, \ldots, n\}$ is denoted by $[n]$. The set of positive reals is denoted by $\mathbb{R}^+$. All logarithms are to base 2.

2.1 Graphs

In this thesis we will consider problems that stem from Graph Theory and are either graph or hypergraph problems. For the notation, we generally follow Diestel's *Graph Theory* [29].

A *graph* G is an ordered pair $(V(G), E(G))$, where $V(G)$ is the set of *vertices* and $E(G)$ is the set of *edges* (disjoint from $V(G)$), together with an *incidence function* ψ_G from the set of edges to the set of unordered pairs of vertices. If $\psi_G(e) = \{v, w\}$ for $e \in E(G)$ and $v, w \in V(G)$, v and w are called *endvertices* of e and e is an edge *between* v and w, or *joins* v and w. Two vertices are *adjacent* if there is an edge between them. For simplicity, we write e instead of $\psi_G(e)$: for instance, we write $e = vw$ and $e \cap \{v\}$. Also, when it is clear from the context we will write V and E instead of $V(G)$ and $E(G)$.

A graph is *finite* if V and E are finite sets. In this thesis we will only consider finite graphs, which will be simply called *graphs*. Unless otherwise specified, the vertex set and edge set of a graph will always be $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$ respectively, with

$n, m \in \mathbb{N}$. Moreover, n and m will always be used to denote the size of the vertex set and of the edge set of a graph G. The *order* of G is the number of its vertices and the *size* is the number of its edges.

Two graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic* if there exists a bijection $\phi : V \to V'$ such that $uv \in E$ if and only if $\phi(u)\phi(v) \in E'$ for all $u, v \in V$.

The *degree of a vertex* $v \in V$ is the number of edges that have v as endvertex. This defines a function $d_G : V \to \mathbb{N}$, called the *degree function.* The *average degree* $d(G)$ of a graph is the average on the degrees of its vertices, *i.e.*, $\frac{1}{n}\sum_{v \in V} d_G(v)$. If the graph contains no loops, $d(G) = \frac{2|E|}{|V|}$. We say that v is an *odd-degree vertex* if the degree of v is odd and is an *even-degree vertex* if it is even.

For $U \subseteq V$, the *neighbourhood* of U is the set of vertices $N_G(U) = \{v \in V \setminus U : \exists u \in U \textit{ such that } uv \in E\}$; a vertex $v \in N_G(U)$ is a *neighbour* for U. The *closed neighbourhood* of U is the set $N_G[U] = N_G(U) \cup U$. It is also possible to define a series $N_j[U]$ of increasingly large neighbourhoods of U, where $N_1[U] = N_G(U)$ and $N_j[U] = N_1[N_{j-1}[U]]$, $j \in \mathbb{N}$.

For disjoint subsets of vertices $U, W \subseteq V$, the set of edges with one endvertex in U and the other in W is denoted by $E(U, W)$.

A set of edges $F \subseteq E$ is a *matching* if they do not share any endvertex. A *maximal* matching is a matching F such that every edge in $E \setminus F$ shares an endvertex with an edge in F. A *maximum* matching is a matching of maximum size. A *perfect* matching is a matching of size $|V|/2$ (*i.e.*, every vertex is an endvertex of an edge in the matching).

A graph is *simple* if $E \subseteq \{v_i v_j : i, j \in [n] \textit{ and } i < j\}$. In a simple graph there cannot be a *loop*, which is an edge whose endvertices correspond, nor *parallel edges*, which are edges who have the same endvertices. A *multigraph* is a graph where parallel edges, but no loops, are admitted.

A graph is *oriented* if each edge $e = v_i v_j$ has one of two possible *directions* $\{>, <\}$ (where '$>$' means it is oriented from v_i to v_j and '$<$' from v_j to v_i). It is *labelled* if each edge has an associated label $l \in L$, where L is a finite set. For any labelled and/or oriented graph G, $\mathcal{U}(G)$ denotes the underlying unoriented and unlabelled graph.

A *weighted* graph is a graph together with a weight function $w_G : E(G) \to \mathbb{R}^+$. For any set $F \subseteq E$ of edges, $w_G(F) = \sum_{e \in F} w_G(e)$. The weight of G is equal to $w_G(E)$.

2.1.1 Subgraphs and supergraphs

A *subgraph* H of a graph G is a graph for which $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$ and ψ_H is the restriction of ψ_G to $E(H)$. In addition, H inherits the orientations, labels and weights of the edges of G. A *supergraph* of G is a graph which G is a subgraph of.

For $U \subseteq V$, the *subgraph* $G[U]$ *of* G *induced by* U is the subgraph of G that has U has vertex set and contains all and only the edges between vertices in U. On the other hand, *deleting* U from G produces the graph $G - U$, which is defined as the graph $G[V \setminus U]$. For $F \subseteq E$, $G[F]$ is the *subgraph of* G *induced by* F, that has vertex set $V(G[F]) = \{v \in V : \exists e \in E \textit{ such that } v \in e\}$ and edge set $E(G[F]) = F$. Finally, *deleting* F from G produces the graph $G \setminus F$, which has V as vertex set and $E \setminus F$ as edge set. Note that every subgraph of G may be obtained deleting a set of vertices and a set of edges.

For $F \subseteq E$, *duplicating* the edges in F produces a supergraph of G which contains one additional copy of every edge in F.

2.1.2 Paths and cycles

Let $r \in \mathbb{N}$. A *walk* in a graph $G = (V, E)$ is a sequence $v_1e_1v_2 \dots v_re_rv_{r+1}$ where v_i and v_{i+1} are the endvertices of e_i for $i \in [r]$: v_1 is the *initial vertex*, v_{r+1} is the *terminal vertex* and $v_2, \dots, v_r$ are the *internal vertices*. The walk is *from* v *to* w if v is the initial vertex and w is the terminal one. A walk is *closed* if $v_1 = v_{r+1}$. The *length* of the walk is the number of edges it contains. A *trail* is a walk $v_1e_1v_2 \dots v_re_rv_{r+1}$ for which the edges e_i, $i \in [r]$, are all distinct. A trail is closed if $v_1 = v_{r+1}$. Two walks or two trails $v_1e_1 \dots e_rv_{r+1}, v'_1e'_1 \dots e'_rv'_{r+1}$ are *edge-disjoint* if $e_i \neq e'_j$ for every $i, j \in [r]$. *Deleting* a walk from a graph means deleting the set of its edges.

A *path* is a graph with vertex set $v_1, \dots, v_{r+1}$ and edge set $v_1v_2, v_2v_3, \dots, v_rv_{r+1}$. Let $r \geq 2$. A *cycle* is a graph with vertex set $v_1, \dots, v_r$ and edge set $v_1v_2, v_2v_3, \dots, v_{r-1}v_r$. A path which contains r edges is an r*-path*, and a cycle which contains r edges is an r*-cycle*. An r-path is an *odd path* if r is odd and an *even path* otherwise; similarly, an r-cycle is an *odd cycle* if r is odd and an *even cycle* otherwise.

A graph G contains a path or a cycle if it contains them as subgraphs. Note that G contains an r-path if and only if it contains a walk $v_1e_1v_2 \dots v_re_rv_{r+1}$ for which the vertices v_i, $i \in [r+1]$, are all distinct. Similarly, it contains an r-cycle if and only if it contains a closed walk $v_1e_1v_2 \dots v_re_rv_1$ for which the vertices v_i, $i \in [r]$, are all distinct.

When there is no possibility of confusion (for instance, if the graph is simple), walks, trails, paths and cycles are denoted by a sequence of vertices and the edges are omitted, *i.e.*, a path $v_1 e_1 v_2 \dots v_r e_r v_{r+1}$ is denoted by $v_1 v_2 \dots v_r v_{r+1}$.

2.1.3 Connectedness and blocks of a graph

A graph $G = (V, E)$ is *connected* if there is a walk from v to w for every pair $v, w \in V$ of vertices with $v \neq w$. A *connected component* of G, or simply a *component*, is a connected subgraph $G[U]$ for some $U \subseteq V$, such that $G[U \cup \{v\}]$ is disconnected for every $v \in V \setminus U$. We denote by $\mathfrak{C}(G)$ the set of connected components of G. A vertex $v \in V$ is a *cutvertex* if $|\mathfrak{C}(G - \{v\})| > |\mathfrak{C}(G)|$: in particular, if G is connected, a cutvertex is a vertex whose deletion disconnects the graph. Similarly, an edge $e \in E$ is a *bridge* if $|\mathfrak{C}(G \setminus \{e\})| > |\mathfrak{C}(G)|$.

A graph G is *l-connected*, $l \geq 2$, if it contains at least $l+1$ vertices and $G - U$ is connected for every $U \subseteq V$ with $|U| \leq l - 1$.

A *block* of G is a connected subgraph $G[U]$, for some $U \subseteq V$, which does not contain a cutvertex and such that $G[U \cup W]$ is disconnected or contains a cutvertex for every $W \subseteq V \setminus U$. Note that if G is 2-connected then the only block of G is G itself. Different blocks of G overlap in at most one vertex, which is a cutvertex of G; the *interior of a block* is the set of vertices which are contained in that block only, and an *interior vertex* is a vertex contained in the interior of a block. Every edge lies in a unique block and the same holds for cycles. The block decomposition $\mathfrak{B}(G)$ of G is the set of its blocks and it can be computed in $\mathcal{O}(|V| + |E|)$ time.

The *block graph* of G is a tree that contains a vertex for every cutvertex of G and a vertex for every block of G, and an edge between them only if the cutvertex is contained in the block. A block is a *pendant block* if it corresponds to a leaf in the block graph. The *root* of a pendant block is the only cutvertex it contains. Note that the interior of a pendant block is never empty.

2.1.4 Some classes of graphs

Let $G = (V, E)$ be a simple graph. We say that G is a *complete* graph if $uv \in E$ for every $u, v \in V$. The complete graph on s vertices ($s \geq 1$) is usually denoted K_s. A set $U \subseteq V$ of vertices is a *clique* in G if $G[U]$ is a complete graph. A clique is an *odd clique* if it contains an odd number of vertices, otherwise it is an *even clique*.

We say that G is a *forest* if it does not contain any cycles. We say that G is a *tree* if it is connected and it does not contain any cycles. A vertex in a forest is a *leaf* if its degree is one. A tree T is a *spanning tree* for a graph G if $V(T) = V(G)$.

We say that G is a *bipartite* graph if there exists a partition V_0, V_1 of V such that $E = E(V_0, V_1)$. This condition is equivalent to asking that G does not contain odd cycles.

A graph is *chordal* if every cycle has a *chord*, that is, an edge between two vertices which are not adjacent in the cycle.

2.1.5 Hypergraphs

A hypergraph is an ordered pair $H = (V(G), \mathcal{E}(G))$, where $V(G)$ is the set of *vertices* and $\mathcal{E}(G)$ is a family of nonempty subsets of V, whose elements are called *hyperedges*, or simply *edges*. When $\mathcal{E}(G)$ only contains subsets of V with exactly two elements then H is a simple graph, hence the concept of hypergraph is a generalization of the concept of graph. As for the graphs, n will be always used to denote the size of the vertex set, and m to denote the size of the edge set. In addition, when it is clear from the context we will write V and $\mathcal{E}$ instead of $V(G)$ and $\mathcal{E}(G)$.

The degree of a vertex v is the cardinality of $\{e \in \mathcal{E} : v \in e\}$. For $U \subseteq V$, the *neighbourhood* of U is the set of vertices $N(U) = \{v \in V \setminus U : \exists e \in \mathcal{E}, \exists u \in U \textit{ such that } \{v, u\} \subseteq e\}$, while the *closed j-neighbourhood* of U, $j \in \mathbb{N}$, is recursively defined as $N_1[U] = N(U) \cup U$ and $N_j[U] = N_1[N_{j-1}[U]]$. A similar notion is available for edges: given $\mathcal{F} \subseteq \mathcal{E}$, the *neighbourhood* of $\mathcal{F}$ is the set of edges $N(\mathcal{F}) = \{e \in \mathcal{E} \setminus \mathcal{F} : \exists f \in \mathcal{F}, f \cap e \neq \emptyset\}$; moreover, the *closed j-neighbourhood* of $\mathcal{F}$, $j \in \mathbb{N}$, is recursively defined as $N_1[\mathcal{F}] = N(\mathcal{F}) \cup \mathcal{F}$ and $N_j[\mathcal{F}] = N_1[N_{j-1}[\mathcal{F}]]$. By $V(N_j[\mathcal{F}])$ we denote the set of vertices contained in the edges in $N_j[\mathcal{F}]$.

A *subhypergraph* I of a hypergraph H is a hypergraph for which $V(I) \subseteq V(H)$ and $\mathcal{E}(I) \subseteq \mathcal{E}(H)$. For $U \subseteq V$, $H - U$ is the subhypergraph of H obtained *deleting* U; it has $V \setminus U$ as vertex set and $\{e \in \mathcal{E} : e \cap U = \emptyset\}$ as edge set. For $\mathcal{F} \subseteq \mathcal{E}$, $H \setminus \mathcal{F}$ is the subhypergraph of H obtained *deleting* $\mathcal{F}$; it has V as vertex set and $\mathcal{E} \setminus \mathcal{F}$ as edge set.

2.2 Structure of the thesis

In the next chapters, we study some parameterized problems, focusing the attention on whether they admit polynomial kernels. In Chapter 3 we study the k-Chinese Postman problem, which, given a connected weighted simple graph $G = (V, E)$ and integers k and p (k being the parameter), asks for k closed nonempty walks which contain every edge of the graph and whose total weight is at most p. For combinatorial optimisation problems such as this one, the choice of a meaningful parameter can be more difficult, as often obvious parameters as the solution size are not interesting. The parameterization we consider was suggested by van Bevern *et al.* [84] and Sorge [81]. We show that the problem is FPT producing a kernel with $\mathcal{O}(k^2 \log k)$ vertices and $\mathcal{O}(k^2 \log k)$ edges.

In Chapter 4 we study the Test Cover problem: given an hypergraph $H = (V, \mathcal{E})$ and an integer p, decide whether there exists a set $\mathcal{T} \subseteq \mathcal{E}$ with at most p edges such that for every $v, w \in V$ there exists $e \in \mathcal{T}$ with $|e \cap \{v, w\}| = 1$. The problem arises in fault analysis, medical diagnostics, pattern recognition, and biological identification [50, 49, 73]. We study the problem using different parameterizations, showing that it is generally difficult to solve (either $W[1]$-hard, or likely not to admit a polynomial kernel). Then, in Section 4.2, we restrict the problem to hypergraphs with edges of bounded size (at most r vertices in each edge). For this special case, we are able to show that the problem admits a polynomial kernel for three out of the four parameterizations we consider. The results are summarised in the next table, where the first row contains the size of the solution for the given parameterization. Note that $n = |V|$, $m = |\mathcal{E}|$, k is the parameter, $\alpha(n) = \lceil \log n \rceil$ in the unbounded case and $\alpha(n) = \lceil \frac{2(n-1)}{r+1} \rceil$ in the restricted one. Also, 'no poly' means that the problem does not admit a polynomial kernel unless $coNP \subseteq NP/poly$, while $\mathcal{O}(\cdot)$ means that the problem admits a kernel with at most $\mathcal{O}(\cdot)$ vertices.

	k	$\alpha(n) + k$	$n - k$	$m - k$
Unbounded	No poly	$W[2]$-hard [73, 26]	No poly	$W[1]$-complete [20]
Bounded (r)	$\mathcal{O}(rk)$	*para-NP*-complete	$\mathcal{O}(rk^3)$	$\mathcal{O}(k^6 r^{16})$

In Chapter 5, we study a class of parameterized problems called $WAPT(\Pi)$, where Π denotes a graph property: given a connected weighted graph G (possibly labelled and/or oriented) and an integer k, decide whether there exists a subgraph H of G which has the property Π and whose weight is at least $\mathsf{pt}(G) + k$ (where $\mathsf{pt}(G)$ is a constant which depends on the property Π and on G). The graph properties which $WAPT(\Pi)$ is defined for are known in the literature as λ-extendible properties, where $0 < \lambda < 1$ is a real number which depends on Π. Well-known examples of λ-extendible properties are 'being bipartite', 'being balanced', 'being acyclic' and 'having a homomorphism into a vertex-transitive graph' [78]. In particular, when the property is 'being bipartite', $WAPT(\Pi)$ corresponds to Weighted Max Cut, which is one of the most central problems in Computational Complexity [62], and has received attention in Parameterized Complexity since its early days [70].

In Section 5.1 we show that if the weights are integral $WAPT(\Pi)$ can be reduced in polynomial time to an easier problem, for which the input graph has a specific structure.

Using this fact, we prove that Weighted Max Cut is FPT when we restrict the problem to graphs with integral weights. Then, in Section 5.2, we show that for $APT(\Pi)$ (the version of $WAPT(\Pi)$ on unweighted graphs) we can obtain better. More precisely, we show that the problem admits a kernel with at most $\mathcal{O}(k^3)$ vertices if one of the following holds: (i) $\lambda \neq \frac{1}{2}$, (ii) all orientations and labels (if applicable) of the graph K_3 have the property Π, or (iii) Π is a hereditary property of simple or oriented graphs. Finally, in Section 5.3 we show that $APT(\Pi)$ admits a kernel with $\mathcal{O}(k^3)$ vertices when Π is the property of 'being balanced', which is needed to prove the results of Section 5.2, but is also an interesting result in itself.

Lastly, Appendix A contains the definitions of all the problems we mention in the thesis.

2.3 Bibliographic Notes

Most of the results of this thesis are based on some published and unpublished papers.

More specifically, the results of Chapter 3 are based on:

[47] Parameterized complexity of k-Chinese Postman Problem.
G. Gutin, G. Muciaccia, and A. Yeo.
Theor. Comput. Sci. 513: 124-128 (2013).

The results of Chapter 4 are based on:

[46] (Non-)existence of polynomial kernels for the Test Cover problem.
G. Gutin, G. Muciaccia, and A. Yeo.
Inf. Process. Lett. 113(4): 123-126 (2013).

[22] Parameterizations of test cover with bounded test sizes.
R. Crowston, G. Gutin, M. Jones, G. Muciaccia, and A. Yeo.
CoRR, abs/1209.6528, (2012).

The results of Chapter 5 are based on discussions with Robert Crowston, Mark Jones, Geevarghese Philip, Ashutosh Rai and Saket Saurabh, and on the following papers:

[24] Polynomial Kernels for λ-extendible Properties Parameterized Above the Poljak-Turzík Bound.
R. Crowston, M. Jones, G. Muciaccia, G. Philip, A. Rai, and S. Saurabh.
FSTTCS, 43-54 (2013).

[21] Maximum balanced subgraph problem parameterized above lower bound.
R. Crowston, G. Gutin, M. Jones, and G. Muciaccia.
Theor. Comput. Sci. 513: 53-64 (2013).

Chapter 3

k-Chinese Postman

In this chapter we study a generalization of the Chinese Postman problem, which is a well-studied problem in combinatorial optimisation.

Chinese Postman (CP)	
Input:	A pair (G, p) where $G = (V, E)$ is a connected weighted simple graph and p is an integer.
Question:	Is there a closed walk W on G such that every edge of G is contained in it and the total weight of the edges in the walk is at most p?

The Chinese Postman problem models the difficulty which a postman encounters when planning the shortest route that goes through every street where mail must be delivered and terminates at the starting point. The problem was first studied in 1962 by a Chinese mathematician, Kuan Mei-Ko, hence the name. This problem is related to one of the oldest problems in Graph Theory, namely the problem of finding an Eulerian cycle in a connected graph, that is a closed trail which contains every edge of the graph. It was proved by Euler and Hierholzer [36, 54] that a connected graph admits an Eulerian cycle if and only if all vertices of the graph have even degree, and for this reason a connected graph is called *Eulerian* if its vertices satisfy this condition.

If an Eulerian cycle exists in a graph G, then it is a solution for CP [1] on G, while if a solution of weight at most p exists, then edges can be added to the graph obtaining a supergraph of weight at most p which admits an Eulerian cycle. In fact, let $1 + c_e$ $(c_e \geq 0)$

[1]Note that in the literature this problem is generally denoted by CPP, but to keep it consistent with the notation used in the rest of the thesis we have preferred to rename it.

be the number of times the walk W of weight at most p contains the edge e. Consider the multigraph G_W obtained from G adding c_e distinct copies of e for every $e \in E$: then W is an Eulerian cycle in G_W and the weight of G_W is at most p.

This observation suggests that it is possible to solve CP in two steps: first construct a supergraph of G which admits an Eulerian cycle and then find this cycle, which will be a solution for CP on G, provided the supergraph was constructed in an optimal way (that is, if it is the lightest supergraph of G which admits an Eulerian cycle). This approach was first described by Edmonds and Johnson [32], and can be implemented using an algorithm which runs in polynomial time.

Theorem 3.1. *The* Chinese Postman *problem can be solved in polynomial time.*

Proof. Let (G, p) be an instance of CP. Assume the graph G is not Eulerian, otherwise we can solve the problem finding an Eulerian cycle in polynomial time using, for instance, Hierholzer's algorithm. This means that G contains vertices of odd degree: by the handshaking lemma [36], there is an even number of these vertices.

Let G' be the complete weighted graph which has as vertices the odd degree vertices of G and where the weight of an edge e' that joins two vertices v and w is defined as the minimum weight of a path in G from v to w: such minimum weight path will be referred to as the *path corresponding to the edge* e'. Let $M' \subseteq E(G')$ be a perfect matching of minimum weight in G' and let $\mathcal{M}$ be the set of paths of G corresponding to the edges in M'. Note that the paths in $\mathcal{M}$ are edge-disjoint: in fact, if two paths shared some edges, then deleting these edges would give two new paths that still induce a perfect matching in G', whose weight is less than the weight of M'.

We claim that the graph $G_{\mathcal{M}}$ obtained from G duplicating all the edges contained in paths in $\mathcal{M}$ contains only vertices of even degree, and that there exists no supergraph of G whose weight is less than the weight of $G_{\mathcal{M}}$ satisfying the same property.

For the first part of the claim, note that every odd degree vertex of G is adjacent to exactly one duplicated edge (as M' is a perfect matching) and every even degree vertex is either an internal vertex of some of the paths or is not contained in any of them, and in the former case is adjacent to an even number of duplicated edges as the paths in $\mathcal{M}$ are edge-disjoint. To check the second part of the claim note that any supergraph $\widetilde{G}$ of G where every vertex has even degree must be obtained duplicating edges which form paths between odd degree vertices. This means that there exists a multiset of edges of G' such that $\widetilde{G}$ is obtained from G duplicating the paths corresponding to these edges.

As for the time needed to compute the algorithm, constructing the graph G' and finding a perfect matching of minimum weight in it can be done in polynomial time, as well as tracing back the paths in $\mathcal{M}$ and constructing the graph $G_{\mathcal{M}}$. Finally, finding an Eulerian cycle in $G_{\mathcal{M}}$ can be done in polynomial time using Hierholzer's algorithm. Hence, CP is polynomial time solvable. □

There exist many generalizations of CP, but unfortunately only few of them can be shown to be polynomial time solvable. In this chapter, we study the following generalization:

k-CHINESE POSTMAN (k-CP)	
Input:	A triplet (G, p, k) where $G = (V, E)$ is a connected weighted simple graph and p and k are integers.
Parameter:	k
Question:	Is there a set $\mathcal{W} = \{W_1, \ldots, W_k\}$ of k closed nonempty walks such that every edge of G is contained in at least one of them and the total weight of the edges in the walks is at most p?

This problem was proved to be NP-complete by Thomassen [83]. An easy way to be convinced of its hardness is considering the NP-complete problem 3-CYCLE PARTITIONING (see Appendix A, Definition 2). An instance $G = (V, E)$ of this problem can be straightforwardly reduced to k-CP setting the weight of every edge to 1, $k = \frac{|E(G)|}{3}$ and $p = |E(G)|$: in fact, the $\frac{|E(G)|}{3}$-CHINESE POSTMAN problem admits a solution that uses every edge exactly once if and only if the graph can be partitioned into 3-cycles.

We say that a solution, for k-CP or for CP, is *optimal* if there is no solution with smaller weight.

Theorem 3.2. *Let G be a connected weighted simple graph. The weight of an optimal solution W for CP is not greater than the weight of an optimal solution $\mathcal{W} = \{W_1, \ldots, W_k\}$ for k-CP.*

Proof. Given a solution $\mathcal{W} = \{W_1, \ldots, W_k\}$ for k-CP, it is easy to construct a solution W for CP. Firstly order $W_1, \ldots, W_k$ in such a way that W_i shares a vertex with W_j, where $i \in [k]$ and $1 \leq j < i$ (this is possible, as every vertex is contained in one of $W_1, \ldots, W_k$ and the graph is connected). Then construct the walk W inductively as follows: let $W^{(1)} = W_1$ and let $W^{(i)}$ be obtained appending W_i to $W^{(i-1)}$ $(2 \leq i \leq k)$. It follows that $W^{(k)} = W$ is a solution for CP and its weight is exactly the weight of $\mathcal{W}$ as no edges are added. □

Note that the restriction of k-CP that asks for k closed nonempty walks containing a fixed vertex $v \in V$, which may seem equivalent to the more general problem, is actually polynomial time solvable [87, 77], hence the difficulty of solving k-CP is related to the difficulty of finding edge-disjoint cycles in a graph. The next lemma and theorem better clarify this point.

Lemma 3.3. *Let G be a connected weighted simple graph and $\widetilde{G}$ be a multigraph obtained from G duplicating some of its edges. If $\widetilde{G}$ is Eulerian and contains k edge-disjoint cycles, then there exists a solution $\mathcal{W}$ for k-CP on G whose weight is equal to the weight of $\widetilde{G}$. Furthermore, if k edge-disjoint cycles in $\widetilde{G}$ are given, then $\mathcal{W}$ can be constructed in polynomial time.*

Proof. Let $C_1, \ldots, C_k$ be k edge-disjoint cycles in $\widetilde{G}$. Deleting them from $\widetilde{G}$ gives a multigraph where it still holds that every vertex has even degree. Then in every component of this graph it is possible to find an Eulerian cycle which can be appended to a cycle C_i, $i \in [k]$, which it shares at least one vertex with. Call $W_1, \ldots, W_k$ the closed walks which are obtained after appending all the Eulerian cycles. Then $\mathcal{W} = \{W_1, \ldots, W_k\}$ is a solution for k-CP on G whose weight is the same as the sum of the weights of the edges in $\widetilde{G}$.

If the k edge-disjoint cycles are given, $\mathcal{W}$ can be constructed in polynomial time, since Eulerian cycles can be found and appended in polynomial time. □

Theorem 3.4. *Let G be a connected weighted simple graph and let W be an optimal solution for CP on G. If the multigraph G_W contains at least k edge-disjoint cycles, then the weight of an optimal solution for k-CP on G is equal to the weight of W. Furthermore, if k edge-disjoint cycles are given, then a solution for k-CP on G can be constructed in polynomial time.*

Proof. Note that G_W is obtained from G duplicating some edges, is Eulerian and contains k edge-disjoint cycles, hence by Lemma 3.3 there exists a solution $\mathcal{W}$ for k-CP on G whose weight is equal to the sum of the weights of the edges in G_W, *i.e.*, to the weight of W. Hence, by Theorem 3.2, $\mathcal{W}$ is optimal.

If the k edge-disjoint cycles are given, Lemma 3.3 ensures that $\mathcal{W}$ can be constructed in polynomial time. □

Theorem 3.4 shows that k-CP can be efficiently solved when G contains many edge-disjoint cycles, as for any solution W to CP, G_W is a supergraph of G and hence it contains at least as many edge-disjoint cycles. We will use this fact to produce a kernel for k-CP.

Corollary 3.5. *Let G be a connected weighted simple graph. If G contains at least k edge-disjoint cycles, then the weight of an optimal solution for k-CP on G is equal to the weight* *of an optimal solution for CP on G. If k edge-disjoint cycles are given, then a solution for k-CP on G can be constructed in polynomial time.*

When dealing with graph problems where the input graph is simple, it is often easier to prove an upper bound on the number of vertices of the graph: the bound on the size follows from the fact that in a simple graph the number of edges is bounded by the square of the number of vertices.

In this case, we will give a bound on the number of vertices according to their degree. In particular, we partition the set of vertices V of G into three sets V_1, which contains the vertices of degree one, V_2, which contains the vertices of degree two, and $V_{\geq 3}$, which contains the vertices of degree greater or equal to three. In fact, it is not difficult to show a bound on the number of vertices of low degree (as is often the case), while the proof of the bound on $|V_{\geq 3}|$ will be more involved.

From now on we will assume that (G, p, k) is an instance for the k-Chinese Postman problem and $k \geq 2$.

Lemma 3.6. *If $|V_1| \geq k$ then an optimal solution for k-CP on G can be found in polynomial time.*

Proof. Let $V_1 = \{v_1, \dots, v_r\}$, $r \geq k$, and let w_i be the neighbour of v_i, $1 \leq i \leq r$ (note that it may be $w_i = w_{i'}$ for $i \neq i'$). For any solution W to CP, the multigraph G_W contains at least two copies of the edge $v_i w_i$, as every vertex has even degree in G_W. Hence G_W contains at least $r \geq k$ edge-disjoint 2-cycles, which can obviously be found in polynomial time, and by Theorem 3.4 an optimal solution for k-CP can be found in polynomial time. □

Lemma 3.6 ensures that every time G contains at least k vertices of degree one, the problem can be efficiently solved, hence from now on we may consider only graphs which contain less than k such vertices.

The situation with vertices of degree two is not as easy, G may contain many of them and still there could be no obvious way of solving the problem in polynomial time. Nevertheless, in this case it is possible to apply a reduction rule to reduce the size of the graph. First of all, we introduce a new definition.

Definition 3.7. Let $u \in V_2$ and let v and w be its neighbours. The operation of bypassing u consists of deleting u and adding an edge vw whose weight is the sum of the weights of uv and uw. Note that this may create parallel edges.

We can now state the reduction rule.

Reduction Rule 3.8. *Let $P = v_0 v_1 \dots v_r v_{r+1}$ be a path in G whose internal vertices have degree two and such that $r > k$. Pick a vertex v_i, $1 \leq i \leq r$, bypass it and let G' be the resulting graph. Choose v_i in such a way that the minimum weight of an edge in G and G' is the same.*

The last condition in the reduction rule is easily applied: we can choose v_i to be v_1; if by this choice the condition is not met, then we can pick v_3 instead (recall that $k \geq 2$). This condition is motivated by the fact that we do not want to delete an edge of minimum weight in the graph if there is only one. Before explaining the reason, it is necessary to define the multigraph $G_{\mathcal{W}}$ associated to every solution $\mathcal{W}$ for k-CP, as it was done with W and G_W for CP.

Definition 3.9. Let $G = (V, E)$ be a connected weighted simple graph and let $\mathcal{W} = \{W_1, \dots, W_k\}$ be a solution to k-CP on G. Let t_e^i be the number of times the walk W_i contains an edge e. The multigraph $G_{\mathcal{W}}$ has V as vertex set and contains $\sum_{i \in [k]} t_e^i$ copies of every edge $e \in E$.

We say that e is used $\sum_{i \in [k]} t_e^i$ times by $\mathcal{W}$.

We already know because of Lemma 3.3 that there exist solutions for k-CP on G corresponding to every supergraph of G which is obtained duplicating some edges, is Eulerian and contains at least k edge-disjoint cycles. Now Definition 3.9 shows that for every solution we can construct such a supergraph, hence these notions correspond to some extent. For this reason, we will sometimes call this supergraph a 'solution' to k-CP on G.

It is useful at this point to study its structure.

Lemma 3.10. *Let uv be an edge of minimum weight in G. There exists an optimal solution $\mathcal{W}$ for k-CP on G such that every edge in $E \setminus \{uv\}$ is used at most twice.*

Proof. Let $G_{\mathcal{W}'}$ be an optimal solution for k-CP on G and assume there exists an edge $u'v' \neq uv$ which is used at least three times. Construct a new solution $G_{\mathcal{W}''}$ deleting two copies of the edge $u'v'$ and adding two copies of the edge uv. The weight of $G_{\mathcal{W}''}$ is equal to the weight of $G_{\mathcal{W}'}$ plus $2(w_G(uv) - w_G(u'v'))$, which is a nonpositive quantity as uv is an edge of minimum weight.

It is only left to show that $G_{\mathcal{W}''}$ is Eulerian and contains k edge-disjoint cycles. The first requirement is obviously satisfied. As for the second one, note that deleting two copies of an edge can reduce the number of cycles by at most one. Indeed, at most two edge-disjoint cycles of $G_{\mathcal{W}'}$ are affected by the deletion of two copies of $u'v'$: let these cycles be C_1 and C_2, then $C_1 \setminus \{u'v'\}$ and $C_2 \setminus \{u'v'\}$ together form a cycle in $G_{\mathcal{W}''}$. At the same time, adding two copies of uv add a cycle that is edge-disjoint from all the others, which means that in $G_{\mathcal{W}''}$ there are at least k edge-disjoint cycles.

Iterating this construction we can produce a solution $\mathcal{W}$ which satisfies the requirements. □

Note that Lemma 3.10 is a generalization of the fact that the multigraph G_W, where W is a solution to CP on G, contains at most two copies of every edge $e \in E$ [18]. Using this lemma we can now prove that Reduction Rule 3.8 is valid.

Lemma 3.11. *Reduction Rule 3.8 is valid.*

Proof. First of all, note that G' is still a simple graph, as there is no edge between v_{i-1} and v_{i+1}. By Lemma 3.10 there exists an optimal solution $G_{\mathcal{W}}$ to k-CP on G which uses every edge in P either once or twice, except for an edge uw of minimum weight (different from v_iv_{i-1} and v_iv_{i+1}) which may be used more than twice. The same is true for G' and the path P' (the one obtained from P bypassing v_i).

Hence, a solution $G_{\mathcal{W}}$ which uses $P \setminus \{uw\}$ only once can be transformed into a solution $G_{\mathcal{W}'}$ that uses $P' \setminus \{uw\}$ only once and uses every other edge the same number of times. Similarly, a solution $G_{\mathcal{W}}$ which uses $P \setminus \{uw\}$ twice can be transformed into a solution $G_{\mathcal{W}'}$ that uses $P' \setminus \{uw\}$ twice. The only care which must be observed is that in this case $G_{\mathcal{W}'}$ may contain less cycles than $G_{\mathcal{W}}$, but as P' contains at least k edges, it holds that $G_{\mathcal{W}'}$ contains at least k edge-disjoint cycles and therefore is a solution for k-CP on G'. In both cases the weight of $G_{\mathcal{W}}$ and $G_{\mathcal{W}'}$ is the same.

The previous reasoning can be also applied to transform a solution $G_{\mathcal{W}'}$ for k-CP on G' into a solution $G_{\mathcal{W}}$ for k-CP on G, which ensures that (G, p, k) is a `YES`-instance if and only if (G', p, k) is.

The fact that the rule can be applied in polynomial time follows from the fact that the set V_2 can be constructed in polynomial time, the paths with internal vertices in V_2 can be found in polynomial time and the operation of bypassing can be performed in polynomial time too. □

Applying Reduction Rule 3.8 does not directly give a bound on $|V_2|$, but it can be used to this purpose once we have obtained a bound on $|V_1|$ and $|V_{\geq 3}|$. To show a bound on $|V_{\geq 3}|$, we will use an approach based on the fact that a graph which contains many edges must contain many edge-disjoint cycles. The next lemma was used by Bodlaender *et al.* to show the existence of a kernel of polynomial size for the Disjoint Cycle Packing problem (see Appendix A, Definition 13) and it proves useful for k-CP too.

Lemma 3.12 ([10]). *There exists a constant c_v such that every graph H with minimum degree at least 3 which contains at least $c_v k \log k$ vertices contains k edge-disjoint cycles. Such k cycles can be found in polynomial time.*

In the original proof of Lemma 3.12 it is not argued that the k cycles can be found in polynomial time. Nevertheless, it is easy to infer it, as the cycles are found using a greedy approach that repeatedly looks for the shortest cycle in H and deletes it, which can be done in polynomial time [58], and stops after k times. Also, Lemma 3.12 was proved only for simple graphs, but it is easy to generalize it to multigraphs, as parallel edges form a 2-cycle, so the greedy algorithm can be designed to delete them until it is left with a simple graph, or it has deleted k pairs.

Lemma 3.13. *If $|V_{\geq 3}| \geq c_v k \log k + k$, where c_v is the constant given by Lemma 3.12, then an optimal solution for k-CP on G has the same weight of an optimal solution for CP on G and it can be found in polynomial time.*

Proof. Let G' be obtained from G by deleting all vertices of degree one and bypassing all vertices of degree two. As we assumed that $|V_1| \leq k$ and because of the lower bound on $|V_{\geq 3}|$ it holds that G' contains at least $c_v k \log k$ vertices and the minimum degree of a vertex is three. Then, by Lemma 3.12, G' contains at least k edge-disjoint cycles which can be found in polynomial time. The same holds for G, as to each of the cycles in G' corresponds a cycle in G. Therefore, by Corollary 3.5, the weight of an optimal solution for k-CP on G is equal to the weight of an optimal solution for CP on G and an optimal solution can be found in polynomial time. □

Combining the results of Lemma 3.6 and 3.13 we can solve k-CP in polynomial time when $|V_1 \cup V_{\geq 3}| \geq 2k + c_v k \log k$, therefore from now on we may assume that $|V_1 \cup V_{\geq 3}| \in \mathcal{O}(k \log k)$. To show how this implies a bound on $|V_2|$ we make use of an auxiliary multigraph G_{-2}: let $V(G_{-2}) = V_1 \cup V_{\geq 3}$ and add an edge between vertices v and w for every path from v to w in G whose internal vertices are in V_2 (note that an edge from v to w counts as a path with no internal vertices, which means that we also add an edge for every edge in G).

Now, if there are at least $2k$ edges in G_{-2} between vertices v and w then there are at least k edge-disjoint cycles in G that can be found in polynomial time. When this does not happen, G_{-2} contains at most $\mathcal{O}(k \cdot k^2 \log^2 k)$ edges, which in turn ensures that $|V_2|$ is in $\mathcal{O}(k^4 \log^2 k)$ if G is reduced under Reduction Rule 3.8. However, it is possible to obtain a better bound using a modified version of Lemma 3.12.

Lemma 3.14. *Let c be any constant. There exists a constant c_e such that every multigraph H with at most $ck \log k$ vertices and at least $c_e k \log k$ edges contains k edge-disjoint cycles. Such k cycles can be found in polynomial time.*

Proof. Alon *et al.* showed that a graph with average degree d and n vertices contains a cycle of length at most $2(\log_{d-1} n + 2)$ [3]. The result easily applies to multigraphs too, as two parallel edges form a 2-cycle. Hence, to find k edge-disjoint cycles in H we can repeatedly find a shortest cycle and delete its edges until we have done it k times (recall that a shortest cycle can be found in polynomial time [58]). We want to define c_e large enough to ensure that even after deleting k cycles in this way the average degree is still at least 3: if this holds, we know that each of the deleted cycles has length at most $2(\log(|V(H)|) + 2) \leq 2(\log c + 2\log k + 2)$. In other words, we want the following inequality to be satisfied:

$$2(c_e k \log k - k(2(\log c + 2\log k + 2))) \geq 3ck \log k$$

This holds if $c_e \geq \frac{3}{2}c + \frac{2(\log c + 2\log k + 2)}{\log k}$, which is true if $c_e \geq \frac{3}{2}c + 2\log c + 6$. □

Now we can finally show the main result of this section.

Theorem 3.15. *The k-Chinese Postman problem admits a kernel with $\mathcal{O}(k^2 \log k)$ vertices and $\mathcal{O}(k^2 \log k)$ edges.*

Proof. Let (G, p, k) be an instance of k-CP. If G contains at least k vertices of degree 1 or $c_v k \log k$ vertices of degree at least 3 then by Lemma 3.6 and 3.13 we conclude that the answer is `YES`. Otherwise, exhaustively apply Reduction Rule 3.8 to the graph: since every time the number of edges decreases, we will obtain a reduced graph after at most $\mathcal{O}(|E|)$ applications. Applying Lemma 3.14 to the auxiliary graph G_{-2} we can see that the number of paths whose internal vertices have degree 2 is bounded by $\mathcal{O}(k \log k)$, or the instance is a `YES`-instance. Since the graph is reduced by Reduction Rule 3.8, each of these paths contains at most k vertices, hence the vertices of degree 2 are at most $\mathcal{O}(k^2 \log k)$. This shows that k-CP admits a kernel with $\mathcal{O}(k^2 \log k)$ vertices.

As for the bound on the number of edges, observe that Lemma 3.14 applied to G_{-2} also ensures that there can be at most $\mathcal{O}(k \log k)$ edges between vertices in $V_1 \cup V_{\geq 3}$. All the other edges have an endvertex in V_2, but for each vertex in V_2 there are exactly two edges having it as endvertex, so there are at most $\mathcal{O}(k^2 \log k)$ edges of this type. $\square$

Chapter 4

Test Cover

The TEST COVER problem is a well-known optimisation problem on hypergraphs. The underlying idea is that we want to distinguish between different objects using as few tests as possible, where a test can only give a positive or a negative answer for any object which is tested. One of the original motivations for the problem comes from a request from the Agricultural University in Wageningen about the identification of potato diseases [27]. Each variety of potatoes is vulnerable to a number of diseases and one seeks to minimize the set of different varieties needed to uniquely identify every disease.

TESTCOVER(p)

Input: A pair (H, p) where $H = (V, \mathcal{E})$ is a hypergraph and p is an integer.

Question: Is there a subset $\mathcal{T} \subseteq \mathcal{E}$ with $|\mathcal{T}| \leq p$ such that for every $v, w \in V$ there exists $e \in \mathcal{T}$ with $|e \cap \{v, w\}| = 1$?

From now on, let $n = |V|$ and $m = |\mathcal{E}|$. A set $\mathcal{T}$ which satisfies the property we look for is called a *test cover*. An edge $e \in \mathcal{E}$ *separates* a pair $v, w \in V$ if $|e \cap \{v, w\}| = 1$. A set of edges $\mathcal{E}' \subseteq \mathcal{E}$ separates v and w if there exists $e \in \mathcal{E}'$ which separates them. Two disjoint sets of vertices $V_1, V_2 \subseteq V$ are separated by $\mathcal{E}'$ if, for every $v_1 \in V_1$ and $v_2 \in V_2$, $\mathcal{E}'$ separates v_1 and v_2.

The objective is to find a subset of edges such that all pairs of vertices are separated by it: when one exists, a solution of minimum size is called an *optimal* solution.

The potato diseases identification involved 28 diseases and 63 varieties, and was solved using a combination of greedy and local improvement algorithms. Commonly, though, instances of the TEST COVER problem are not as small. For instance, one of the other earlier applications of the problem, the protein identification by epitope recognition [49, 26], may involve around 40 000 − 100 000 different proteins when the method is applied to the human organism.

Theorem 4.1. *[41] TestCover(p) is NP-complete.*

Proof (sketch). It is in NP, as we can guess $\mathcal{T}$, if it exists, and then check that it is a test cover in polynomial time. To show that it is NP-hard we can reduce from 3-Dimensional Matching (see Appendix A, Definition 3). Let (V, T) be an instance of the latter problem. Let $V(H) = W \cup X \cup Y \cup \{w_0, x_0\}$, where W, X, Y are three copies of V and w_0, x_0 are new vertices, and let $\mathcal{E}(H) = \{\{w, x, y\} : w \in W, x \in X, y \in Y, (w, x, y) \in T\} \cup \{W \cup \{w_0\}\} \cup \{X \cup \{x_0\}\}$. It follows that (V, T) admits a 3-dimensional matching if and only if H admits a test cover which uses $|V| + 2$ edges. □

The previous proof can be refined using a reduction from the NP-complete problem P_3-Packing (see Appendix A, Definition 6), which leads to the following result:

Theorem 4.2. *[26] TestCover(p) is NP-complete even when the size of every edge is bounded by 2.*

From the point of view of approximation, the situation is not particularly bright. The optimisation version of the Test Cover problem, which asks for an optimal solution, admits a reduction both from and to the optimisation version of Set Cover (see Appendix A, Definition 4). Hence, the greedy algorithm has a performance ratio of $\mathcal{O}(\log n)$ [73], but there is no $o(\log n)$-approximation unless $P = NP$ and no $(1 - \varepsilon) \log n$-approximation for any $\varepsilon > 0$ unless $NP \subseteq DTIME(n^{\log \log n})$ [50].

When we turn to consider the Test Cover problem in the framework of Parameterized Complexity we are faced with the difficulty of the choice of the parameter. In the early 2000s, when the field was young, the usual approach was to consider the size of the solution as the parameter, which is known in fact as the *standard parameterization.* Later the situation changed dramatically and the variety in the parameter choice eventually started to reflect the spirit of the original definition, for which the parameter is any element of Σ^*.

For the Test Cover problem there are many interesting different parameterizations. To make their description easier, consider first the following generic parameterization.

TESTCOVER(p,k)	
Input:	A triplet (H,p,k) where $H=(V,\mathcal{E})$ is a hypergraph and p and k are integers.
Parameter:	k
Question:	Is there a subset $\mathcal{T}\subseteq\mathcal{E}$ with $\lvert\mathcal{T}\rvert\le p$ such that for every $v,w\in V$ there exists $e\in\mathcal{T}$ with $\lvert e\cap\{v,w\}\rvert=1$?

In this chapter we consider four different parameterizations of the TEST COVER problem, namely the ones obtained when $p=k$, $p=n-k$, $p=m-k$ or $p=\lceil\log n\rceil+k$ [1].

Note that if $\mathcal{E}$ is not a test cover, then the answer to the problem is NO for every p. On the other hand, to check whether a set of edges is a test cover takes polynomial time, hence we will always assume that $\mathcal{E}$ is a test cover.

The rest of the definitions and proofs of this section are technical tools which will be useful later.

To any subset $\mathcal{E}'\subseteq\mathcal{E}$ we can associate an equivalence relation on V: two vertices $v,v'\in V$ are in the same class if and only if for any $e\in\mathcal{E}'$ either $v,v'\in e$ or $v,v'\notin e$. The classes induced by this equivalence relation are called the *classes induced by* $\mathcal{E}'$. In other words, these classes contain objects that cannot be distinguished using tests in $\mathcal{E}'$. Note that a test cover is a set of edges which induces n classes.

Let $C\subseteq V$ be a class induced by $\mathcal{E}'$. We say that an edge $e\in\mathcal{E}$ *splits* C if there exist $v,v'\in C$ such that $v\in e$ and $v'\notin e$. Observe that e cannot be in $\mathcal{E}'$, and when adding e to $\mathcal{E}'$ the class C is replaced by two classes $C_1,C_2\subseteq C$ with $C_1\cup C_2=C$.

We say that $\mathcal{E}'\subseteq\mathcal{E}$ *isolates* $U\subseteq V$ if it separates U and $V\setminus U$. An *isolated* vertex is a vertex which is not contained in any edge (note that there may be at most one in an instance, otherwise the answer to the problem is NO).

Lemma 4.3. *For every class $C\subseteq V$ induced by $\mathcal{E}'\subseteq\mathcal{E}$ and for every $e\in\mathcal{E}'$, either $C\subseteq e$ or $C\cap e=\emptyset$. Also, for every pair of classes C_1,C_2 induced by $\mathcal{E}'$, there exists $e'\in\mathcal{E}'$ such that e' contains exactly one of them.*

Proof. If $C\cap e\neq\emptyset\neq C\setminus e$, then e splits C, which therefore cannot be a class induced by $\mathcal{E}'$. Moreover, if for every $e\in\mathcal{E}'$ either $C_1\cup C_2\subseteq e$ or $(C_1\cup C_2)\cap e=\emptyset$, then C_1 and C_2 are not separated by $\mathcal{E}'$, which is a contradiction. □

[1] In section 4.2, $\lceil\log n\rceil+k$ will be replaced by $\lceil\frac{2(n-1)}{r+1}\rceil+k$.

Lemma 4.4. *[22] If $\mathcal{E}$ induces $t \geq 2$ classes in a hypergraph $H = (V, \mathcal{E})$ and $i \in [t-1]$, then there exists a subset $\mathcal{E}'$ of $\mathcal{E}$ with i edges that induces at least $i+1$ classes.*

Proof. By induction on $i \in [t-1]$. To see that the lemma holds for $i = 1$ set $\mathcal{E}' = \{e\}$, where e is any edge of $\mathcal{E}$ with less than $|V|$ vertices. Let $\mathcal{E}'$ be a subset of $\mathcal{E}$ with $i-1$ edges that induces at least i classes, let v, w be vertices separated by $\mathcal{E}$ not separated by $\mathcal{E}'$, and let $e \in \mathcal{E} \setminus \mathcal{E}'$ be an edge separating v and w. It remains to observe that $\mathcal{E}' \cup \{e\}$ induces at least $i+1$ classes. $\square$

Theorem 4.5. *Let $H = (V, \mathcal{E})$ be a hypergraph and let $\mathcal{E}$ be a test cover. Then there exists a test cover $\mathcal{E}' \subseteq \mathcal{E}$ such that $|\mathcal{E}'| \leq n-1$.*

Proof. Since a test cover is a set of edges which induces n classes, Lemma 4.4 immediately implies the result. $\square$

Theorem 4.5 shows an interesting upper bound on the size of an optimal solution, which complements the trivial upper bound provided by the number of edges. It will be used in Section 4.1.3 to show that the difference between the number of vertices and the size of an optimal solution can be used as a parameter.

Definition 4.6. Let $H = (V, \mathcal{E})$ be a hypergraph, $V' \subseteq V$ and $\mathcal{E}' \subseteq \mathcal{E}$. Let $C_1, \ldots, C_l$, $l \in \mathbb{N}$, be the classes induced by $\mathcal{E}'$ which contain vertices from V'. The hypergraph $H_{\mathcal{E}'}[V']$ associated to $\mathcal{E}'$ and V' has vertex set $\{v'_1, \ldots, v'_l\}$ and edge set $\{e' = \{v'_j : C_j \subseteq e\} : e \in \mathcal{E}'\}$.

In other words, $H_{\mathcal{E}'}[V']$ is obtained from H keeping only the vertices in V', but identifying the ones that are in the same classes induced by $\mathcal{E}'$. This construction works because by Lemma 4.3 an edge in $\mathcal{E}'$ either contains or does not intersect a class induced by $\mathcal{E}'$. Note that for every edge in $\mathcal{E}'$ there is an edge corresponding to it in $\mathcal{E}(H_{\mathcal{E}'}[V'])$. In particular, the edges corresponding to $\mathcal{E}'$ form a test cover in $H_{\mathcal{E}'}[V']$.

Lemma 4.7. *Let $H = (V, \mathcal{E})$ be a hypergraph, $V_1, V_2 \subseteq V$ be such that $V_1 \cap V_2 = \emptyset$ and $\mathcal{E}' \subseteq \mathcal{E}$ be a set of edges that separates V_1 and V_2. Let t_1 be the number of classes induced by $\mathcal{E}'$ in V_1, and t_2 be the number of classes induced by $\mathcal{E}'$ in V_2. Then there exists a subset $\mathcal{F}$ of $\mathcal{E}'$ that separates V_1 and V_2 and contains at most $t_1 + t_2 - 1$ edges.*

In particular, it is always possible to separate V_1 and V_2 using at most $|V_1| + |V_2| - 1$ edges.

Proof. Let $H_{\mathcal{E}'}[V_1 \cup V_2]$ be as in Definition 4.6. Note that this hypergraph contains $t_1 + t_2$ vertices and its edge set is a test cover, hence by Theorem 4.5 there exists a subset of at most $t_1 + t_2 - 1$ edges which is a test cover in $H_{\mathcal{E}'}[V_1 \cup V_2]$. The corresponding set of edges in $\mathcal{E}'$ separates every class which has nonempty intersection with V_1 or V_2, and in particular separates V_1 and V_2. $\square$

4.1 Four parameterizations of Test Cover

4.1.1 The standard parameterization

Firstly we consider TestCover(k, k), that is, the parameterization where the parameter is the size of the solution. The following theorem is a well-known property of every test cover [50].

Theorem 4.8. *Any test cover $\mathcal{T}$ contains at least $\lceil \log n \rceil$ edges.*

Proof. We count the maximum number of classes induced by a set of cardinality s. Assume that a set $\mathcal{E}' \subseteq \mathcal{E}$ induces t classes. In order to induce new classes when added to $\mathcal{E}'$, an edge $e \notin \mathcal{E}'$ has to split some of the classes induced by $\mathcal{E}'$: each class which is split is replaced by two classes, which means that adding e can increase the classes by at most t. On the other hand, a set containing only one edge can induce at most 2 classes. Hence it is not difficult to see by induction that a set of cardinality s induces at most 2^s classes.

Since a test cover $\mathcal{T}$ induces n classes, it must hold that $2^{|\mathcal{T}|} \geq n$, which implies $|\mathcal{T}| \geq \lceil \log n \rceil$. $\square$

Since there exists this lower bound on the size of a test cover, the proof of the next theorem is easy.

Theorem 4.9. TestCover(k, k) *is fixed-parameter tractable.*

Proof. If $k < \lceil \log n \rceil$ then answer `NO`. Otherwise, it holds that $n \leq 2^k$, which in turn implies $m \leq 2^{2^k}$. Hence, even a brute force search can be performed in $\mathcal{O}(2^{2^{2^k}})$ time. $\square$

Theorem 4.9 shows in fact that TestCover(k, k) admits a kernel, but the size of the kernel is definitely not practical. One may ask whether it is possible to obtain something better; in particular, one may try to find a polynomial kernel for the problem. Unfortunately, TestCover(k, k) is unlikely to admit one.

Theorem 4.10. TestCover(k, k) *does not admit a polynomial bikernel unless coNP $\subseteq$ NP/poly.*

Proof. We will use Corollary 1.24 with Q being TestCover(k,k) and L being 3-Dimensional Matching. Firstly, we define a polynomial equivalence relation $\mathcal{R}$ on Σ^*: two instances (V_1,T_1) and (V_2,T_2) of 3-Dimensional Matching are equivalent if and only if $|V_1| = |V_2|$, while two strings $x,y \in \Sigma^*$ which are not instances of 3-Dimensional Matching are always equivalent to each other. It is not difficult to verify that $\mathcal{R}$ is a polynomial equivalence relation; in particular, note that given a finite set $S \subseteq \Sigma^*$, its elements belong to at most $\max_{x\in S}|x| + 1$ different classes.

Now, let $(V_1,T_1),\ldots,(V_t,T_t)$ be t instances of 3-Dimensional Matching belonging to the same equivalence class, that is, $|V_1| = |V_2| = \cdots = |V_t| = n$. Without loss of generality we may suppose that $V_i = V_j$ for $i,j \in [t]$. We will now define an instance (H,k) of TestCover(k,k). Let W,X and Y be three copies of V and let $V(H) = W \cup X \cup Y \cup \{w_0,x_0,y_0\} \cup \{z_i^j : 0 \le i \le \lceil \log t\rceil, 0 \le j \le n-1\} \cup \{\widetilde{z}_i : 0 \le i \le \lceil \log t\rceil\}$, where w_0,x_0,y_0, z_i^j and $\widetilde{z}_i$ are new vertices. Call Z the set $\{z_i^j : 0 \le i \le \lceil \log t\rceil, 0 \le j \le n-1\}$: this set can be thought of as a matrix whose purpose is distinguishing between the different instances of 3-Dimensional Matching.

For $l \in [t]$, let r_l be the base-2 representation of l modulo $2^{\lceil \log t\rceil}$ which uses $\lceil \log t\rceil$ digits (*e.g.*, if $t = 28$ and $l = 5$, $r_l = 00101$, while if $t = 32 = l$, $r_l = 00000$). Let r_{li} be the ith digit in r_l for $1 \le i \le \lceil \log t\rceil$. For $l \in [t]$ and $0 \le j \le n-1$, let $E_l^j = \{z_0^j\} \cup \{z_i^{j+r_{li}} : 1 \le i \le \lceil \log t\rceil\}$, where superscripts are taken modulo n. In other words, E_l^j roughly corresponds to the j-th column of Z, where some of the vertices are taken from the $(j+1)$-th column and the choice is made according to the base-2 representation of l.

Note that $\cup_{0\le j\le n-1} E_l^j = Z$ for every $l \in [t]$, but there is no other way of covering Z using n sets of this form. First of all, note that the chosen sets must be disjoint, as their cardinality is $1 + \lceil \log t\rceil$ and the cardinality of Z is $(1 + \lceil \log t\rceil)n$. Now, suppose E_l^j and $E_{l'}^{j'}$ are used, with $l \ne l'$. Then there exists $1 \le i \le \lceil \log t\rceil$ such that $r_{li} \ne r_{l'i}$: without loss of generality, assume $r_{li} = 1$. This means that $z_0^j \in E_l^j$, $z_i^j \notin E_l^j$ and $z_0^j, z_i^j \in E_{l'}^j$, hence the only way to cover z_i^j without covering twice z_0^j is to use E_l^{j-1}. Iterating this reasoning we can show that $E_l^{j'}$ must be used, which is not allowed.

Let $Z_i = \{z_i^j : 0 \le j \le n-1\} \cup \{\widetilde{z}_i\}$ for $0 \le i \le \lceil \log t\rceil$ be the i-th row of Z (with the addition of the vertex $\widetilde{z}_i$). For $l \in [t]$, let $\mathcal{E}_l = \{\{w,x,y\} \cup E_l^j : w \in W, x \in X, y \in Y, (w,x,y) \in T_l,\ \textit{and}\ 0 \le j \le n-1\}$. We are now able to describe the edge set of H: $\mathcal{E}(H) = (\cup_{l\in[t]}\mathcal{E}_l) \cup \{Z_i : 0 \le i \le \lceil \log t\rceil\} \cup \{W \cup \{w_0\}\} \cup \{X \cup \{x_0\}\}$. Finally, let $k = n + \log t + 3$.

If (V_l,T_l) admits a 3-dimensional matching $M_l = \{m_0,\ldots,m_{n-1}\}$ for $l \in [t]$, then $\mathcal{T} =$

$\{m_j \cup E_l^j : 0 \leq j \leq n-1\} \cup \{Z_i : 0 \leq i \leq \lceil \log t \rceil\} \cup \{W \cup \{w_0\}\} \cup \{X \cup \{x_0\}\}$ is a test cover for H containing $n + \log t + 3$ edges. In fact, y_0 is the only vertex not contained in any edge, w_0 and x_0 are only contained in $W \cup \{w_0\}$ and $X \cup \{x_0\}$ respectively, $w \in W$ is the only vertex contained in $m \cap (W \cup \{w_0\})$ for some $m \in M_l$ (and similarly for $x \in X$ and $y \in Y$), z_i^j is the only vertex contained in $Z_i \cap E_l^j$ (or $Z_i \cap E_l^{j-1}$ if $r_{li} = 1$), and $\widetilde{z}_i$ is contained in Z_i only.

On the other hand, if H admits a test cover $\mathcal{T}$ with at most $n + \log t + 3$ edges, $W \cup \{w_0\}$, $X \cup \{x_0\}$ and Z_i, $0 \leq i \leq \lceil \log t \rceil$, must be in $\mathcal{T}$ because they are the only edges containing w_0, x_0 and $\widetilde{z}_i$ respectively (and since y_0 is not contained in any edge, every other vertex must be). This leaves n edges available to separate the other vertices: using the previous reasoning, these edges must be chosen in the same $\mathcal{E}_l$, $l \in [t]$ (otherwise they cannot cover Z, which is necessary to separate z_i^j from $\widetilde{z}_i$), and they induce a 3-dimensional matching on (V_l, T_l). □

4.1.2 Parameterization above a tight lower bound

The well-known result of Theorem 4.8 suggests that it is possible to study TestCover($\lceil \log n \rceil + k, k$), that is the parameterization of the Test Cover problem where we look for a solution of size at most $\lceil \log n \rceil + k$ and k is the parameter. Firstly, we prove that $\lceil \log n \rceil$ is a tight bound.

Theorem 4.11. *For every $n \in \mathbb{N}$, there exists a hypergraph $H = (V, \mathcal{T})$ such that $|V| = n$, $|\mathcal{T}| = \lceil \log n \rceil$ and $\mathcal{T}$ is a test cover.*

Proof. Let $V = \{v_1, \ldots, v_n\}$ and let r_l, $l \in [n]$, be the base-2 representation of l modulo $2^{\lceil \log n \rceil}$ which uses $\lceil \log n \rceil$ digits (see the proof of Theorem 4.10 for an example). Let r_{li} be the ith digit in r_l for $1 \leq i \leq \lceil \log n \rceil$. Let $\mathcal{T} = \{e_i : 1 \leq i \leq \lceil \log n \rceil\}$, where $e_i = \{v_l : r_{li} = 1\}$. Since the base-2 representation is unique, every vertex $v \in V$ is contained in a different subset of edges in $\mathcal{T}$, hence $\mathcal{T}$ is a test cover containing $\lceil \log n \rceil$ edges. □

Note that since in this case the parameter is always smaller than the size of the solution, Theorem 4.10 already excludes the possibility of the existence of a polynomial kernel, unless $coNP \subseteq NP/poly$. The situation is actually worse, as TestCover($\lceil \log n \rceil + k, k$) is not even likely to be in FPT. In fact, it is possible to construct a parameterized reduction from k-Set Cover (see Appendix A, Definition 7) to TestCover($\lceil \log n \rceil + k, k$). The proof is originally due to Moret and Shapiro [73], but we will follow the construction of De Bontridder *et al.* [26].

Let (H, k) be an instance of k-Set Cover with $V = \{v_1, \dots, v_n\}$ and $\mathcal{E} = \{e_1, \dots, e_m\}$. Let $V' = \{f_1, m_1, \dots, f_n, m_n\}$, that is, the vertex set of the instance of TestCover$(\lceil \log n \rceil + k, k)$ we want to construct contains a *female* vertex and a *male* vertex for every vertex in V. Let $\mathcal{M}$ be a set of $\lceil \log n \rceil$ edges separating all pairs of male vertices in V', which exists by Theorem 4.11, and let $\mathcal{M}'$ be obtained adding to the edges of $\mathcal{M}$ the female counterpart of every male vertex they contain, that is, $\mathcal{M}' = \{\{f_j, m_j : m_j \in e\} : e \in \mathcal{M}\}$.

The edge set $\mathcal{E}'$ of the TestCover$(\lceil \log n \rceil + k, k)$-instance will contain all edges in $\mathcal{M}'$ and, in addition, all $e' = \{f_j : v_j \in e\}$ for every $e \in \mathcal{E}$.

A test cover for $H' = (V', \mathcal{E}')$ must contain all the edges in $\mathcal{M}'$ in order to separate all the pairs of male vertices. This in turn separates all the pairs of female vertices, hence it is only left to separate f_i and m_i for every $i \in [n]$. This is possible using k edges if and only if H admits a set cover containing k edges.

Theorem 4.12. TestCover$(\lceil \log n \rceil + k, k)$ *is* $W[2]$*-hard.*

Proof. This is due to the fact that k-Set Cover is $W[2]$-complete (see Appendix A, Definition 7) and that there exists a parameterized reduction from it to TestCover$(\lceil \log n \rceil + k, k)$. □

4.1.3 Parameterization below the number of vertices

When the study of the standard parameterization of a problem does not lead to appealing FPT-algorithms, the attention is generally turned towards different parameterizations. Often, *dual* parameterizations are considered. Roughly speaking, a dual parameterization is obtained when the parameter is modified by a certain 'quantity', which is specific to the instance. We recall the definition of s-dual given in Section 1.5. Observe that here we assume that parameterized problems are defined as in Definition 1.11.

Definition 4.13. Let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem and let $s : \Sigma^* \to \mathbb{N}$ be a mapping such that $0 \le k \le s(x)$ for every $(x, k) \in Q$ and $s(x) \le |x|$ for every $x \in \Sigma^*$. The *s-dual* Q_s of Q is the parameterized problem corresponding to the language $Q_s = \{(x, s(x) - k) : (x, k) \in Q\}$.

Note that no requirements on the original parameterization of Q are made in the definition. When Q is parameterized according to the standard parameterization, a dual parameterization is obtained when the size of the solution is bounded above or below by $s(x) - k$. The function s is sometimes called a *size function* for the problem.

The size function may not be unique and a problem usually admits multiple duals. Nevertheless, the s-dual of the s-dual of Q is Q itself, which motivates the choice of the name. Generally speaking, the fact that a problem is FPT or admits a polynomial kernel does not imply anything about the duals, as it is the case with VERTEX COVER, which admits a linear kernel (in the number of vertices), and INDEPENDENT SET, which is $W[1]$-complete (see Appendix A, Definitions 11 and 12). Sometimes, though, the notion of duality can be used in a 'negative' way: an example of this can be found in the work of Chen *et al.* [13] (see Section 1.5), while another example will be given here.

Theorem 4.14. *Let Q be a parameterized problem which admits a cross-composition and let s be a size function. Let (x^*, k^*) be the instance associated to $(x_1, \ldots, x_t) \in (\Sigma^*)^t$, $t \in \mathbb{N}$, by the cross-composition algorithm $\mathcal{C}$. If for every x^* it holds that $s(x^*) \in \mathcal{O}((\max_{i=1}^t |x_i| + \log t)^{\mathcal{O}(1)})$, then the s-dual problem Q_s does not admit a polynomial kernel, unless $coNP \subseteq NP/poly$.*

Proof. We will show that the same cross-composition algorithm $\mathcal{C}$ produces a cross-composition for Q_s. Let $L \subseteq \Sigma^*$ be the NP-hard language which cross-composes into Q. By definition, $x_i \in L$ for some $i \in [t]$ if and only if $\mathcal{C}(x_1, \ldots, x_t) = (x^*, k^*) \in Q$. However, $(x^*, k^*) \in Q$ if and only if $(x^*, s(x^*) - k^*) \in Q_s$. Hence, this is a valid cross-composition if $s(x^*) - k^*$ is bounded by a polynomial in $\max_{i=1}^t |x_i| + \log t$, which is the case when the hypothesis hold. $\square$

We will soon show an application of Theorem 4.14, but first we give a description of TESTCOVER$(n - k, k)$. This problem is the s-dual parameterization of TESTCOVER(k, k) with size function $s((H, k)) = |V| = n$. Note that the size function is well-defined because if an instance admits a test cover, then it admits a test cover of size at most $n - 1$ (see Theorem 4.5).

An FPT-algorithm for TESTCOVER$(n - k, k)$ was first found by Crowston *et al.* [20] and it was later improved by Basavaraju *et al.* [4], which described an algorithm running in $\mathcal{O}(2^{\mathcal{O}(k^2)}(m + n)^{\mathcal{O}(1)})$ time. Their algorithm is based on a *partially polynomial kernelization* [5], that is, they show that it is possible to reduce the instance to one that contains at most $\mathcal{O}(k^7)$ vertices (but without a similar bound on the number of edges).

The existence of a 'totally' polynomial kernel, though, is ruled out under the usual $coNP \nsubseteq NP/poly$ assumption.

Theorem 4.15. TESTCOVER$(n - k, k)$ *does not admit a polynomial bikernel unless $coNP \subseteq NP/poly$.*

Proof. Theorem 4.10 describes a cross-composition from 3-DIMENSIONAL MATCHING into TESTCOVER(k, k). Recall that H denotes the hypergraph associated to the t instances $(V_1, T_1), \dots (V_t, T_t)$ of 3-DIMENSIONAL MATCHING and note that $s(H) = 3|V| + 3 + (\lceil \log t \rceil + 1)(|V| + 1)$, which is a polynomial in $\max_{i=1}^{t} |(V_i, T_i)| + \log t$. Then we conclude using Theorem 4.14. □

4.1.4 Parameterization below the number of edges

The last parameterization we consider is another dual of TESTCOVER(k, k). This time the size function s is the number of edges in the instance, which gives TESTCOVER$(m - k, k)$.

Theorem 4.16. *[20] TESTCOVER$(m - k, k)$ is $W[1]$-complete.*

Proof (sketch). The problem is in $W[1]$ because it can be reduced to $(m-k)$-SET COVER, the s-dual of k-SET COVER with s being the number of edges, which is $W[1]$-complete [73, 44]. To prove that it is $W[1]$-hard too, it is possible to show that there exists a parameterized reduction from k-INDEPENDENT SET. Let $G = (V, E)$ be an instance of the latter, we construct an instance $H' = (V', \mathcal{E}')$ of TESTCOVER$(m - k, k)$, where $V' = \{e, e' : e \in E\}$ and $\mathcal{E}' = \{\{e : v \in e\} : v \in V\} \cup \{\{e, e'\} : e \in E\} = \mathcal{E}_1 \cup \mathcal{E}_2$. It follows that the edges in $\mathcal{E}_2$ are always needed, while the edges in $\mathcal{E}_1$ that can be removed correspond to an independent set. □

4.2 Test cover with edges of bounded size

We have seen that the TEST COVER problem is difficult from the point of view of Parameterized Complexity. Out of the four parameterizations we have considered, the problem turns out to be $W[1]$-hard in two cases and does not admit a polynomial kernel in the other two.

To deal with it, a possible approach is to consider a restriction of the problem, the TEST-r-COVER problem. Here, as in the rest of this section, r is a fixed integer.

TEST-r-COVER(p)

Input:	A pair (H, p) where $H = (V, \mathcal{E})$ is a hypergraph such that $|e| \leq r$ for every $e \in \mathcal{E}$, and p is an integer.
Question:	Is there a subset $\mathcal{T} \subseteq \mathcal{E}$ with $|\mathcal{T}| \leq p$ such that for every $v, w \in V$ there exists $e \in \mathcal{T}$ with $|e \cap \{v, w\}| = 1$?

For some applications of the TEST COVER problem, edges do contain only a small number of vertices [26, 50], which gives the motivation for the study of this restriction. As before, we define the generic parameterization of the TEST-r-COVER problem.

TEST-r-COVER(p, k)

Input: A triplet (H, p, k) where $H = (V, \mathcal{E})$ is a hypergraph such that $|e| \leq r$ for every $e \in \mathcal{E}$, and p and k are integers.

Parameter: k

Question: Is there a subset $\mathcal{T} \subseteq \mathcal{E}$ with $|\mathcal{T}| \leq p$ such that for every $v, w \in V$ there exists $e \in \mathcal{T}$ with $|e \cap \{v, w\}| = 1$?

Note that in this special case obtaining a bound on the number of vertices is enough. The next lemma clarifies this point.

Lemma 4.17. *Let $H = (V, \mathcal{E})$ be a hypergraph such that $|e| \leq r$ for every $e \in \mathcal{E}$, and k be any integer. If $|V| \in \mathcal{O}(k^{c_1})$ for a constant $c_1 \in \mathbb{N}$, then $|\mathcal{E}| \in \mathcal{O}(k^{rc_1})$.*

Proof. Assume that $|V| \leq c_2 k^{c_1}$ for some constant $c_2 \in \mathbb{N}$. Using a well-known binomial inequality [61], the number m of edges is at most $\sum_{s=1}^{r} \binom{c_2 k^{c_1}}{s} \leq \sum_{s=1}^{r} (\frac{e c_2 k^{c_1}}{s})^s \in \mathcal{O}(r(e c_2 k^{c_1})^r) = \mathcal{O}(k^{rc_1})$. □

4.2.1 The standard parameterization (bounded case)

We have already seen that TESTCOVER(k, k) admits a trivial FPT-algorithm. In this section we will show that TEST-r-COVER(k, k) admits a kernel of polynomial size.

Theorem 4.18. TEST-r-COVER(k, k) *admits a kernel with at most $r(1+\max(0, (k - \lfloor \log r \rfloor)))$ vertices.*

Proof. The bound can be obtained with a counting argument as in the proof of Theorem 4.8, but making use of the fact that each edge has bounded size. We have already seen that if $\mathcal{E}' \subseteq \mathcal{E}$ induces t classes, $\mathcal{E}' \cup \{e\}$ $(e \in \mathcal{E} \setminus \mathcal{E}')$ can induce at most $2t$ classes. In addition, since e contains at most r vertices, e can split at most r of the classes induced by $\mathcal{E}'$, hence $\mathcal{E}' \cup \{e\}$ can induce at most $t + r$ classes. This means that $\mathcal{E}' \cup \{e\}$ induces at most $t + \min(t, r)$ classes.

We will now prove by induction that $\mathcal{E}'$ induces at most $2^{\lfloor \log r \rfloor} + \max(0, r(s - \lfloor \log r \rfloor))$ classes when $|\mathcal{E}'| \leq s$. If $s = 0$, $\mathcal{E}'$ induces one class and $2^{\lfloor \log r \rfloor} \geq 1$. Assume now that the bound holds for sets with at most $s - 1$ edges and that $|\mathcal{E}'| = s$. If $s \leq \lfloor \log r \rfloor$, then $\mathcal{E}'$ induces at most $2^{\lfloor \log r \rfloor}$ classes (see proof of Theorem 4.8). Otherwise, let $e \in \mathcal{E}'$ and observe that by induction hypothesis $\mathcal{E}' \setminus e$ induces at most $2^{\lfloor \log r \rfloor} + r(s - 1 - \lfloor \log r \rfloor)$ classes, as

$s-1-\lfloor \log r \rfloor \geq 0$. Then $\mathcal{E}'$ induces at most $2^{\lfloor \log r \rfloor}+r(s-1-\lfloor \log r \rfloor)+r \leq 2^{\lfloor \log r \rfloor}+r(s-\lfloor \log r \rfloor)$ classes, which concludes the inductive step.

Hence, if $|V| > 2^{\lfloor \log r \rfloor} + \max(0, r(k-\lfloor \log r \rfloor))$ the answer is NO, and otherwise $|V| \leq r(1+\max(0,(k-\lfloor \log r \rfloor)))$. □

Theorem 4.18 proves the existence of a kernel for TEST-r-COVER(k,k) with a linear number of vertices. Nevertheless, it is possible to obtain different bounds.

Lemma 4.19. *Any test cover $\mathcal{T}$ contains at least $\lceil \frac{2(n-1)}{r+1} \rceil$ edges, and this bound is tight.*

Proof. For any test cover $\mathcal{T}$ with $|\mathcal{T}| = t$ there exist at most one vertex which is not contained in any edge, at most t vertices which are contained in exactly one edge (one vertex for every edge), and at most $\frac{tr}{2}$ vertices which are contained in two or more edges (where tr is the sum over the vertices contained in all the edges and we divide by 2 to take into account that every vertex is contained in at least two edges). Let t' be the number of edges in $\mathcal{T}$ which only contain vertices which are contained in at least two edges in $\mathcal{T}$. Then it follows that $n \leq 1+t-t'+\frac{t(r-1)+t'}{2} \leq 1+t+\frac{t(r-1)}{2}$, which implies that $t \geq \frac{2(n-1)}{r+1}$. Since t is an integer, $t \geq \lceil \frac{2(n-1)}{r+1} \rceil$.

To show that the bound is tight, we will describe a class of instances for which there exists a test cover of size $\lceil \frac{2(n-1)}{r+1} \rceil$. Let r be any integer and let $V = \{v_{ij} : i,j \in [r]\}$. The set $\mathcal{E}$ of edges contains the 'rows' and 'columns' of V (except the last ones), that is, $\mathcal{E} = \{e_i = \{v_{i1},\dots,v_{ir}\} : i \in [r-1]\} \cup \{e'_j = \{v_{1j},\dots,v_{rj}\} : j \in [r-1]\}$. For $i,j \in [r-1]$, v_{ij} is the only vertex contained in both e_i and e'_j, v_{rj} is the only vertex contained in e'_j only and the same holds for v_{ir} and e_i. Finally, v_{rr} is the only vertex which is not contained in any edge. It follows that $\mathcal{E}$ is a test cover. Also, $n = r^2$ and $|\mathcal{E}| = 2(r-1) = 2\frac{r^2-1}{r+1}$. □

Note that the proof of Lemma 4.19 also shows that when $|\mathcal{T}| = \frac{2(n-1)}{r+1}$ every vertex is contained in at most two edges, $|\mathcal{T}|$ vertices are contained in exactly one edge and there exists a vertex which is not contained in any edge.

Theorem 4.20. TEST-r-COVER(k,k) *admits a kernel with at most $\frac{k(r+1)}{2}+1$ vertices.*

Proof. If $k < \lceil \frac{2(n-1)}{r+1} \rceil$ by Lemma 4.19 the answer is NO, otherwise $k \geq \frac{2(n-1)}{r+1}$, that is, $n \leq \frac{k(r+1)}{2}+1$. □

The bound provided by Theorem 4.20 is better that the bound provided by Theorem 4.18 when r is small compared to k, which is likely to be the case as r is assumed to be a 'small' constant.

Theorem 4.21. TEST-r-COVER(k,k) *admits a kernel with* $\mathcal{O}(rk)$ *vertices and* $\mathcal{O}(k^r)$ *edges.*

Proof. The results of Theorem 4.18 or Theorem 4.20 imply the bound on the number of vertices. The bound on the number of edges is ensured by Lemma 4.17. $\square$

4.2.2 Parameterization above a tight lower bound (bounded case)

Lemma 4.19 is useful in itself, as it shows a lower bound on the size of a test cover which is better than the trivial lower bound $\lceil \log n \rceil$. Hence, in this section we study TEST-r-COVER$(\lceil \frac{2(n-1)}{r+1} \rceil + k, k)$ instead of TEST-r-COVER$(\lceil \log n \rceil + k, k)$.

Lemma 4.22. *If* $r \geq 3$, TEST-r-COVER$(\lceil \frac{2(n-1)}{r+1} \rceil + k, k)$ *is* NP*-hard for* $k = 0$.

Proof. We will show that there exists a reduction from r-DIMENSIONAL MATCHING. Let (V', T') be an instance of the latter, with $V' = \{v'_1, \ldots, v'_{n'}\}$. Assume that n' is divisible by $r-1$ [2] and let $s = \frac{n'}{r-1}$. We will construct an equivalent instance $(H = (V, \mathcal{E}), 0)$ of TEST-r-COVER$(\lceil \frac{2(n-1)}{r+1} \rceil + k, k)$ as follows. Let

$$V = V_1 \cup W_1 \cup W_0 = \{v_{ij} : i \in [r], j \in [n']\} \cup \{w_{ij} : i \in [r-1], j \in [s]\} \cup \{w_0\}$$

$$\begin{aligned}
\mathcal{E} = \quad & \mathcal{E}_1 \cup \mathcal{E}_2 \\
= \quad & \{\{v_{1j_1}, \ldots, v_{rj_r}\} : (v'_{j_1}, \ldots, v'_{j_r}) \in T'\} \cup \\
& \cup \{e_{ij} = \{v_{ij_1}, \ldots, v_{ij_{r-1}}, w_{ij}\} : i \in [r-1], j \in [s], j_l = (r-1)(j-1) + l\}
\end{aligned}$$

The edges in $\mathcal{E}_1$ correspond to the subsets in T', while the edges in $\mathcal{E}_2$ contain $r-1$ vertices from V_1 and one vertex from W_1. It holds that $n = |V| = rn' + (r-1)\frac{n'}{r-1} + 1 = (r+1)n' + 1$, and $\lceil \frac{2(n-1)}{r+1} \rceil = 2n'$.

Note that, since w_0 is not contained in any edge, every other vertex must be, hence the edges in $\mathcal{E}_2$ are contained in every test cover (otherwise there would exist vertices in W_1 which are not contained in any edge). Moreover, $|\mathcal{E}_2| = s(r-1) = n'$. We will show that (V', T') admits an r-dimensional matching M if and only if $(H, 0)$ is a `YES`-instance of TEST-r-COVER$(\lceil \frac{2(n-1)}{r+1} \rceil + k, k)$.

[2] It is not difficult to see that if r-DIMENSIONAL MATCHING is polynomial time solvable on instances (V', T') where $|V'| = n'$ is fixed, then it is polynomial time solvable on instances (V'', T'') with $|V''| \leq n'$. Hence the assumption on n' is not restrictive.

If M is an r-dimensional matching for (V',T'), let $\widetilde{\mathcal{E}} = \{\{v_{1j_1},\dots,v_{rj_r}\} : (v'_{j_1},\dots,v'_{j_r}) \in M\}$. Since the size of M is n', then the size of $\widetilde{\mathcal{E}} \cup \mathcal{E}_2$ is $2n'$. By the property defining an r-dimensional matching, for every $i \in [r], j \in [n']$ the vertex v_{ij} is contained in exactly one edge in $\widetilde{\mathcal{E}}$: denote this edge $\widetilde{e}_{ij}$ (obviously the edges $\widetilde{e}$ thus defined are not all distinct). Hence, every vertex $v_{ij} \in V_1$, $0 \le j \le r-1$, is the only vertex contained in both $\widetilde{e}_{ij}$ and $e_{ij'}$, where $j' = \lfloor \frac{j}{r-1} \rfloor + 1$, while v_{rj} is the only vertex contained in $\widetilde{e}_{rj}$ only. Moreover, w_{ij} is the only vertex contained in e_{ij} only and w_0 is the only vertex which is not contained in any edge. This shows that $\widetilde{\mathcal{E}} \cup \mathcal{E}_2$ is a test cover of size $\lceil \frac{2(n-1)}{r+1} \rceil$ for H.

On the other hand, suppose $\mathcal{T} = \widetilde{\mathcal{E}} \cup \mathcal{E}_2$ is a test cover of size $\lceil \frac{2(n-1)}{r+1} \rceil = 2n'$ for H. In this case, we know by Lemma 4.19 that there is one vertex, which has to be w_0, which is not contained in any edge of $\mathcal{T}$, there are $|\mathcal{T}|$ vertices which are contained in one edge only ($|\mathcal{T}|/2 = n'$ of them have to be the vertices in W_1) and all the others are contained in exactly two edges. For $v \in V$, let $d(v)$ be the degree of v in the 'solution' hypergraph $H_{\mathcal{T}} = (V,\mathcal{T})$. It holds that $\sum_{v \in V_1} d(v) = n' + 2(r-1)n'$. Since edges in $\mathcal{E}_2$ contain $(r-1)n'$ vertices of V_1, edges in $\widetilde{\mathcal{E}}$ have to contain rn' of them, which means that they must be disjoint: this ensures that they correspond to subsets that form an r-dimensional matching M for (V',T'). □

Lemma 4.23. Test-2-Cover($\lceil \frac{2}{3}(n-1) \rceil + k, k$) *is NP-hard for $k = 0$.*

Proof. We will show that there exists a reduction from the NP-complete problem P_3-Packing (see Appendix A, Definition 6). Let $G' = (V',E')$ be an instance of P_3-Packing, let $V' = \{v_1,\dots,v_{n'}\}$ and let $n'/3 = s$. We will construct an equivalent instance $(H = (V,\mathcal{E}),0)$ of Test-2-Cover($\lceil \frac{2}{3}(n-1) \rceil + k, k$). Let $V = V' \cup \{v_0\}$, where v_0 is a new vertex, and let $\mathcal{E} = \{\{v_i,v_j\} : v_iv_j \in E'\}$. It holds that $\lceil \frac{2}{3}(|V|-1) \rceil = \frac{2}{3}n'$.

Assume G' contains $n'/3$ vertex-disjoint copies of P_3 and let E'' be the set of edges contained in these copies. In this case $\mathcal{T} = \{\{v_i,v_j\} : v_iv_j \in E''\}$ is a test cover containing $\frac{2}{3}n'$ edges.

On the other hand, assume $\mathcal{T} \subseteq \mathcal{E}$ is a test cover containing $\frac{2}{3}n'$ edges. By Lemma 4.19 there is one vertex of H which is not contained in any edge of $\mathcal{T}$ (which can only be v_0), $\frac{2}{3}n'$ vertices contained in one edge only and $\frac{1}{3}n'$ vertices contained in two edges. This means that the edges in $\mathcal{T}$ correspond to $n'/3$ vertex-disjoint copies of P_3 in G'. □

Theorem 4.24. Test-r-Cover($\lceil \frac{2(n-1)}{r+1} \rceil + k, k$) *is para-$NP$-complete for each $r \ge 2$.*

Proof. By Lemma 4.22 and Lemma 4.23, Test-r-Cover($\lceil \frac{2(n-1)}{r+1} \rceil + k, k$) is NP-hard for $k = 0$ and $r \ge 2$. Since the problem is obviously in NP, it follows that Test-r-Cover($\lceil \frac{2(n-1)}{r+1} \rceil +$

$k, k)$ is NP-complete for $k = 0$ and $r \geq 2$. Hence, we conclude using Theorem 1.15. □

4.2.3 Parameterization below the number of vertices (bounded case)

The situation in this case resembles what happens with the standard parameterization. In fact, we will show that the restriction on the size of the edges allows to find a polynomial kernel for TEST-r-COVER$(n-k, k)$. However, the proof is more involved than the one needed for TEST-r-COVER(k, k).

Let $(H = (V, \mathcal{E}), k)$ be an instance of TEST-r-COVER$(n-k, k)$. Recall that we have assumed that $\mathcal{E}$ is always a test cover for $H = (V, \mathcal{E})$. So when $k = 1$, the answer to the problem is `YES` (see Theorem 4.5), therefore we may assume $k \geq 2$.

Definition 4.25. A set of edges $\mathcal{F} \subseteq \mathcal{E}$ is a k-mini test cover if $|\mathcal{F}| \leq 2k$ and the number of classes induced by $\mathcal{F}$ is at least $|\mathcal{F}| + k$.

The notion of k-mini test cover allows us to define a greedy algorithm that was used by Crowston *et al.* [20] to show that TESTCOVER$(n-k, k)$ is FPT, and that will be used here as the starting point for the kernelization. The description of the greedy algorithm is as follows.

```
   input  : $(H, k)$ and $\mathcal{E}' \subseteq \mathcal{E}$
   output: A set $\mathcal{F} \subseteq \mathcal{E}'$ with at most $2k - 1$ edges
1  Set $\mathcal{F} = \emptyset$ ;
2  while $|\mathcal{F}| < 2k - 2$ do
3      if there exists $e \in \mathcal{E} \setminus \mathcal{F}$ such that $e$ splits at least two classes induced
       by $\mathcal{F}$ then                                                   /* Case 1 */
4          Add $e$ to $\mathcal{F}$ ;
5      else if if there exist $e, e' \in \mathcal{E} \setminus \mathcal{F}$ such that $e$ splits one class induced by $\mathcal{F}$ and $e'$
       splits at least two classes induced by $\mathcal{F} \cup \{e\}$ then              /* Case 2 */
6          Add $e$ and $e'$ to $\mathcal{F}$ ;
7      else
9          return $\mathcal{F}$ ;
10
11 end
13 return $\mathcal{F}$ ;
```

Algorithm 4.1: Greedy k-mini test cover

Lemma 4.26. *If Algorithm 4.1 terminates at step 13, then $\mathcal{F}$ is a k-mini test cover.*

Proof. The `while` loop is repeated as long as $|\mathcal{F}| < 2k - 2$ and every time at most two edges are added, hence it always holds that $|\mathcal{F}| < 2k$. In addition, $\mathcal{F}$ induces at least $\frac{3}{2}|\mathcal{F}|+1$ classes. In fact, the empty set already induces a class (the whole set V of vertices) and whenever Case 1 or Case 2 applies the number of induced classes increases by at least $\frac{3}{2}s$, where s is the number of new edges which are added to $\mathcal{F}$.

Therefore, if the algorithm terminates at step 13, then $2k - 2 \leq |\mathcal{F}| < 2k$ and $\mathcal{F}$ induces at least $|\mathcal{F}| + \frac{1}{2}|\mathcal{F}| + 1 \geq |\mathcal{F}| + k$ classes. □

Lemma 4.27. *The following statements are equivalent:*

1. *$\mathcal{E}$ contains a test cover with at most $n - k$ edges;*

2. *$\mathcal{E}$ contains a k-mini test cover.*

Proof. If 2 holds, let $\mathcal{F}$ be a k-mini test cover. Repeatedly add to $\mathcal{F}$ an edge $e \in \mathcal{E} \setminus \mathcal{F}$ which splits at least one class and call $\mathcal{T}$ the resulting set: $\mathcal{T}$ is a test cover as $\mathcal{E}$ is a test cover; moreover, the number of edges that we have added is at most $n - (|\mathcal{F}| + k)$, which means that $|\mathcal{T}| \leq |\mathcal{F}| + n - (|\mathcal{F}| + k) = n - k$.

On the other hand, if 1 holds, let $\mathcal{T} \subseteq \mathcal{E}$ be a test cover with at most $n - k$ edges. Use Algorithm 4.1 on $\mathcal{T}$ to produce a set $\mathcal{F}$. If the algorithm terminates at step 13 we are done by Lemma 4.26, so assume that it terminates at step 9. Observe that this means that $|\mathcal{F}| < 2k-2$ and Case 1 and 2 of the algorithm do not apply.

We claim that for every $\mathcal{F}' \subseteq \mathcal{T}$ which contains $\mathcal{F}$, every $e \in \mathcal{T} \setminus \mathcal{F}'$ splits at most one class induced by $\mathcal{F}'$. By induction on $s = |\mathcal{F}'| - |\mathcal{F}|$, if $s = 1$ the claim holds, or Case 1 of the algorithm would apply. For the inductive step, let $\mathcal{F}'$ contain $s + 1$ edges more than $\mathcal{F}$ and assume for the sake of contradiction that there exists $e \in \mathcal{T} \setminus \mathcal{F}'$ which splits two classes C_1, C_2 induced by $\mathcal{F}'$. Note that these classes must be contained in the same class C induced by $\mathcal{F}$ (otherwise Case 1 of the algorithm would hold and e would be added to $\mathcal{F}$). Let e' be an edge of $\mathcal{F}'$ which contains exactly one between C_1 and C_2 (which exists by Lemma 4.3). Then e and e' together satisfy the requirements of Case 2 of the algorithm, which is a contradiction.

Hence, if we add edges to $\mathcal{F}$ one by one to form $\mathcal{T}$, every edge increases the number of classes by at most one. Let s be the number of classes induced by $\mathcal{F}$. Then $n \leq s + |\mathcal{T}| - |\mathcal{F}|$, that is, $s \geq |\mathcal{F}| + k$. So in every case the output of Algorithm 4.1 is a k-mini test cover. □

Lemma 4.27 shows that the problem of finding a test cover of size $n - k$ is equivalent to the problem of finding a k-mini test cover. Moreover, it shows that given a test cover of size $n-k$ we can compute a k-mini test cover, and vice versa. On the other hand, the next lemma shows that the notion of k-mini test cover can be relaxed, without losing this property.

Lemma 4.28. *If there exists a set of edges $\mathcal{E}' \subseteq \mathcal{E}$ which induces at least $|\mathcal{E}'| + k$ classes, then $\mathcal{E}$ contains a k-mini test cover $\mathcal{F}$ such that $\mathcal{F} \subseteq \mathcal{E}'$.*

Proof. Let l be the number of classes induced by $\mathcal{E}'$ and let $H_{\mathcal{E}'}[V]$ be as in Definition 4.6. For simplicity, let $\widetilde{V}$ denote the vertex set of $H_{\mathcal{E}'}[V]$ and $\widetilde{\mathcal{E}}$ denote its edge set. Then $|\widetilde{V}|-k = l-k \geq |\mathcal{E}'| = |\widetilde{\mathcal{E}}|$, and $\widetilde{\mathcal{E}}$ is a test cover in $H_{\mathcal{E}'}[V]$. Hence by Lemma 4.27 there exists a k-mini test cover $\widetilde{\mathcal{E}}'$ in $H_{\mathcal{E}'}[V]$. The corresponding edge set $\mathcal{F} = \{e_j : e'_j \in \widetilde{\mathcal{E}}'\}$ is a k-mini test cover in H, which is contained in $\mathcal{E}'$. □

Note that Algorithm 4.1 provides a k-mini test cover when the input is a test cover which contains at most $n - k$ edges. Otherwise, the result may not be a k-mini test cover, but it is still a set of edges with useful properties, as the next lemma shows.

Lemma 4.29. *In polynomial time we may either find a k-mini test cover or find $\mathcal{F} \subseteq \mathcal{E}$ such that:*

1. *$|\mathcal{F}| < 2k$;*
2. *$\mathcal{F}$ induces less than $|\mathcal{F}| + k$ classes;*
3. *every $e \in \mathcal{E} \setminus \mathcal{F}$ splits at most one class induced by $\mathcal{F}$;*
4. *for any $e, e' \in \mathcal{E} \setminus \mathcal{F}$ and any class C induced by $\mathcal{F}$, at least one of $(e \cap e') \cap C$, $(e \setminus e') \cap C$, $(e' \setminus e) \cap C$ and $C \setminus (e \cup e')$ is empty.*

Proof. Use Algorithm 4.1 on $\mathcal{E}$ (it is not difficult to see that the algorithm runs in polynomial time). The output is a set $\mathcal{F} \subseteq \mathcal{E}$ which contains less than $2k$ edges. If $\mathcal{F}$ is not a k-mini test cover, then 2 holds and, by Lemma 4.26, the algorithm terminated at step 9. Hence, 3 and 4 hold because otherwise Case 1 or 2 of the algorithm would apply. □

In order to obtain a kernel, we want to show that even when Algorithm 4.1 does not compute a k-mini test cover it is still possible to reduce the instance to one that has a number of vertices bounded by a polynomial in k.

Let $\mathcal{F} \subseteq \mathcal{E}$ be a set of edges produced by Algorithm 4.1, satisfying the Conditions 1, 2, 3 and 4 of Lemma 4.29. Call $C_1, \ldots, C_l, G$ ($l \in \mathbb{N}$) the classes induced by $\mathcal{F}$, where G is the class of vertices which are not contained in any edge $e \in \mathcal{F}$. Let $\mathcal{C}$ be the set of classes $C_1, \ldots, C_l$ and let $C = C_1 \cup \cdots \cup C_l$. Let $\mathcal{G} \subseteq \mathcal{E}$ be the set of edges that contains vertices in G. For each edge $e \in \mathcal{G}$, we say that $e \cap G$ is the *G-portion* of e. A subset Γ of G is a *component* if Γ is the G-portion of an edge $e \in \mathcal{G}$ and $\Gamma \not\subset e' \cap G$ for all $e' \in \mathcal{G}$. Every component Γ contains at most r vertices.

Note that $|C| \leq (2k-1)r$, as $|\mathcal{F}| < 2k$ and every edge contains at most r vertices. If $|G| \leq 2r$, then $|V| = |C| + |G| \leq (2k+1)r$, that is, the number of vertices is linearly bounded in k. Suppose this is not the case. It follows that $G \setminus (e \cup e')$ is nonempty for every $e, e' \in \mathcal{E}$. Then, by Condition 4 of Lemma 4.29, one of $(e \cap e') \cap G = \emptyset$, $(e \setminus e') \cap G$, $(e' \setminus e) \cap G$ is empty.

Algorithm 4.2 (Kernelization for Test-r-Cover$(n-k, k)$).

> **Step 1:** For each pair $(i, j) \in [l] \times [l]$, $i \neq j$, mark $2k$ unmarked components of G which contain the G-portion of an edge which contains C_i and does not intersect C_j and mark these edges too. If there are less than $2k$ components mark them all. Let E_{ij} denote the set of marked edges.
>
> For each $i \in [l]$, mark $2k+1$ unmarked components of G which contain the G-portion of an edge containing C_i and mark these edges too. If there are less than $2k+1$ components, mark them all. Let E_i denote the set of marked edges.
>
> **Step 2:** Delete every edge in $\mathcal{G}$ whose G-portion is not contained in a marked component of G, then delete every vertex which is not contained in any edge anymore, except one vertex w_0 (if it exists).

Algorithm 4.2 is the main tool which will be used to produce a kernel for Test-r-Cover$(n-k, k)$. We will apply it only to instances with $|G| > 2r$, in order to have the property that two G-portions are either disjoint or one is contained in the other. Call $(H' = (V', \mathcal{E}'), k)$ the instance it computes. It is not difficult to see that Algorithm 4.2 runs in polynomial time. The next lemma shows that it is effective in reducing the size of H, while Lemma 4.31 shows that (H', k) is a `YES`-instance if and only if (H, k) is.

Let $G' \subseteq G$ be the set of vertices which are not deleted by Algorithm 4.2. Clearly, $V' = C \cup G'$.

Lemma 4.30. *V' contains at most $\mathcal{O}(rk^3)$ vertices.*

Proof. Vertices of G' which are not deleted must be contained in components which were marked. The algorithm marks at most $2kl(l-1) + (2k+1)l$ components and each of them contains at most r vertices. Using the fact that $l < |\mathcal{F}|+k < 3k$, we have that $|G'| \leq r(2k(3k-1)(3k-2)+(2k+1)(3k-1))$. Since $|C| \leq (2k-1)r$, we conclude that $|V'| \in \mathcal{O}(rk^3)$. $\square$

Lemma 4.31. *If $|V| > (9k^2+4k-1)r$, the instance (H', k) computed by Algorithm 4.2 is equivalent to the original one.*

Proof. We will show that (H', k) admits a k-mini test cover if and only if (H, k) admits one.

Obviously, if (H', k) admits a k-mini test cover, this is a k-mini test cover for (H, k) too. For the other direction, suppose $\mathcal{T}$ is a k-mini test cover for $(V, \mathcal{E}, k)$ such that $\mathcal{T} \setminus \mathcal{E}'$ is as small as possible. For the sake of contradiction, suppose that $\mathcal{T}$ contains at least one edge e in $\mathcal{T} \setminus \mathcal{E}'$. We claim that it is possible to construct a set $\mathcal{T}'''$ which induces at least $|\mathcal{T}'''|+k$ classes, such that $\mathcal{T}''' \setminus \mathcal{E}' = (\mathcal{T} \setminus \mathcal{E}') \setminus \{e\}$. Then, by applying Lemma 4.28 to $\mathcal{T}'''$, we obtain a k-mini test cover in H which is a subset of $\mathcal{T}'''$, which is a contradiction as it contains fewer edges from $\mathcal{E} \setminus \mathcal{E}'$ than $\mathcal{T}$.

Start with $\mathcal{T}' = \mathcal{T} \setminus \{e\}$ and let $i, j \in [l]$. Since e is not in $\mathcal{E}'$, e must be in $\mathcal{G}$, and the G-portion of e must not be contained in any marked component. Furthermore, for each $C_i, C_j \in \mathcal{C}$ $(i \neq j)$ with $C_i \subseteq e$ and $e \cap C_j = \emptyset$, we note that $E_{i,j}$ must contain $2k$ edges, as otherwise e would be in $\mathcal{E}'$. Similarly, for each C_i contained in e we note that E_i must contain $2k+1$ edges.

For any i, j such that $|E_{i,j}| = 2k$, let $e_{i,j}$ be an edge in $E_{i,j}$ whose G-portion is disjoint from the G-portion of any edge in $\mathcal{T}'$. This must exist as $|\mathcal{T}'| \leq 2k-1$. For any i such that $|E_i| = 2k+1$ let e_i, e_i' be edges in E_i whose G-portions are disjoint from the G-portion of any edge in $\mathcal{T}'$. These edges must exist as $|\mathcal{T}'| \leq 2k-1$.

Let C_0^* be the class induced by $\mathcal{T}'$ that consists of all vertices not in any edge in $\mathcal{T}'$ (which exists by Claim A below). We will need the following claims.

Claim A: C_0^* exists and $|G \cap C_0^*| > (9k^2+1)r$.

Proof of Claim A: If C_0^* does not exist or $|G \cap C_0^*| \leq (9k^2+1)r$, then the following holds and we have a contradiction to the assumption on $|V|$ in the hypothesis:

$$|V| \leq (2k-1)r + |C_0^*| \leq 2kr + (|C| + |G \cap C_0^*|) \leq 2(2k-1)r + (9k^2+1)r = (9k^2+4k-1)r$$

Claim B: For each edge $e' \in \mathcal{E}$ that splits G and every C_i we have $C_i \subseteq e'$ or $C_i \cap e' = \emptyset$. In particular, $C_i \subseteq e$ or $C_i \cap e = \emptyset$.

Proof of Claim B: If Claim B is false then $\mathcal{F}$ does not satisfy Condition 3 of Lemma 4.29, as e' splits both G and C_i.

Claim C: There is at most one class C_G^* induced by $\mathcal{T}'$, such that $G \cap (C_G^* \cap e) \neq \emptyset$ and $G \cap (C_G^* \setminus e) \neq \emptyset$.

Proof of Claim C: For the sake of contradiction assume that there are two such classes C_G' and C_G''. By Lemma 4.3 there exists $e' \in \mathcal{T}'$ which contains one but not the other. Hence adding e and e' to $\mathcal{F}$ would increase the number of classes by at least three, as vertices in $G \cap (C_G' \cap e)$, $G \cap (C_G' \setminus e)$, $G \cap (C_G'' \cap e)$ and $G \cap (C_G'' \setminus e)$ are separated by $\mathcal{F} \cup \{e, e'\}$, which means that $\mathcal{F}$ does not satisfy Condition 4 of Lemma 4.29.

Let $C_1^*, \ldots, C_t^*$ be the classes induced by $\mathcal{T}'$ that are split by e and are different from C_G^* (if it exists) and from C_0^*. Each C_s^*, $s \in [t]$, must be contained in an edge, say e_s^*, in $\mathcal{T}'$ and contain vertices from C, by the definitions on C_G^* and C_0^*. We are going to create a set of edges $\mathcal{T}''$ such that each C_s^* is split by an edge in $\mathcal{T}''$ and also $\mathcal{T}''$ induces $|\mathcal{T}''|$ extra classes in C_0^*. Initially let $\mathcal{T}'' = \emptyset$. For each $s \in [t]$ in turn, consider the following two cases.

Case 1*:* For some $i \neq j$, e contains C_i but not C_j and $C_i \cap C_s^* \neq \emptyset \neq C_j \cap C_s^*$.

In this case observe that $|E_{i,j}| = 2k$, as otherwise e would be marked. Then add the edge $e_{i,j}$ to $\mathcal{T}''$, if $e_{i,j}$ is not in $\mathcal{T}''$ already. Note that $e_{i,j}$ separates C_i from C_j and therefore splits C_s^*, and also creates an extra class in C_0^*, as desired.

Case 2*:* Case 1 does not hold. Using Claim B, this means that $C \cap C_s^* \subseteq e$ or $(C \cap C_s^*) \cap e = \emptyset$.

Recall that, since C_s^* is different from C_G^*, there exists a C_i such that $C_s^* \cap C_i \neq \emptyset$. Suppose e does not contain C_i. Then $(C \cap C_s^*) \cap e = \emptyset$ and, since e splits C_s^*, it must contain vertices from $C_s^* \cap G \subseteq e_s^* \cap G$ (which also means that $G \cap (e \cap e_s^*) \neq \emptyset$). Then e_s^* splits G and the G-portion of e_s^* is in the same component as the G-portion of e, and therefore e_s^* is an unmarked edge. Furthermore since e_s^* splits G it does not split C_i, and therefore $C_i \subseteq e_s^*$. Thus, we have that either e or e_s^* is an unmarked edge containing C_i, and therefore $|E_i| = 2k+1$. Then add e_i to $\mathcal{T}''$, if e_i is not already in $\mathcal{T}''$. Observe that e_i splits C_s^* as it contains vertices in $C_i \cap C_s^*$ but no vertex from $(C_s^* \cap G) \cap e$ (which is nonempty since otherwise e cannot split C_s^*), and e_i creates an extra class in C_0^*, as required.

This completes Case 1 and Case 2. Note that the edges in $\mathcal{T}''$ all have disjoint G-portions, as they are in distinct $E_{i,j}$'s and E_i's, and their G-portions are all contained in C_0^*. Also, note that $|\mathcal{T}''| \leq l(l-1) + l \leq 3k(3k-1) + 3k$, which means that the union of the G-portions of edges in $\mathcal{T}''$ contains at most $r(3k(3k-1)+3k) = 9k^2r < |C_0^* \cap G|$ vertices. Moreover, for every $s \in [t]$ some edge in $\mathcal{T}''$ splits C_s^*. Therefore, if we add the edges from $\mathcal{T}''$ to $\mathcal{T}'$, this creates at least $|\mathcal{T}''| + t$ additional classes ($|\mathcal{T}''|$ in C_0^* and t in $V' \setminus C_0^*$).

We now consider Case (i) and Case (ii) below, which will complete the proof.

Case (i): C_G^* does not exist or is equal to C_0^* or e does not split C_0^* or does not split C_G^*.

In this case removing e from $\mathcal{T}$ decreases the number of classes by at most $t+1$ and adding $\mathcal{T}''$ increases the number of classes by at least $t + |\mathcal{T}''|$. Hence by increasing the number of edges by $|\mathcal{T}''| - 1$ we have increased the number of classes by at least $|\mathcal{T}''| - 1$ and therefore we still have at least k more classes than edges.

Case (ii): Case (i) does not hold. That is, C_G^* exists and is distinct from C_0^* and e splits both C_G^* and C_0^*.

By Claim C we note that e either contains all of $C_0^* \cap G$ or none of $C_0^* \cap G$. By Claim A e must contain none of $C_0^* \cap G$. As e splits C_0^* we must have $C \cap e \cap C_0^* \neq \emptyset$. Therefore there exists C_i such that e contains vertices from $C_i \cap C_0^*$, and so $|E_i| = 2k + 1$. Add e_i and e_i' to $\mathcal{T}''$ (unless e_i is already in $\mathcal{T}'$, in which case just add e_i'). Observe that the G-portions of e_i and e_i' are disjoint by construction, and so the G-portions of all edges in $\mathcal{T}''$ are still disjoint.

Note that adding e_i and e_i' to $\mathcal{T}'$ creates three new classes in C_0^* (C_0^* now being split into the class $C_0^* \cap e_i \cap e_i'$ which contains vertices from C_i, the G-portion of e_i, the G-portion of e_i' and the class of vertices not in any edge). Adding every other edge from $\mathcal{T}''$ to $\mathcal{T}'$ increases the number of classes in C_0^* by one (as by Claim A we note that some vertex in $G \cap C_0^*$ is not contained in any edge in $\mathcal{T}''$).

So let $\mathcal{T}''' = \mathcal{T}' \cup \mathcal{T}'' = (\mathcal{T} \setminus e) \cup \mathcal{T}''$. Removing e from $\mathcal{T}$ decreases the number of classes by $t+2$ and adding $\mathcal{T}''$ increases the number of classes by at least $t+|\mathcal{T}''|+1$. So by increasing the number of edges by $|\mathcal{T}''| - 1$ we have increased the number of classes by at least $|\mathcal{T}''| - 1$ and therefore we still have at least k more classes than edges. □

Note that the lower bound on the size of $|V|$ in the hypothesis of Lemma 4.31 is only needed to ensure that the G-portions of the edges that we add to $\mathcal{T}''$ cannot entirely cover $C_0^* \cap G$, in which case the last edge we add would not induce an additional class in C_0^*.

Theorem 4.32. Test-r-Cover$(n-k, k)$ *admits a kernel with at most $\mathcal{O}(rk^3)$ vertices and $\mathcal{O}(k^{3r})$ edges.*

Proof. Let (H, k) be an instance of Test-r-Cover$(n-k, k)$. Run Algorithm 4.1 on $\mathcal{E}$. If the output $\mathcal{F}$ is a k-mini test cover, then (H, k) is a `YES`-instance. Otherwise, assume $|V| > (9k^2 + 4k + 1)r$, which in particular ensures that $|G| > 2r$, run Algorithm 4.2 on H and let (H', k) be the instance which is computed. By Lemma 4.31 H' admits a k-mini test cover if and only if H admits one. In addition, by Lemma 4.30 H' contains at most $\mathcal{O}(rk^3)$

vertices. Finally, by Lemma 4.17, H' contains at most $\mathcal{O}(k^{3r})$ edges. $\square$

4.2.4 Parameterization below the number of edges (bounded case)

The case where the bound on the size of the edges appears to significantly reduce the difficulty of the problem is Test-r-Cover$(m-k,k)$. Recall that TestCover$(m-k,k)$ is $W[1]$-complete. Instead, we will see in this section that Test-r-Cover$(m-k,k)$ is not only FPT, but it admits a polynomial kernel.

Let $(H=(V,\mathcal{E}),k)$ be an instance of Test-r-Cover$(m-k,k)$, with $|V|=n$ and $|\mathcal{E}|=m$. We consider the following generalization of the problem:

Subset-Test-r-Cover$(m-k)$

Input:	A triplet $(H,\mathcal{B},k)$ where $H=(V,\mathcal{E})$ is a hypergraph such that $\lvert e\rvert \leq r$ for every $e\in\mathcal{E}$, $\mathcal{B}\subseteq\mathcal{E}$ and k is an integer.
Parameter:	k
Question:	Is there a subset $\mathcal{T}\subseteq\mathcal{E}$ with $\lvert\mathcal{T}\rvert\leq p$ such that $\mathcal{B}\subseteq\mathcal{T}$ and for every $v,w\in V$ there exists $e\in\mathcal{T}$ with $\lvert e\cap\{v,w\}\rvert=1$?

In other words, edges in $\mathcal{B}$ cannot be removed from $\mathcal{E}$ to form $\mathcal{T}$. We call these edges *black edges*.

Reduction Rule 4.33. *Let $v\in V$ be a vertex of degree 1 and let $b\in\mathcal{B}$ be a black edge which contains only v. Delete b and v.*

Reduction Rule 4.34. *Let $b\in\mathcal{B}$. If there exists $e\in\mathcal{E}$ with $b\subset e$, then delete e and add $e\setminus b$ to $\mathcal{E}$ if e is not black, and to $\mathcal{B}$ otherwise.*

Reduction Rule 4.35. *Let $b,b'\in\mathcal{B}$. If $b\cap b'\neq\emptyset$, delete b and b' and add $b\cap b'$, $b\setminus b'$ and $b'\setminus b$ to $\mathcal{B}$.*

Rules 4.33, 4.34 and 4.35 will be applied to an instance in this order, that is, we always apply the first applicable reduction rule.

Notice that for every edge $e\in\mathcal{E}$, every pair v,w of vertices that is separated by $\mathcal{E}$ but is not separated by $\mathcal{E}\setminus\{e\}$ is of the form $v\in e$ and $w\notin e$ (or vice versa). This fact will be often used in the following proofs.

Lemma 4.36. *Reduction Rules 4.33, 4.34 and 4.35 are valid.*

Proof. It is not difficult to see that each of these rules can be applied in polynomial time. Let $(H' = (V', \mathcal{E}'), \mathcal{B}', k)$ be an instance obtained after an application of Rule 4.33, 4.34 or 4.35.

Rule 4.33: Suppose $(H', \mathcal{B}', k)$ is a `YES`-instance, with solution $\mathcal{T}'$. Then observe that $\mathcal{T} = \mathcal{T}' \cup \{b\}$ is a solution for $(H, \mathcal{B}, k)$. Conversely, if $\mathcal{T}$ is a solution for $(H, \mathcal{B}, k)$, then $\mathcal{T}$ must contain b, and $\mathcal{T} \setminus \{b\}$ is a solution for $(H', \mathcal{B}', k)$.

Rule 4.34: It is sufficient to show that for any $\mathcal{T} \subseteq \mathcal{E}$ containing e and b, $\mathcal{T}$ is a test cover if and only if $(\mathcal{T} \setminus \{e\}) \cup \{e \setminus b\}$ is a test cover. To see this, observe that for any $v \in e, w \notin e$, v and w are separated either by b or $e \setminus b$, and for any $v \in e \setminus b$, $w \notin e \setminus b$, v and w are separated either by b or e.

Rule 4.35: Similar to the previous case, if v and w are separated by one of b, b' then they are also separated by at least one of $b \setminus b'$, $b' \setminus b$, $b \cap b'$, and if they are separated by one of $b \setminus b'$, $b' \setminus b$, $b \cap b'$ then they are also separated by at least one of b, b'. □

From now on, assume that $(H, \mathcal{B}, k)$ is reduced under Reduction Rules 4.33, 4.34 and 4.35.

Lemma 4.37. *For every black edge $b \in \mathcal{B}$ there exists a non-black edge $e \in \mathcal{E} \setminus \mathcal{B}$ such that $b \cap e \neq \emptyset$.*

Proof. Note that every edge contains at most one vertex of degree one. Since H is reduced under Reduction Rule 4.33, then b contains at least one vertex of degree at least two. Let e be a different edge containing it. Since H is reduced under Reduction Rule 4.35, e cannot be black. □

Consider now the following algorithm.

```
input  : $(H,k)$ and a vertex $v \in V$ with degree greater than $kr^2$
output: A set $\widetilde{V}$ of vertices
1  Set $\widetilde{\mathcal{E}} = \mathcal{E}$, $i = 1$, $\widetilde{V} = \{v\}$, $j = 1$;
2  while $i \leq k+1$ do
3      if $\widetilde{\mathcal{E}}$ isolates $\widetilde{V}$ then
4          Let $e_i$ be an edge containing $\widetilde{V}$, and construct a set $E_i \subseteq \widetilde{\mathcal{E}}$ such that $E_i \cup \{e_i\}$
           isolates $\widetilde{V}$ and $|E_i| \leq r-1$;
5          Set $\widetilde{\mathcal{E}} = \widetilde{\mathcal{E}} \setminus (E_i \cup \{e_i\})$ ;
6          Set $i = i+1$;
7      else
8          Let $C$ be the class induced by $\widetilde{\mathcal{E}}$ containing $\widetilde{V}$;
9          Set $\widetilde{V} = C, i = 1, j = j+1$;
10     end
11 end
12 return $\widetilde{V}$ ;
```

Algorithm 4.3: Vertices with bounded degree

Note that Algorithm 4.3 runs in time polynomial in its input. Its purpose is to compute an instance in which every vertex has bounded degree, as the next lemma shows.

Lemma 4.38. *The instance (H,k) can be reduced in polynomial time to an equivalent instance such that every vertex has degree at most kr^2.*

Proof. Assume that there exists a vertex v with degree greater than kr^2. We will show that we are able to produce an equivalent instance in which either k or the degree of v is reduced. Clearly this reduction can only take place a polynomial number of times, so in polynomial time we will reduce to an instance in which every vertex has degree bounded by kr^2.

Use Algorithm 4.3 on H and v to produce a special set $\widetilde{V}$. Observe that at any step of the algorithm, by construction any edge in $\widetilde{\mathcal{E}}$ which contains v also contains $\widetilde{V}$ as a subset, $|\widetilde{V}| \geq j$ and $\widetilde{\mathcal{E}} \subseteq \mathcal{E}$. Also, at any step at most r edges are deleted, and at most kr edges can be deleted before j is increased. Hence, if the algorithm ever sets $j = r+1$, then at that point at most kr^2 edges have been deleted from $\widetilde{\mathcal{E}}$, and, as $|\widetilde{V}| \geq r+1$, no remaining edges in $\widetilde{\mathcal{E}}$ contain v. But this is a contradiction as the degree of v is greater than kr^2. Therefore we may assume the algorithm never reaches $j = r+1$. Hence the algorithm must terminate for some $j \leq r$.

We now show that as long as $\widetilde{\mathcal{E}}$ isolates $\widetilde{V}$ we can find e_i and E_i. Since v has degree greater than kr^2 and at most $kr(r-1)$ edges are removed from $\widetilde{\mathcal{E}}$ earlier in the algorithm, we can always find an edge e_i containing v and therefore containing $\widetilde{V}$. To see that E_i can be constructed using at most $r-1$ edges, apply Lemma 4.7 to $\widetilde{V}$ and $e_i \setminus \widetilde{V}$.

Now consider the set $\widetilde{V}$ formed by the algorithm. When the algorithm terminated, e_i and E_i were found for all $i \leq k+1$. If we delete k arbitrary edges in $\mathcal{E}$, it is still possible to find $i \in [k+1]$ such that no edges in $E_i \cup \{e_i\}$ have been deleted. This means that as long as we delete at most k edges, $\widetilde{V}$ is still isolated. Therefore if $\widetilde{V}$ is an edge in $\mathcal{E}$, delete it and reduce k by 1 (note that $\widetilde{V}$ cannot be a black edge, as it is contained in at least two distinct edges). If $\widetilde{V}$ is not an edge in $\mathcal{E}$, add a new black edge $\widetilde{V}$ to $\mathcal{B}$, keeping k the same, and apply Rules 4.33, 4.34 and 4.35. Observe that since $\widetilde{V}$ is properly contained in at least two edges (even before adding the black edge), this will decrease the degree of every vertex in $\widetilde{V}$. □

We can now assume that $(H, \mathcal{B}, k)$ is reduced under Reduction Rules 4.33, 4.34 and 4.35, and the degree of every vertex is at most kr^2. To describe the rest of the kernelization we need to colour all the edges of the instance in the following way.

Algorithm 4.4. Let e be any edge in $\mathcal{E} \setminus \mathcal{B}$. If $\mathcal{E} \setminus \{e\}$ is not a test cover, then colour e black (and add it to $\mathcal{B}$), then apply Reduction Rules 4.33, 4.34 and 4.35 as long as possible. If $\mathcal{E} \setminus \{e\}$ is a test cover and e contains a vertex of degree one, then colour e orange. Otherwise colour e green.

Note that there will be orange edges only if there is no isolated vertex in the instance.

Reduction Rule 4.39. *Let $o \in \mathcal{E}$ be an orange edge. If $N_2[o]$ contains no green edges, then delete o, decrease k by one, run again Algorithm 4.4 and apply Rules 4.33, 4.34 and 4.35 as long as possible.*

Notice that running Algorithm 4.4 after an application of Rule 4.39 will turn all the remaining orange edges into black edges, as deleting an orange edge creates an isolated vertex. Hence this rule may apply only once.

Lemma 4.40. *Reduction Rule 4.39 is valid.*

Proof. It is not difficult to see that this rule can be applied in polynomial time. To show that the instance it produces is equivalent to the original one is sufficient to prove that in the original instance there exists a test cover of minimum size which does not contain o.

Suppose $\mathcal{T}$ is a test cover which contains o. If there exists a vertex v which is not contained in any edge in $\mathcal{T}$, let $e \in \mathcal{E}$ be an edge containing it (which exists as otherwise no edge would be coloured orange). Otherwise, let e be any edge in $\mathcal{E} \setminus \mathcal{T}$.

Consider $\mathcal{T}' = (\mathcal{T} \setminus \{o\}) \cup \{e\}$. We claim that $\mathcal{T}'$ is a test cover.

First of all, note that if there is an orange edge $o' \in \mathcal{E} \setminus \mathcal{T}$, this edge must be e. In fact, o' contains a vertex v' of degree one, which is the only isolated vertex in $\mathcal{T}$, namely v; furthermore, o' is the only edge containing it and is therefore equal to e. Since by hypothesis $N_2[o]$ may contain only black or orange edges, then we may assume that $N_2[o] \setminus \{o\} \subseteq \mathcal{T}'$.

The only vertices that may no longer be separated in $\mathcal{T}'$ are pairs u, w where $u \in o$ and $w \in V \setminus o$. If u is a vertex of degree one in $\mathcal{E}$, then it is an isolated vertex in $\mathcal{T}'$, and it is the only one because of the choice of e. In any other case there exists an edge $e' \in N_1(o)$ which contains u. So the only case left to consider is when $u \in o \cap e'$ and $w \in e' \setminus o$ with $e' \in N_1(o)$. Since $\mathcal{E} \setminus \{o\}$ is a test cover, there exists an edge e'' different from o separating u and w, which means that $e'' \in N_2[o] \setminus \{o\} \subseteq \mathcal{T}'$ and we are done. $\square$

We assume that $(H, \mathcal{B}, k)$ is reduced under Reduction Rules 4.33, 4.34, 4.35 and 4.39, and the degree of every vertex is at most kr^2.

Lemma 4.41. *Let v be a vertex of degree at least one. Then $v \in V(N_3[g])$ for some green edge $g \in \mathcal{E}$.*

Proof. If v is contained in a green edge, we are done. If v is contained in an orange edge o, since H is reduced under Reduction Rule 4.39 then $N_2[o]$ contains a green edge g, and $v \in V(N_2[g]) \subseteq V(N_3[g])$. Finally, if v is contained in a black edge $b \in \mathcal{B}$, then, by Lemma 4.37, b has nonempty intersection with some edge $e \notin \mathcal{B}$. If e is green we are done, otherwise there exists a green edge $g \in N_2[e]$, which means that $v \in V(N_3[g])$. $\square$

Lemma 4.42. *If $\mathcal{G}$ is a set of green edges such that, for every pair $g_1, g_2 \in \mathcal{G}$, $N_1[g_1] \cap N_1[g_2]$ is empty, then $\mathcal{E} \setminus \mathcal{G}$ is a test cover.*

Proof. The proof is by induction on $|\mathcal{G}|$. If $|\mathcal{G}| = 1$, then $\mathcal{E} \setminus \mathcal{G}$ is a test cover by definition. Assume now that $\mathcal{G} = \{g_1, \dots, g_{s+1}\}$, $s \geq 1$, and $\mathcal{E} \setminus \{g_1, \dots, g_s\}$ is a test cover. For the sake of contradiction, assume that v and w are vertices that are not separated in $\mathcal{E} \setminus \mathcal{G}$. Then one of them must be in g_{s+1} and the other in g_i for some $i \in [s]$. Let $v \in g_{s+1}$. Since g_{s+1} is green, there must be an edge $e \in \mathcal{E}$, different from g_{s+1}, which contains v. Since v and w are not separated in $\mathcal{E} \setminus \mathcal{G}$, then $w \in e$, but this implies that $e \in N_1[g_i]$, which is a contradiction. $\square$

We are now ready to prove the existence of a polynomial kernel. The idea behind the proof is that if there are many green edges then we can find k of them with disjoint neighbourhoods and solve the problem using Lemma 4.42, otherwise Lemma 4.41 shows that the remaining edges are not 'far' from a small set of green edges.

Theorem 4.43. Subset-Test-r-Cover$(m-k)$ *admits a kernel with at most* $(k-1)k^5r^{16}+1$ *vertices and* $(k-1)k^5r^{16}+k$ *edges.*

Proof. Let $(H, \mathcal{B}, k)$ be an instance coloured with Algorithm 4.4, reduced under Reduction Rules 4.33, 4.34, 4.35 and 4.39, and where the degree of every vertex is bounded by kr^2. This can be done in polynomial time, as all the rules are valid and they either decrease the number of vertices or reduce the degree of some of them.

Now, greedily construct a set $\mathcal{G} \subseteq \mathcal{E}$ of green edges for which the hypothesis of Lemma 4.42 hold. If $|\mathcal{G}| \geq k$ then we are done. Otherwise, $|\mathcal{G}| \leq k-1$ and every green edge is in $N_2[\mathcal{G}]$. Hence, by Lemma 4.41, every edge $e \in \mathcal{E}$ is contained in $N_3[N_2[\mathcal{G}]] = N_5[\mathcal{G}]$. In particular, every vertex of degree at least one is contained in $V(N_5[\mathcal{G}])$, whose cardinality is less than $r|N_5[\mathcal{G}]|$.

Furthermore, observe that for every $\mathcal{F} \subseteq \mathcal{E}$, $|N_1[\mathcal{F}]| \leq (kr^2)r|\mathcal{F}|$ because of the bound on the degree. Hence it holds that $r|N_5[\mathcal{G}]| \leq r^5(kr^2)^5|\mathcal{G}|r \leq r^5(kr^2)^5(k-1)r$. In addition, in a `YES`-instance there is at most one isolated vertex, which implies that $|V| \leq (k-1)k^5r^{16}+1$.

To bound the number of edges we show that there exists a solution of size at most $|V|$ (note that due to the black edges we cannot use the $|V|-1$ bound). Start with $\mathcal{T} = \mathcal{B}$. Since the black edges are all disjoint (as the hypergraph is reduced under Reduction Rule 4.35), then $|\mathcal{B}| \leq |V|$; moreover, for the same reason, they induce at least $|\mathcal{T}|$ classes. If $\mathcal{T}$ is a test cover we are done, otherwise repeatedly add an edge which splits at least one class, which is always possible if $\mathcal{E}$ is a test cover. Eventually, $\mathcal{T}$ will induce $|V|$ classes and will contain at most $|V|$ edges. Hence if $|\mathcal{E}| - k \geq |V|$ and $\mathcal{E}$ is a test cover, then the instance is a `YES`-instance. Otherwise, $|\mathcal{E}| \leq |V| + k - 1 = (k-1)k^5r^{16} + k$. □

Corollary 4.44. Test-r-Cover$(m-k,k)$ *admits a kernel with at most* $5((k-1)k^5r^{16}+1)$ *vertices and* $3(k-1)k^5r^{16}+k+2$ *edges.*

Proof. First we transform an instance (H,k) of Test-r-Cover$(m-k,k)$ to an equivalent instance $(H, \mathcal{B}, k)$ of Subset-Test-r-Cover$(m-k)$, with $\mathcal{B} = \emptyset$. Then we apply the kernelization on $(H, \mathcal{B}, k)$ and we compute an instance $(H', \mathcal{B}', k')$ where the bounds of Theorem 4.43 hold. Finally we transform $(H', \mathcal{B}', k')$ to an equivalent instance $(H'' = (V'', \mathcal{E}''), k'')$

of TEST-r-COVER$(m-k,k)$ in the following way. Let $b \in \mathcal{B}'$ be a black edge. If $\mathcal{E} \setminus \{b\}$ is not a test cover then simply uncolour b. Otherwise, note that edges are coloured black only by Rules 4.34 and 4.35, Lemma 4.38 and Algorithm 4.4: in the first two cases the edge is a proper subset of some edge previously coloured black and in the third case it is contained in at least $k''+1$ edges, hence it holds that $|b| \le r-1$; finally, in the last case, deleting b does not produce a test cover, which is against the assumptions. We will show that it is possible to replace b with a small gadget which forces b to be in the solution.

Add vertices v_1, v_2, v_3, v_4 to V'', delete b and add $b \cup \{v_1\}$, $e' = \{v_1, v_2, v_3\}$ and $e'' = \{v_3, v_4\}$. Observe that e' is necessary to separate v_3 and v_4, e'' is necessary to separate v_2 and v_3 and $b \cup \{v_1\}$ is necessary to separate v_1 and v_2. Hence all three edges must be in a test cover. Moreover, having b in a test cover in H' is equivalent to having $b \cup \{v_1\}, e', e''$ in a test cover in H''.

The last thing to observe is that for every black edge we add four new vertices and two new edges. Since $|\mathcal{B}'| \le |V'|$, it holds that $|V''| \le |V'| + 4|V'| \le 5((k-1)k^5r^{16}+1)$ and $|\mathcal{E}''| \le |\mathcal{E}'| + 2|V'| \le 3((k-1)k^5r^{16}) + k + 2$. □

We conclude this section describing an FPT-algorithm for SUBSET-TEST-r-COVER$(m-k)$, which works for TEST-r-COVER$(m-k,k)$ too.

Theorem 4.45. SUBSET-TEST-r-COVER$(m-k)$ *can be solved in time* $(r^2+1)^k(n+m)^{\mathcal{O}(1)}$ *on* $(H, \mathcal{B}, k)$.

Proof. We first need to guess whether there will be a vertex not contained in any edge in the solution, and if so, which vertex it will be. If there already exists a vertex w_0 not in any edge in $\mathcal{E}$, then we are done. Otherwise, either pick a vertex v which is not contained in any black edge, or guess that every vertex in V will be contained in an edge in the solution. If a vertex v is picked, delete all the edges containing v, and reduce k by the number of deleted edges. If it is guessed that every vertex in V will be contained in an edge, add a new vertex w_0 which is not in any edge. Observe that this does not change the solution to the problem. By doing this we have split the problem into at most $n+1$ separate instances, with each instance containing an isolated vertex. Thus we may now assume that there exists a vertex w_0 which is not contained in any edge in $\mathcal{E}$.

Consider an edge $e \in \mathcal{E} \setminus \mathcal{B}$, and suppose that $\mathcal{E} \setminus \{e\}$ is a test cover. Let $\mathcal{E}_0$ be a minimal set of edges in $\mathcal{E} \setminus \{e\}$ which contain every vertex in e. Note that such a set must exist, as otherwise there would be vertices in e which are not contained in any edge in $\mathcal{E} \setminus \{e\}$ and cannot be separated from w_0. Furthermore, we may assume $|\mathcal{E}_0| \leq |e| \leq r$. Now for each $e' \in \mathcal{E}_0$, let $\mathcal{E}_{e'}$ be a minimal set of edges in $\mathcal{E} \setminus \{e\}$ separating every vertex in $e' \setminus e$ from every vertex in $e' \cap e$. By Lemma 4.7, we may assume that $|\mathcal{E}_{e'}| \leq r - 1$.

Now let $\mathcal{E}' = \mathcal{E}_0 \cup (\bigcup_{e' \in \mathcal{E}_0} \mathcal{E}_{e'})$, and observe that $\mathcal{E}'$ isolates e. Thus, in every solution with minimum number of edges, at least one edge from $\mathcal{E}' \cup \{e\}$ will be missing. Note that $|\mathcal{E}'| \leq r + r(r-1) = r^2$.

We now describe a depth-bounded search tree algorithm which solves the problem. If $\mathcal{E}$ is not a test cover, return `NO`. Otherwise if $k = 0$ return `YES`. Otherwise, for each edge $e \in \mathcal{E} \setminus \mathcal{B}$ check whether $\mathcal{E} \setminus \{e\}$ is a test cover. If for all $e \in \mathcal{E} \setminus \mathcal{B}$, $\mathcal{E} \setminus \{e\}$ is not a test cover, then a test cover must contain all m edges and so return `NO`. Otherwise, let e be a non-black edge such that $\mathcal{E} \setminus \{e\}$ is a test cover, and construct the set $\mathcal{E}'$ as described above. Then we may assume one of $\mathcal{E}' \cup \{e\}$ is not in the solution. Thus we may pick one non-black edge from $\mathcal{E}' \cup \{e\}$, delete it, and reduce k by 1. So we split into at most $r^2 + 1$ instances with reduced parameter.

We therefore have a search tree with at most $(r^2 + 1)^k$ leaves and where the distance from every leaf to the root (the vertex corresponding to the original instance) is at most k. Hence, the total number of nodes is at most $2(r^2 + 1)^k - 1$. Note also that guessing the isolated vertex at the start split the problem into at most $n+1$ instances, so there are at most $(n+1)(2(r^2+1)^k - 1)$ nodes to compute in total. As each node in the tree takes polynomial time to compute, we have an algorithm with total running time $(r^2 + 1)^k (n + m)^{\mathcal{O}(1)}$. □

Chapter 5

λ-Extendible Properties

The class of problems we study in this chapter arises from the generalization of a classical combinatorial problem, namely Max Cut.

Max Cut	
Input:	A pair (G, k) where $G = (V, E)$ is a connected simple graph and k is an integer.
Question:	Is there a subset $U \subseteq V$ such that $\lvert E(U, V \setminus U)\rvert \geq k$?

As always, let $n = |V|$ and $m = |E|$. For any $U \subseteq V$, the set $E(U, V \setminus U)$ is a *cut*. In the weighted version of the problem, denoted Weighted Max Cut, every edge has a positive weight and the objective is to find a cut whose weight (that is, the sum of the weights of its edges) is greater than k. Under this formulation, the problem appears in Karp's list of 21 NP-complete problems [62].

Another way to look at the problem is in terms of bipartite subgraphs. Note that the graph $H = G\big[E(U, V \setminus U)\big]$ is a bipartite subgraph of G whose weight is equal to the weight of the cut $E(U, V \setminus U)$. On the other hand, the edge set of every maximal bipartite subgraph H of G (that is, a bipartite subgraph which is not properly contained in any other bipartite subgraph) forms a cut in G of the same weight. It is clear then that the problem of finding a cut of weight at least k is equivalent to the problem of finding a bipartite subgraph of weight at least k. Therefore, the Weighted Max Cut problem is part of the wide class of problems whose objective is to find a heavy (*i.e.*, large) subgraph satisfying some particular properties.

These problems are often NP-hard, but sometimes it is possible to find lower bounds on the weight of an optimal solution (that is, one whose weight is the greatest possible).

In particular, in 1965 Erdős described a randomized algorithm for Max Cut, running in polynomial time, which computes a cut with at least $\frac{m}{2}$ edges [35], and in 1973 Edwards improved the result, showing that a connected graph always contains a cut with $\frac{m}{2} + \frac{n-1}{4}$ edges [34, 33]: this became known as the Edwards-Erdős bound.

Initially, Erdős had conjectured that it was possible to obtain a solution with $\frac{m}{2}+\varepsilon m$ edges, but later it was proved that finding a cut of that size is NP-hard for every $\varepsilon > 0$ [48, 74]. Additionally, constant approximation over the $\frac{m}{2}$ lower bound is impossible if the Unique Game Conjecture holds [63], and the best one can hope for is a cut with $\left(\frac{1}{2} + \Omega(\frac{\varepsilon}{\log(1/\varepsilon)})\right)m$ edges when the graph contains a cut of size $(\frac{1}{2} + \varepsilon)m$. On the positive side, it is possible to find in randomized polynomial time a cut which contains an $\alpha - \varepsilon$ fraction of the edges of a maximum cut, where $\alpha > 0.87856$ and $\varepsilon > 0$ [42], but if the Unique Game Conjecture holds this is the best possible approximation [64].

The Max Cut problem has been studied from the point of view of Parameterized Complexity too. In 1997 Mahajan and Raman proved that there is an algorithm to find a cut of size $\frac{m}{2} + k$ running in $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time [70]. In 2002 Bollobàs and Scott obtained a similar result using the strongest $\frac{m}{2} + \frac{1}{8}(\sqrt{8m+1} - 1)$ bound [11]. The existence of an FPT-algorithm for the parameterization above the Edwards-Erdős bound remained an open question for a time, until in 2012 it was positively solved by Crowston *et al.* [23], who also showed the existence of a kernel with $\mathcal{O}(k^5)$ vertices.

Observe that the Edwards-Erdős bound is tight for infinitely many non-isomorphic graphs, as it is tight for every complete graph with an odd number of vertices. In addition, a cut whose size is at least the Edwards-Erdős bound can be found in $\mathcal{O}(m)$ time [74]. When a problem admits this sort of lower bound, parameterizing by the size of the solution, the so-called standard parameterization, is not particularly interesting. In the case of Max Cut, suppose we are looking for a cut of size k: if $k < \frac{m}{2}$, then the answer is obviously `YES`, otherwise $m < 2k$. This reasoning produces a kernel of linear size for the standard parameterization of Max Cut, but it does not shed light on how it is possible to efficiently compute solutions whose size is greater than the Edwards-Erdős bound.

We have seen examples of parameterizations above tight lower bounds in Chapter 4 with TestCover($\lceil \log n \rceil + k, k$) and Test-$r$-Cover($\lceil \frac{2(n-1)}{r+1} \rceil + k, k$), which also showed how often this kind of parameterizations are significantly harder to solve than the standard ones. In this chapter, though, we will see an example of a class of problems which admit a polynomial kernel when parameterized above a tight lower bound.

For the rest of this chapter, let $\mathcal{G}$ denote a class of (possibly labelled and/or oriented) graphs, such that $\mathcal{U}(G)$ is a simple graph for every $G \in \mathcal{G}$. A *graph property* Π is a subset of $\mathcal{G}$. Let K_s be the complete (unlabelled and unoriented) graph with s vertices: we say that $K_s \in \Pi$ if $G \in \Pi$ for every (oriented, labelled) graph $G \in \mathcal{G}$ with $\mathcal{U}(G) = K_s$.

Definition 5.1. [78] Let $\lambda \in \mathbb{R}^+$ and $0 < \lambda < 1$. A graph property $\Pi \subseteq \mathcal{G}$ is λ-*extendible* if it satisfies:

1. Inclusiveness: K_1 and K_2 are in Π,
2. Block additivity: a graph $G \in \mathcal{G}$ is in Π if and only if each block of G is in Π,
3. λ-edge extension: given a graph $G = (V, E)$, a weight function $w_G : E \to \mathbb{R}^+$ and a partition U, W of V such that $U = \{u, v\}$, with $uv \in E$, and $G[W] \in \Pi$, in polynomial time it is possible to find $F \subseteq E(U, W)$ satisfying $w_G(F) \geq \lambda w_G\big(E(U, W)\big)$ and so that the graph $G \setminus \big(E(U, W) \setminus F\big) \in \Pi$.

Poljak and Turzík introduced the notion of λ-extendible property with the objective of generalizing the Weighted Max Cut problem. The next theorem will make this point clearer.

Theorem 5.2. *Let $\mathcal{G}_\mathcal{S}$ contain all simple graphs, let $\Pi_{BP} \subseteq \mathcal{G}_\mathcal{S}$ contain all bipartite graphs and let $\lambda = \frac{1}{2}$. Then Π_{BP} is a λ-extendible property.*

Proof. K_1 and K_2 are bipartite graphs, so Inclusiveness holds. As for Block additivity, note that every cycle of a graph is contained in one of its blocks and, additionally, a graph is bipartite if and only if it contains no odd cycles, so combining these facts implies that a graph is bipartite if and only if each of its blocks is. Finally, note that a graph $G = (V, E)$ is bipartite if and only if there exists a partition V_0, V_1 of V such that $E = E(V_0, V_1)$. Hence, assume that $U = \{u, v\}$ and W are given as in the hypothesis of λ-edge extension and that a partition V_0, V_1 is given for $V - \{u, v\}$, such that $E[W] = E(V_0, V_1)$. Either add u to V_0 and v to V_1, or add u to V_1 and v to V_0, then delete the edges of G which have both endpoints in V_i, $i = 0, 1$. The graph G' which is obtained is bipartite by construction, and in at least one of the two cases the weight of the edges in $E(V \setminus \{u, v\}, \{u, v\})$ which G' contains is at least half of the weight of all the edges in $E(V \setminus \{u, v\}, \{u, v\})$. □

Observe that the Weighted Max Cut problem is equivalent to the problem of finding a subgraph H of a graph G such that $H \in \Pi_{BP}$ and the weight of H is at least k. By abuse of language, we can say that Weighted Max Cut is a λ-extendible property for $\lambda = \frac{1}{2}$.

Other well-known λ-extendible properties are the property of 'being q-colourable' for simple graphs (where q is fixed) and the property of 'being acyclic' for oriented graphs [78]. Moreover, in Section 5.3 we will study another property ('being signed', a generalization of 'being bipartite' to signed graphs) which is λ-extendible. See also Chapter 6 for more examples.

The characteristic that Poljak and Turzík wanted to capture with their definition is the existence of the Edwards-Erdős bound. Indeed, for λ-extendible properties, a generalization of the Edwards-Erdős bound holds.

Lemma 5.3. *Let $G = (V, E)$ be a 2-connected graph. Then for each vertex $u \in V$ there exist edges uv and uw such that both $G - \{u, v\}$ and $G - \{u, w\}$ are connected.*

Proof. Since G is 2-connected, $G - \{u\}$ is connected. If $G - \{u\}$ has only one block, then u has at least two neighbours v and w in it, otherwise its only neighbour would be a cutvertex in G. If $G - \{u\}$ has more than one block, then it has more than one pendant block, and u has a neighbour in the interior of every pendant block, otherwise the root of that block is a cutvertex in G. Thus, let v and w be two neighbours of u contained in the interiors of two different pendant blocks. In both cases, both $G - \{u, v\}$ and $G - \{u, w\}$ are connected. $\square$

Let $\Pi \subseteq \mathcal{G}$ be a λ-extendible property for $0 < \lambda < 1$. For every weighted graph $G \in \mathcal{G}$, let $\mathsf{opt}(G)$ be the maximum weight of a subgraph of G which is in Π and let $\mathsf{wt}(G)$ be the minimum weight of a spanning tree of G. Recall that $\mathfrak{B}(G)$ denotes the set of blocks of G.

Theorem 5.4. *For any graph $G = (V, E) \in \mathcal{G}$ and any weight function $w_G : E \to \mathbb{R}^+$, it is possible to find in polynomial time a subgraph $H \in \Pi$ such that $w_G(E(H)) \geq \lambda w_G(E) + \frac{1-\lambda}{2}\mathsf{wt}(G)$. In particular, $\mathsf{opt}(G) \geq \lambda w_G(E) + \frac{1-\lambda}{2}\mathsf{wt}(G)$.*

Proof. The proof is by induction on $|V|$. If G is disconnected, we can prove the result for every component separately, so we may assume that G is connected. Since by Inclusiveness K_1 and K_2 are in Π, the result holds when $|V| \leq 2$. Now, assume that $|V| \geq 3$. First, assume that G is 2-connected. Let u be a vertex in V, and let v and w be as in Lemma 5.3. Without loss of generality assume that $w_G(uv) \geq w_G(uw)$ and let $\widetilde{F} = E(V \setminus \{u, v\}, \{u, v\}) \subseteq E$. Using the inductive hypothesis, we may assume that $G - \{u, v\}$ contains a subgraph $H' = (V', E') \in \Pi$ such that $w_G(E') \geq \lambda w_G(E \setminus (\widetilde{F} \cup \{uv\})) + \frac{1-\lambda}{2}(w_G(T'))$, where T' is a spanning tree of $G - \{u, v\}$ of minimum weight. By λ-edge extension, there exists $F \subseteq \widetilde{F}$ satisfying $w_G(F) \geq \lambda w_G(\widetilde{F})$ and so that the subgraph H of G with vertex set $V(H) = V' \cup \{u, v\}$ and edge set $E(H) = E' \cup F \cup \{uv\}$ is in Π. Let T'' be a spanning tree of G formed adding uv and uw to T'. Hence,

$$\begin{aligned} w_G(E(H)) &= w_G(E') + w_G(F) + w_G(uv) \\ &\geq \lambda w_G(E \setminus (\widetilde{F} \cup \{uv\})) + \frac{1-\lambda}{2}(w_G(T')) + \lambda w_G(\widetilde{F}) + w_G(uv) \\ &= \lambda w_G(E) + \frac{1-\lambda}{2}(w_G(T')) + (1-\lambda) w_G(uv) \\ &\geq \lambda w_G(E) + \frac{1-\lambda}{2}(w_G(T'')) \end{aligned}$$

where the last inequality holds since $w_G(uv) \geq w_G(uw)$. Then we conclude using the fact that by minimality $w_G(T'') \geq \mathsf{wt}(G)$.

If, on the other hand, G is not 2-connected, we may apply the inductive hypothesis on each of its blocks, which is justified because of Block additivity and because $\lambda w_G(E) + \frac{1-\lambda}{2}\mathsf{wt}(G) = \sum_{B \in \mathfrak{B}(G)} \left(\lambda w_G(E(B)) + \frac{1-\lambda}{2}\mathsf{wt}(B)\right)$ [1] .

Note that a recursive algorithm performing these steps takes polynomial time, as the block decomposition of a graph can be found in $\mathcal{O}(m+n)$ time, the number of recursive calls is bounded by $\mathcal{O}(m)$ and each step can be performed in polynomial time. □

The lower bound of Theorem 5.4 is known as the Poljak-Turzík bound. We denote by $\mathsf{pt}(G)$ the value of the Poljak-Turzík bound on a graph G. Note that if $G = (V, E)$ is a connected graph in which every edge has weight one, and $\lambda = \frac{1}{2}$, then $\mathsf{pt}(G) = \frac{1}{2}|E| + \frac{1}{4}(|V| - 1)$, which corresponds to the Edwards-Erdős bound on G.

Let $\mathsf{ex}(G) = \mathsf{opt}(G) - \mathsf{pt}(G)$ be the *excess* of Π on $G \in \mathcal{G}$. Observe that by Theorem 5.4, $\mathsf{ex}(G) \geq 0$ for every $G \in \mathcal{G}$. We are interested in the following generic parameterized problem:

Weighted Above Poljak-Turzík (Π) ($WAPT(\Pi)$)	
Input:	A pair (G, k) where $G = (V, E)$ is a connected weighted graph, and k is an integer.
Parameter:	k
Question:	Does it hold that $\mathsf{ex}(G) \geq k$?

In Section 5.1, we will study a restriction of the $WAPT(\Pi)$ (which is still quite general) and show that it can be reduced to an easier problem. Then in Section 5.2, building on

[1]Note that every edge is contained in exactly one block and a minimum weight spanning tree for G results from the union of minimum weight spanning trees for every block.

this result, we will show that the $APT(\Pi)$, the unweighted version of $WAPT(\Pi)$, admits a polynomial kernel in most of the cases.

Before concluding this section, we prove some technical results which will be useful in the rest of the chapter. Recall that $\mathfrak{C}(H)$ denotes the connected components of a graph H.

Lemma 5.5. *Let $G = (V, E) \in \mathcal{G}$ and let v be a cutvertex in G. Then*

$$\mathsf{ex}(G) = \sum_{C \in \mathfrak{C}(G - \{v\})} \mathsf{ex}(G[V(C) \cup \{v\}])$$

Proof. Let $G_C = G[V(C) \cup \{v\}]$ for $C \in \mathfrak{C}(G - \{v\})$. To begin with, note that $\mathsf{pt}(G) = \sum_{C \in \mathfrak{C}(G-\{v\})} \mathsf{pt}(G_C)$. In fact, every edge of G is contained in exactly one of G_C, and T is a minimum weight spanning tree for G if and only if $T[V(G_C)]$ is a minimum weight spanning tree for G_C for every $C \in \mathfrak{C}(G - \{v\})$.

Now, let $H_C \in \Pi$ be a subgraph of G_C with $w_G(H_C) = \mathsf{opt}(G_C)$ and let H be the subgraph of G obtained by the union of the different H_C's. By Block additivity it holds that $H \in \Pi$, hence $\mathsf{opt}(G) \geq \sum_{C \in \mathfrak{C}(G-\{v\})} \mathsf{opt}(G_C)$. On the other hand, let $H \in \Pi$ be a subgraph of G with $w_G(H) = \mathsf{opt}(G)$ and let $H_C = H[V(G_C)]$. Again, by Block additivity it holds that $H_C \in \Pi$ for every $C \in \mathfrak{C}(G - \{v\})$, hence $\mathsf{opt}(G) \leq \sum_{C \in \mathfrak{C}(G-\{v\})} \mathsf{opt}(G_C)$.

To conclude it is enough to observe that by definition $\mathsf{ex}(G) = \mathsf{opt}(G) - \mathsf{pt}(G)$ and $\mathsf{ex}(G_C) = \mathsf{opt}(G_C) - \mathsf{pt}(G_C)$. □

Lemma 5.6. *Let $T \in \mathcal{G}$ be a weighted tree. Then $T \in \Pi$ and $\mathsf{ex}(T) = \frac{1-\lambda}{2} w_T(E(T))$.*

Proof. It follows from Inclusiveness and Block additivity, since the blocks of $\mathcal{U}(T)$ are all isomorphic to K_2, and $\mathsf{ex}(K_2) = w_{K_2}(E(K_2)) - \mathsf{pt}(K_2) = \frac{1-\lambda}{2} w_{K_2}(E(K_2))$. □

5.1 A polynomial time reduction for $WAPT(\Pi)$

The notion of λ-extendible property does not appear to be strong enough to ensure an FPT-algorithm in general. We consider the following refinement.

Definition 5.7. [72] Let $0 < \lambda < 1$. A graph property $\Pi \subseteq \mathcal{G}$ is *strongly λ-extendible* if it satisfies:

1. Inclusiveness: K_1 and K_2 are in Π,

2. Block additivity: a graph $G \in \mathcal{G}$ is in Π if and only if each block of G is in Π,

3. STRONG λ-SUBGRAPH EXTENSION: given a graph $G = (V, E)$, a weight function $w_G : E \to \mathbb{R}^+$ and a partition U, W of V such that $G[U] \in \Pi$ and $G[W] \in \Pi$, in polynomial time it is possible to find $F \subseteq E(U, W)$ satisfying $w_G(F) \geq \lambda w_G\big(E(U, W)\big)$ and so that the graph $G \setminus \big(E(U, W) \setminus F\big) \in \Pi$.

Essentially, a strongly λ-extendible property is a λ-extendible property which satisfies a strengthened version of λ-EDGE EXTENSION. This does not appear to be a particularly strict requirement, as, for instance, all λ-extendible properties described by Poljak and Turzík [78] can be shown to be strongly λ-extendible [72, 21]. In particular, this is true for WEIGHTED MAX CUT.

Theorem 5.8. *Let $\mathcal{G}_S$ contain all simple graphs, let $\Pi_{BP} \subseteq \mathcal{G}_S$ contain all bipartite graphs and let $\lambda = \frac{1}{2}$. Then Π_{BP} is a strongly λ-extendible property.*

Proof. By Theorem 5.2, we only have to show that STRONG λ-SUBGRAPH EXTENSION holds. Assume $G[U]$ and $G[W]$ are bipartite graphs for a partition U, W of V. Let U_0 and U_1 be a partition of U such that $E(G[U]) = E(U_0, U_1)$, and choose W_0, W_1 in W in the same way. Either let $F = E(U_0, W_1) \cup E(U_1, W_0)$ or $F = E(U_0, W_0) \cup E(U_1, W_1)$: observe that $G \backslash (E(U, W) \backslash F)$ is bipartite in both cases, while in at least one case $w_G(F) \geq \frac{1}{2} w_G(E(U, W))$. $\square$

The aim of this section is to prove that, whenever $\Pi \subseteq \mathcal{G}$ is a strongly λ-extendible property, the WEIGHTED ABOVE POLJAK-TURZÍK (Π) problem, restricted to graphs with integral weights, can be reduced in polynomial time to a restricted problem for which the input has a 'simple' structure.

Definition 5.9. A *uniform clique* in a weighted graph is a clique such that all edges between its vertices have the same weight. A connected weighted graph is a *tree of uniform cliques* if the vertex set of every block is a uniform clique. A weighted graph is a *forest of uniform cliques* if every component is a tree of uniform cliques.

WEIGHTED STRUCTURED ABOVE POLJAK-TURZÍK (Π) (*WSAPT*(Π))	
Input:	A triplet (G, S, k) where $G = (V, E)$ is a connected weighted graph, $S \subseteq V$ contains at most $\frac{6k}{1-\lambda}$ vertices, $G - S$ is a forest of uniform cliques, and k is an integer.
Parameter:	k
Question:	Does it hold that $\mathsf{ex}(G) \geq k$?

The input of the Weighted Structured Above Poljak-Turzík (Π) problem is a graph which contains a small set of vertices whose deletion makes the graph a forest of uniform cliques.

For the rest of this section, let Π denote a generic strongly λ-extendible property on $\mathcal{G}$ and let $G = (V, E) \in \mathcal{G}$ be a connected weighted graph G with weight function $w_G : E \to \mathbb{N}^+$. Let P_3 denote the path with two edges. We say that $v_1, v_2, v_3 \in V$ *form an induced* P_3 if $v_1v_2 \in E$, $v_2v_3 \in E$ and $v_1v_3 \notin E$. If this happens, we say that G contains an induced P_3.

We will use Algorithm 5.1 to compute a small set S such that $G - S$ is a forest of uniform cliques. The rest of this section is devoted to prove the correctness of the algorithm.

Lemma 5.10. *Algorithm 5.1 with input G runs in time polynomial in $|V|$ and outputs a pair (S, t) such that $t \geq \frac{1-\lambda}{6}|S|$ and $\mathsf{ex}(G) \geq t$.*

Proof. Firstly, we will show that the algorithm runs in polynomial time. Note that checking whether Case 1, 2 or 3 applies can be done in time $\mathcal{O}(n^4)$, therefore we are done if we can bound the number of recursive calls. Consider the tree associated to the execution of Algorithm 5.1 on G. Since every time a recursive call is made it is made on a connected graph, this means that every leaf of the tree corresponds either to a graph with at least one edge or to a graph with only one vertex. In addition, note that Case 2 and 3 do not add any edges to the graph and every time Case 1 applies every edge appears in exactly one of the recursive calls. This ensures that the number of leaves is bounded by $\mathcal{O}(n + m) = \mathcal{O}(n^2)$. As for the vertices of degree 2, they correspond to either an application of Case 2 or Case 3: every time a recursive call is made in these cases, at least one edge is deleted, so at most $\mathcal{O}(m)$ of these recursive calls can be made. In total, this gives $\mathcal{O}(n^2)$ vertices in the tree.

The rest of the proof is by induction on $|V|$. First suppose $|V| = 1$. In this case $\textsc{Struct}(G) = (\emptyset, 0)$, then $t \geq \frac{(1-\lambda)}{6}|S|$ trivially holds and $\mathsf{ex}(G) \geq t$ holds by Theorem 5.4. Now suppose that the result holds for any graph with at most n vertices and that $|V| = n + 1$. First of all, note that at every iteration of the algorithm, if we add any vertices to S, we add at most 3 vertices to S and increase t by at least $\frac{(1-\lambda)}{2}$, as the weights are integral. Therefore $t \geq \frac{(1-\lambda)}{6}|S|$ holds.

Concerning the other inequality, note that if $\textsc{Struct}(G) = (\emptyset, 0)$, it is true by Theorem 5.4. Therefore we may assume that one of Cases 1, 2 or 3 applies.

Case 1 (*G contains a cutvertex v*)**:** For each component C of $G - \{v\}$, let $G_C = G[V(C) \cup \{v\}]$ and let $(S_C, t_C) = \textsc{Struct}(G_C)$. By Lemma 5.5 and the inductive hypothesis, $\mathsf{ex}(G) = \sum_{C \in \mathfrak{C}(G-\{v\})} \mathsf{ex}(G_C) \geq \sum_{C \in \mathfrak{C}(G-\{v\})} t_C = t$.

```
input  : A connected weighted graph $G = (V, E) \in \mathcal{G}$
output: A set $S \subseteq V$ and a nonnegative real number $t$
1  Set $S = \emptyset$; Set $t$=0;
2  if $G$ contains a cutvertex $v$ then                                  /* Case 1 */
3      foreach component $C$ of $G - \{v\}$ do
4          Set $(S', t') = \textsc{Struct}(G[V(C) \cup \{v\}])$ ;
5          Set $S = S \cup S'$;
6          Set $t = t + t'$;
7      end
8  else if $\exists\{v_1, v_2, v_3\} \subseteq V$ such that $v_1v_2, v_2v_3 \in E$, $w_G(v_1v_2) > w_G(v_2v_3)$ and
   $G - \{v_1, v_2\}$ is connected then                                  /* Case 2 */
9      Set $(S', t') = \textsc{Struct}(G - \{v_1, v_2\})$;
10     Set $S = S' \cup \{v_1, v_2\}$;
11     Set $t = t' + \frac{(1-\lambda)(w_G(v_1v_2) - w_G(v_2v_3))}{2}$;
12 else if $\exists\{v_1, v_2, v_3, z\} \subseteq V$ such that $v_1, v_2, v_3$ form an induced $P_3$, $G - \{v_1, v_2, v_3\}$ is
   connected, and there exists $xz \in E$, $x \in \{v_1, v_2, v_3\}$, such that $w_G(xz) \leq w_G(v_1v_2)$
   then                                                                  /* Case 3 */
13     Set $(S', t') = \textsc{Struct}(G - \{v_1, v_2, v_3\})$;
14     Set $S = S' \cup \{v_1, v_2, v_3\}$;
15     Set $t = t' + \frac{(1-\lambda)w_G(v_1v_2)}{2}$;
16 end
17 return $(S, t)$;
```

Algorithm 5.1: Struct

Case 2 (*$\exists\{v_1,v_2,v_3\} \subseteq V$ such that $v_1v_2, v_2v_3 \in E$, $w_G(v_1v_2) > w_G(v_2v_3)$ and $G-\{v_1,v_2\}$ is connected*)**:** Let $G' = G - \{v_1,v_2\}$ and $\widetilde{F} = E(\{v_1,v_2\}, V\setminus\{v_1,v_2\})$. Clearly $G[\{v_1,v_2\}] \in \Pi$, as its underlying graph is isomorphic to K_2. Then by λ-EDGE EXTENSION, $\mathsf{opt}(G) \geq w_G(v_1v_2) + \mathsf{opt}(G') + \lambda w_G(\widetilde{F})$. Let $(S',t') = \textsc{Struct}(G')$.

Observe that we can form a spanning tree of G by taking a minimum weight spanning tree of G' and adding the edges v_1v_2, v_2v_3, and therefore $\mathsf{wt}(G) \leq \mathsf{wt}(G') + w_G(v_1v_2) + w_G(v_2v_3)$. Then

$$\begin{aligned}
\mathsf{opt}(G) - t &\geq w_G(v_1v_2) + \mathsf{opt}(G') + \lambda w_G(\widetilde{F}) - t \\
&\geq w_G(v_1v_2) + \lambda w_G(E(G')) + \frac{(1-\lambda)\mathsf{wt}(G')}{2} + t' + \lambda w_G(\widetilde{F}) - t \\
&= \lambda w_G(E) + (1-\lambda)w_G(v_1v_2) + \frac{(1-\lambda)\mathsf{wt}(G')}{2} - \frac{(1-\lambda)(w_G(v_1v_2) - w_G(v_2v_3))}{2} \\
&= \lambda w_G(E) + \frac{(1-\lambda)(\mathsf{wt}(G') + w_G(v_1v_2) + w_G(v_2v_3))}{2} \\
&\geq \lambda w_G(E) + \frac{(1-\lambda)\mathsf{wt}(G)}{2} \\
&= \mathsf{pt}(G).
\end{aligned}$$

Case 3 (*$\exists\{v_1,v_2,v_3,z\} \subseteq V$ such that v_1,v_2,v_3 form an induced P_3, $G-\{v_1,v_2,v_3\}$ is connected, and there exists $xz \in E$, $x \in \{v_1,v_2,v_3\}$, such that $w_G(xz) \leq w_G(v_1v_2)$*)**:** Since we may assume Case 1 and Case 2 do not apply, then $G-\{v_1,v_2\}$ and $G-\{v_2,v_3\}$ are both connected and we have $w_G(v_1v_2) = w_G(v_2v_3) = w_G(xz)$. Let $G' = G-\{v_1,v_2,v_3\}$ and $\widetilde{F} = E(\{v_1,v_2,v_3\}, V\setminus\{v_1,v_2,v_3\})$. By Lemma 5.6, $G[\{v_1,v_2,v_3\}] \in \Pi$. Therefore, by STRONG λ-SUBGRAPH EXTENSION, $\mathsf{opt}(G) \geq 2w_G(v_1v_2) + \mathsf{opt}(G') + \lambda w_G(\widetilde{F})$. Let $(S',t') = \textsc{Struct}(G')$.

Observe that we can form a spanning tree of G by taking a minimum weight spanning tree of G' and adding the edges xz, v_1v_2, v_2v_3, and therefore $\mathsf{wt}(G) \leq \mathsf{wt}(G') + 3w_G(v_1v_2)$. Then,

$$\begin{aligned}
\mathsf{opt}(G) - t &\geq 2w_G(v_1v_2) + \mathsf{opt}(G') + \lambda w_G(\widetilde{F}) - t \\
&\geq 2w_G(v_1v_2) + \lambda w_G(E(G')) + \frac{(1-\lambda)\mathsf{wt}(G')}{2} + t' + \lambda w_G(\widetilde{F}) - t \\
&= \lambda w_G(E) + 2(1-\lambda)w_G(v_1v_2) + \frac{(1-\lambda)\mathsf{wt}(G')}{2} - \frac{(1-\lambda)w_G(v_1v_2)}{2} \\
&\geq \lambda w_G(E) + \frac{(1-\lambda)\mathsf{wt}(G)}{2} \\
&\geq \mathsf{pt}(G).
\end{aligned}$$

□

Lemma 5.11. *Let G be nonempty. Then one of the following holds:*

1. *G contains a cutvertex;*

2. *G contains vertices v_1, v_2, v_3 such that $v_1v_2, v_2v_3 \in E$, $w_G(v_1v_2) > w_G(v_2v_3)$ and $G - \{v_1, v_2\}$ is connected;*

3. *G contains vertices v_1, v_2, v_3 such that v_1, v_2, v_3 form an induced P_3, $G - \{v_1, v_2, v_3\}$ is connected, and there exist $xz \in E$ with $x \in \{v_1, v_2, v_3\}, z \notin \{v_1, v_2, v_3\}$ such that $w_G(xz) \leq w_G(v_1v_2)$;*

4. *V is a uniform clique.*

Proof. We consider the connectivity of G.

G is connected, but not 2-connected: Then G contains a cutvertex and so case 1 holds.

G is 2-connected, but not 3-connected: Let v, w be two vertices such that $G - \{v, w\}$ is disconnected, and observe that v is a cutvertex for $G - \{w\}$. Therefore $G - \{w\}$ has at least two blocks, and in particular at least two pendant blocks. Furthermore, every pendant block must contain an interior vertex adjacent to w, as otherwise G is not 2-connected. So now let v_1, v_3 be vertices such that v_1 and v_3 are interior vertices of different pendant blocks in $G - \{w\}$, and both v_1 and v_3 are adjacent to w. Then observe that v_1, v_2, v_3, with $v_2 = w$, is an induced P_3 and $G - \{v_1, v_2, v_3\}$ is connected.

If there exists an edge between $\{v_1, v_2, v_3\}$ and $V \setminus \{v_1, v_2, v_3\}$ with weight at most $w_G(v_1v_2)$, then Case 3 applies. So now assume that $w_G(xz) > w_G(v_1v_2)$ for all $x \in \{v_1, v_2, v_3\}$ and $z \notin \{v_1, v_2, v_3\}$. We will show that v_1 has a neighbour $u \notin \{v_1, v_2, v_3\}$ such that $G - \{v_1, u\}$ is connected. Since $w_G(v_1u) > w_G(v_1v_2)$, it follows that Case 2 holds. For a generic vertex $y \in V$, let $\mathfrak{R}(y, W)$ be the set of vertices connected to y in the graph $G - W$, for any set of vertices W not containing y. Let us call $u' \in V \setminus \{v_1, v_2, v_3\}$ an *important neighbour* of v_1 if $v_1u' \in E$, and $\mathfrak{R}(v_3, \{v_1, u'\}) \not\subset \mathfrak{R}(v_3, \{v_1, u''\})$ for any $u'' \in V \setminus \{v_1, v_2, v_3\}$ with $v_1u'' \in E$.

Note that v_1 must have a neighbour in $V \setminus \{v_1, v_2, v_3\}$, as otherwise v_2 is a cutvertex for G, a contradiction. So it follows that v_1 has an important neighbour. Let u be an important neighbour of v_1. Note that u is adjacent to a vertex in $\mathfrak{R}(v_3, \{v_1, u\})$, as otherwise v_1 is a cutvertex for G. Suppose $G - \{v_1, u\}$ is not connected, and let C be a component of $G - \{v_1, u\}$ not containing v_3. If C contains a vertex u' adjacent to v_1, then u' is a neighbour of v_1 not in $\{v_1, v_2, v_3\}$ and $\mathfrak{R}(v_3, \{v_1, u'\}) \supseteq \mathfrak{R}(v_3, \{v_1, u\}) \cup \{u\}$, a contradiction as u is an important neighbour of v_1. On the other hand if C contains no vertices adjacent to v_1, then u is a cutvertex of G, a contradiction. So we must have that $G - \{v_1, u\}$ is connected, as required.

G **is 3-connected:** Since deleting any pair of vertices leaves the graph connected, we may assume that every pair of edges that share a vertex have the same weight, as otherwise Case 2 holds. As G is connected, it follows that all edges in G have the same weight.

Observe that if G is complete, then V is a uniform clique, and so Case 4 holds. Thus we may assume G is not complete, and so contains an induced P_3. We will show that Case 3 holds.

Let v be an arbitrary vertex in V. If for any v_1, v_2, v_3 which form an induced P_3 it holds that $v \in \{v_1, v_2, v_3\}$, then $G - \{v\}$ contains no induced P_3, and so $V \setminus \{v\}$ must be a clique: therefore, $G - \{v_1, v_2, v_3\}$ is connected for any v_1, v_2, v_3 which form an induced P_3.

Suppose this is not the case. We say that a set of vertices $\{v_1, v_2, v_3\}$ is an *important* P_3 if v_1, v_2, v_3 form an induced P_3 and $\mathfrak{R}(v, \{v_1, v_2, v_3\}) \not\subset \mathfrak{R}(v, \{v_1', v_2', v_3'\})$, for any $v_1', v_2', v_3' \in V$ which form an induced P_3. Assume $\{v_1, v_2, v_3\}$ is an important P_3 and $v \notin \{v_1, v_2, v_3\}$. If $\mathfrak{R}(v, \{v_1, v_2, v_3\}) = V \setminus \{v_1, v_2, v_3\}$ then $G - \{v_1, v_2, v_3\}$ is connected. Otherwise, let C be a component of $G - \{v_1, v_2, v_3\}$ not containing v. Since G is 3-connected both $\mathfrak{R}(v, \{v_1, v_2, v_3\})$ and C must have vertices adjacent to each of v_1, v_2, v_3. Therefore there must be a path $u_0u_1 \dots u_l$, where $u_0 = v_1, u_l = v_3$, and $u_i \in C$ for all $1 \leq i < l$. By taking a shortest such path, and considering three consecutive vertices v_1', v_2', v_3' on this path, we have that v_1', v_2', v_3' induce a P_3 such that $\mathfrak{R}(v, \{v_1', v_2', v_3'\}) \supseteq \mathfrak{R}(v, \{v_1, v_2, v_3\}) \cup \{v_2\}$, a contradiction as $\{v_1, v_2, v_3\}$ is an important P_3.

Thus $G - \{v_1, v_2, v_3\}$ is connected, as required. As all weights are equal, Case 3 holds. □

Lemma 5.12. *Let G be nonempty and let $(S, t) = \textsc{Struct}(G)$. Then $G - S$ is a forest of uniform cliques.*

Proof. In what follows, it will be useful to note that if $G' - S'$ is a forest of uniform cliques for some subgraph G' of G and $S' \subseteq S$, then $G' - S$ is also a forest of uniform cliques.

We prove the claim by induction on $|V|$. Observe that if $|V| = 1$ then $G - S$ contains at most one vertex and is therefore a forest of uniform cliques. So now suppose that $|V| = n$ and the result is true for smaller graphs. Consider the three cases of the `if` statement of Algorithm 5.1:

Case 1 (*G contains a cutvertex v*)**:** For each component C of $G - \{v\}$, let $G_C = G[V(C) \cup \{v\}]$ and let $(S_C, t_C) = \textsc{Struct}(G_C)$. By the inductive hypothesis, $G_C - S_C$ is a forest of uniform cliques and so $G_C - S$ is also a forest of uniform cliques, where $S = \cup_{C \in \mathfrak{C}(G-\{v\})} S_C$. Since $G - S$ is formed either by taking the disjoint union of all $G_C - S$ (if $v \in S$), or by joining all $G_C - S$ at a single vertex (if $v \notin S$), it follows that $G - S$ is also a forest of uniform cliques.

Case 2 *($\exists\{v_1, v_2, v_3\} \subseteq V$ such that $v_1v_2, v_2v_3 \in E$, $w_G(v_1v_2) > w_G(v_2v_3)$ and $G - \{v_1, v_2\}$ is connected)***:** Let $G' = G - \{v_1, v_2\}$ and let $(S', t') = \textsc{Struct}(G')$. Then since $S = S' \cup \{v_1, v_2\}$, $G - S = G' - S'$, which is a forest of uniform cliques by the inductive hypothesis.

Case 3 *($\exists\{v_1, v_2, v_3, z\} \subseteq V$ such that v_1, v_2, v_3 form an induced P_3, $G - \{v_1, v_2, v_3\}$ is connected, and there exists $xz \in E$, $x \in \{v_1, v_2, v_3\}$, such that $w_G(xz) \leq w_G(v_1v_2)$)***:** Let $G' = G - \{v_1, v_2, v_3\}$ and let $(S', t') = \textsc{Struct}(G')$. Then since $S = S' \cup \{v_1, v_2, v_3\}$, $G - S = G' - S'$, which is a forest of uniform cliques by the inductive hypothesis.

Finally, if none of the previous cases apply, then, by Lemma 5.11, V is a uniform clique, hence G is a tree of uniform cliques containing only one block. □

Theorem 5.13. *Let (G, k) be an instance of $WAPT(\Pi)$ such that the weight function of G takes values in $\mathbb{N}^+$. In polynomial time we can either answer* `YES` *or compute a set $S \subseteq V$ such that $|S| \leq \frac{6k}{1-\lambda}$ and $G - S$ is a forest of uniform cliques.*

Proof. Run Algorithm 5.1 on G and let (S, t) be the output. By Lemma 5.10, $\mathsf{ex}(G) \geq t$. Hence if $k \leq t$, the instance is a `YES`-instance. Otherwise, by Lemma 5.10, $k > t \geq \frac{1-\lambda}{6}|S|$. Hence $|S| < \frac{6k}{1-\lambda}$. In addition, by Lemma 5.12, $G - S$ is a forest of uniform cliques. □

From Theorem 5.13, by using the fact that Algorithm 5.1 does not modify the graph, the following corollary immediately follows.

Corollary 5.14. *$WAPT(\Pi)$ restricted to graphs with integral weights can be reduced in polynomial time to $WSAPT(\Pi)$, and the reduction preserves the weights of the edges.*

5.1.1 Weighted Max Cut

Unfortunately, Corollary 5.14 is not enough in itself to ensure the existence of an *FPT*-algorithm. Nonetheless, it is generally easier to prove that $WSAPT(\Pi)$ is *FPT*, or admits a polynomial kernel, than to prove it for $WAPT(\Pi)$. As an example, we consider Weighted Max Cut restricted to graphs with integral weights.

Weighted Max Cut APT

Input:	A pair (G, k) where $G = (V, E)$ is a connected weighted graph with weight function $w_G : E \to \mathbb{N}^+$, and k is an integer.
Parameter:	k
Question:	Is there a bipartite subgraph H of G such that $w_G(E(H)) \geq \mathsf{pt}(G) + k$?

The result of this section is based on the work of Crowston *et al.* [23]. Also, note that a proof which uses the same approach has already been given by Jones [60].

Consider first the following auxiliary problem.

MAX CUT WITH WEIGHTED VERTICES	
Input:	A quadruplet (G, w_0, w_1, p) where $G = (V, E)$ is a weighted graph with weight function $w_G : E \to \mathbb{N}^+$, w_0 and w_1 are functions from V to $\mathbb{N}$, and p is an integer.
Question:	Is there an assignment $f : V \to \{0, 1\}$ such that $\sum_{uv \in E} \lvert f(u) - f(v) \rvert w_G(uv) + \sum_{f(v)=0} w_0(v) + \sum_{f(v)=1} w_1(v) \geq p$?

Lemma 5.15. *The* MAX CUT WITH WEIGHTED VERTICES *problem can be solved in polynomial time if G is a forest of uniform cliques.*

Proof. We provide a polynomial time transformation that replaces an instance (G, w_0, w_1, p) with an equivalent instance (G', w_0', w_1', p') such that G' has fewer vertices than G. By applying the transformation at most $|V|$ times to get a trivial instance, we have a polynomial time algorithm to decide (G, w_0, w_1, p).

Let B be a pendant block of G and let u be its root, unless G consists of a single block, in which case let u be an arbitrary vertex and $B = G$. Let $C = B - \{u\}$. Recall that by definition of forest of uniform cliques, $V(B)$ is a uniform clique. For each possible assignment to u, we will in polynomial time calculate the optimal extension to the vertices in C (this optimal extension depends only on the assignment to u, since no other vertices are adjacent to vertices in C). We can then delete all the vertices in C, and change the values of $w_0(u)$ and $w_1(u)$ to reflect the optimal extension for each assignment.

Suppose we assign u the value 1. Let $\varepsilon(v) = w_1(v) - w_0(v)$ for each $v \in V(C)$. Now arrange the vertices of C in order $v_1, v_2, \ldots v_{n'}$ (where $n' = |V(C)|$), such that if $i < j$ then $\varepsilon(v_i) \geq \varepsilon(v_j)$. We claim that there is an optimal assignment for which v_i is assigned 1 for every $i \leq t$, and v_i is assigned 0 for every $i > t$, for some $0 \leq t \leq n'$. In fact, consider an assignment for which $f(v_i) = 0$ and $f(v_j) = 1$, for $i < j$, and observe that switching the assignments of v_i and v_j will increase $\sum_{f(v)=0} w_0(v) + \sum_{f(v)=1} w_1(v)$ by $\varepsilon(v_i) - \varepsilon(v_j)$, which is a nonnegative quantity. At the same time, $\sum_{vw \in E} \lvert f(v) - f(w) \rvert w_G(vw)$ does not change, as the edges between vertices in C all have the same weight.

Therefore the claim holds and we only need to try $n' + 1$ different assignments to the vertices in C in order to find the optimal one for $f(u) = 1$. Let w_1 be

$$\sum_{vw \in E(B)} |f(v) - f(w)| w_G(vw) + \sum_{v \in V(B): f(v)=0} w_0(v) + \sum_{v \in V(B): f(v)=1} w_1(v)$$

where f is the optimal assignment computed in this way.

By a similar method we can find the optimal assignment when u is assigned 0. Let w_0 be its value. Now we can delete the vertices in C and let $w_1(u) = w_1$ and $w_0(u) = w_0$. $\square$

Theorem 5.16. *Let $\mathcal{G_S}$ contain all simple graphs, let $\Pi_{BP} \subseteq \mathcal{G_S}$ contain all bipartite graphs and let $\lambda = \frac{1}{2}$. Then the* Weighted Structured Above Poljak-Turzík *(Π_{BP}) problem restricted to graphs with integral weights is FPT.*

Proof. Let (G, S, k) be an instance of Weighted Structured Above Poljak-Turzík (Π_{BP}), with $G = (V, E)$ and weight function $w_G : E \to \mathbb{N}^+$. We will show that every partition V_0^S, V_1^S of S can be optimally extended in polynomial time to a partition V_0, V_1 of V. Let V_0^S, V_1^S be any such partition. For every vertex $v \in V \setminus S$, let $w_0(v) = \sum_{s \in (V_1^S \cap N(v))} w_G(vs)$ and $w_1(v) = \sum_{s \in (V_0^S \cap N(v))} w_G(vs)$, then delete S and let $G' = G - S$.

Let l be the weight of the edges in $E(V_0^S, V_1^S)$ and let $p = \mathsf{pt}(G) + k - l$. Note that (G', w_0, w_1, p) is an instance of Max Cut with Weighted Vertices. For an assignment to the vertices of $G - S$, the total weight of edges in G with one endvertex assigned 0 and the other 1 would be exactly $\sum_{uv \in E(G-S)} |f(u) - f(v)| w_G(uv) + \sum_{f(v)=0} w_0(v) + \sum_{f(v)=1} w_1(v) + l$. Thus, V_0^S, V_1^S can be extended to a partition V_0, V_1 of V such that $w_G(E(V_0, V_1)) \geq \mathsf{pt}(G) + k$ if and only if (G', w_0, w_1, p) is a `YES`-instance. By Lemma 5.15, the latter problem can be solved in polynomial time.

Since there exist at most $2^{\mathcal{O}(k)}$ different partitions of S, we conclude that the problem is FPT. $\square$

Corollary 5.17. Weighted Max Cut APT *is FPT.*

Proof. Note that Weighted Max Cut APT is equivalent to the Weighted Above Poljak-Turzík (Π_{BP}) problem on graphs with integral weights. Let (G, k) be an instance of the latter. By Theorem 5.8 and Theorem 5.13, in polynomial time we can either answer `YES` or transform it into an equivalent instance (G, S, k) of Weighted Structured Above Poljak-Turzík (Π_{BP}), which by Theorem 5.16 can be solved in $2^{\mathcal{O}(k)} |V|^{\mathcal{O}(1)}$ time. $\square$

5.2 Polynomial kernel for $APT(\Pi)$

We restrict the attention to unweighted graphs, that is, graphs where every edge has weight one. We will prove that for (nearly) every strongly λ-extendible property Π, the ABOVE POLJAK-TURZÍK (Π) problem admits a polynomial kernel.

We restate the definition of the problem:

ABOVE POLJAK-TURZÍK (Π) ($APT(\Pi)$)	
Input:	A pair (G, k) where $G = (V, E)$ is a connected graph, and k is an integer.
Parameter:	k
Question:	Does it hold that $\mathsf{ex}(G) \geq k$?

By Corollary 5.14, though, the ABOVE POLJAK-TURZÍK (Π) problem can be reduced in polynomial time to the the STRUCTURED ABOVE POLJAK-TURZÍK (Π) problem. Note that we simply call *forest of cliques* an unweighted forest of uniform cliques.

STRUCTURED ABOVE POLJAK-TURZÍK (Π) ($SAPT(\Pi)$)	
Input:	A triplet (G, S, k) where $G = (V, E)$ is a connected graph, $S \subseteq V$ contains at most $\frac{6k}{1-\lambda}$ vertices, $G - S$ is a forest of cliques, and k is an integer.
Parameter:	k
Question:	Does it hold that $\mathsf{ex}(G) \geq k$?

Hence, we may assume that we are given a set S of $\mathcal{O}(k)$ vertices such that $G - S$ is a forest of cliques, and we only need to find a bound on the number of vertices in $G - S$.

Observe that so far we have not used the STRONG λ-SUBGRAPH EXTENSION hypothesis heavily: in fact, we only used it when $\mathcal{U}(G[U])$ is isomorphic to K_2 or P_3. The situation will change here, as the next lemma will be one of the most useful tools to prove the existence of a kernel, and its validity depends on the fact that $\mathcal{U}(G[U])$ can be any graph in Π.

Lemma 5.18. *Let $G = (V, E) \in \mathcal{G}$ be a connected graph, and let V_1, V_2 be a partition of V. Let c_1 be the number of components of $G[V_1]$ and c_2 be the number of components of $G[V_2]$. If $\mathsf{ex}(G[V_1]) \geq k_1$ and $\mathsf{ex}(G[V_2]) \geq k_2$, then $\mathsf{ex}(G) \geq k_1 + k_2 - \frac{1-\lambda}{2}(c_1 + c_2 - 1)$.*

Proof. Let $\widetilde{F} = E(V_1, V_2)$. Using STRONG λ-SUBGRAPH EXTENSION it is not difficult to see that $\mathsf{opt}(G) \geq \mathsf{opt}(G[V_1]) + \mathsf{opt}(G[V_2]) + \lambda|\widetilde{F}|$. Moreover,

$$\begin{aligned}
\mathsf{pt}(G) &= \lambda|E| + \frac{1-\lambda}{2}(|V|-1) \\
&= \lambda(|E(G[V_1])| + |E(G[V_2])| + |\widetilde{F}|) + \frac{1-\lambda}{2}(|V_1|+|V_2|-1) \\
&= \Big(\lambda|E(G[V_1])| + \frac{1-\lambda}{2}(|V_1|-c_1)\Big) + \Big(\lambda|E(G[V_2])| + \frac{1-\lambda}{2}(|V_2|-c_2)\Big) \\
&+ \lambda|\widetilde{F}| + \frac{1-\lambda}{2}(c_1+c_2-1) \\
&= \mathsf{pt}(G[V_1]) + \mathsf{pt}(G[V_2]) + \lambda|\widetilde{F}| + \frac{1-\lambda}{2}(c_1+c_2-1).
\end{aligned}$$

It follows that $\mathsf{ex}(G) \geq k_1 + k_2 - \frac{1-\lambda}{2}(c_1+c_2-1)$. □

The definition of strongly λ-extendible property is quite generic and includes many different properties, but a possible rough classification consists of distinguishing between properties which *diverge* and properties which do not. Intuitively, a property diverges when the Poljak-Turzík bound is not tight on complete graphs, that is, the size of an optimal solution increases faster than the Poljak-Turzík bound when we consider complete graphs of increasing size. We will see that when a property diverges it is easy to produce a polynomial kernel, and that this happens in most of the cases.

Let $\mathsf{ex}(K_j)$ denote the minimum value of $\mathsf{ex}(G)$ for any (oriented, labelled) graph G such that $K_j = \mathcal{U}(G)$. Thus, for example, if $\mathsf{ex}(K_3) = t$ then any graph G with underlying graph K_3 has a subgraph $H \in \Pi$ with at least $\mathsf{pt}(G)+t$ edges, regardless of orientations or labellings on the edges of G.

Definition 5.19. A strongly λ-extendible property Π diverges on cliques if there exists $j \in \mathbb{N}^+$ such that $\mathsf{ex}(K_j) > \frac{1-\lambda}{2}$.

Lemma 5.20. *Let Π be a strongly λ-extendible property which diverges on cliques, and let j, a be such that $\mathsf{ex}(K_j) = \frac{1-\lambda}{2} + a$, $a > 0$. Then $\mathsf{ex}(K_{rj}) \geq \frac{1-\lambda}{2} + ra$ for each $r \in \mathbb{N}^+$. Furthermore, $\lim_{s\to+\infty} \mathsf{ex}(K_s) = +\infty$.*

Lemma 5.20 formalises the intuition behind the notion of diverging properties. To prove it, we need an auxiliary lemma, but we want to prove a slightly more general version of this lemma, as it will be useful later. We need first the following definitions. We say that a simple connected graph $\widetilde{K}$ is *almost-complete* if it is either complete or it can be made complete by deleting one vertex. For an almost-complete graph $\widetilde{K}$, we use $\mathsf{ex}(\widetilde{K})$ to denote the minimum value of $\mathsf{ex}(G)$ for any (oriented, labelled) graph G such that $\widetilde{K} = \mathcal{U}(G)$, and we say that $\widetilde{K} \in \Pi$ if and only if $G \in \Pi$ for every (oriented, labelled) graph G with underlying graph $\widetilde{K}$.

Lemma 5.21. *Let* $\mathsf{ex}(K_j) = a \geq \frac{1-\lambda}{2}$ *for some* $j \in \mathbb{N}$*. Then, for every almost-complete graph* $\widetilde{K}$ *with at least* $j+1$ *vertices,* $\mathsf{ex}(\widetilde{K}) \geq a - \frac{1-\lambda}{2}$*.*

Proof. Let $G = (V, E) \in \mathcal{G}$ be a graph such that $\mathcal{U}(G) = \widetilde{K}$, where $\widetilde{K}$ is an almost-complete graph with at least $j+1$ vertices. Let V' be a minimum-sized subset of V such that $G - V'$ is a complete graph. Set V_1 to be any subset of exactly $|V| - j$ vertices of G such that $V' \subseteq V_1$ and $G[V_1]$ is connected. Set $V_2 = V \setminus V_1$. Observe that G is connected, V_1, V_2 is a partition of V, $G[V_1]$ is connected, and $\mathcal{U}(G[V_2]) = K_j$. Furthermore, $\mathsf{ex}(G[V_1])$ is obviously at least 0, and $\mathsf{ex}(G[V_2])$ is at least a by assumption. So by Lemma 5.18, we get that $\mathsf{ex}(G) \geq a - \frac{1-\lambda}{2}$. □

In particular, Lemma 5.21 holds when $\widetilde{K}$ is a complete graph. We are now ready to prove Lemma 5.20.

Proof of Lemma 5.20. We prove the first part of the lemma by induction on r. The claim holds for $r = 1$ by assumption. Suppose that the claim holds for some $r \geq 1$. We show that it holds for $r + 1$ as well. Let $G = (V, E) \in \mathcal{G}$ be a graph such that $\mathcal{U}(G)$ is isomorphic to $K_{(r+1)j}$, and consider a partition of V into two parts V_1, V_2 with $|V_1| = j, |V_2| = rj$. Note that $\mathcal{U}(G[V_1])$ is isomorphic to K_j and $\mathcal{U}(G[V_2])$ is isomorphic to K_{rj}. By assumption we have that $\mathsf{ex}(G[V_1]) \geq \frac{1-\lambda}{2} + a$, and from the induction hypothesis we get that $\mathsf{ex}(G[V_2]) \geq \frac{1-\lambda}{2} + ra$. Lemma 5.18 now tells us that $\mathsf{ex}(G) \geq \frac{1-\lambda}{2} + (r+1)a$, and this completes the induction step.

Now consider the function $f : \mathbb{N}^+ \to \mathbb{R}^+$ defined as $f(r) = \mathsf{ex}(K_{rj})$. Our arguments above show also that f is an *unbounded* function. We use this to argue that given any $x \in \mathbb{R}^+$, there is an $r_x \in \mathbb{N}^+$ such that $\mathsf{ex}(K_r) > x$ for all $r \geq r_x$; this would prove the second part of the lemma. So let $x \in \mathbb{R}^+$. We choose $p \in \mathbb{N}^+$ such that $f(p) = \mathsf{ex}(K_{pj}) > x + \frac{1-\lambda}{2}$. Since f is unbounded, such a choice of p exists. We set $r_x = pj$, and from Lemma 5.21 we get that $\mathsf{ex}(K_r) > x$ for all $r \geq r_x$. □

5.2.1 Diverging properties

Let (G, S, k) be an instance of the Structured Above Poljak-Turzík (Π) problem, with $G = (V, E)$. Let Q be the set of cutvertices of $G - S$. For any block B of $G - S$, let $B_{\mathsf{int}} = V(B) \setminus Q$ be the *interior* of B. Let $\mathcal{B}$ be the set of blocks of $G - S$, that is, $\mathcal{B} = \mathfrak{B}(G - S)$. A *block neighbour* of a block B is a block B' different from B such that $|V(B) \cap V(B')| = 1$. Given a sequence of blocks $B_0, B_1, \ldots, B_l, B_{l+1}$ in $G - S$, the subgraph induced by $V(B_1) \cup \cdots \cup V(B_l)$ is a *block path* if, for every $1 \leq i \leq l$, $V(B_i)$ contains exactly two vertices from Q, and B_i has exactly two block neighbours B_{i-1} and B_{i+1}. A block B in

$G-S$ is a *leaf block* if $V(B)$ contains exactly one vertex from Q, which is called its *root* [2] . A block in $G-S$ is an *isolated block* if it contains no vertex from Q. Observe that an isolated block has no block neighbour, while a leaf block has *at least* one block neighbour.

Let $\mathcal{B}_0$ and $\mathcal{B}_1$ be the set of isolated blocks and leaf blocks, respectively, contained in $\mathcal{B}$. Let $\mathcal{B}_2$ be the set of blocks $B \in \mathcal{B}$ such that B is a block in some block path of $G-S$. Finally, let $\mathcal{B}_{\geq 3} = \mathcal{B} \setminus (\mathcal{B}_0 \cup \mathcal{B}_1 \cup \mathcal{B}_2)$. Thus:

- $\mathcal{B}_0$ is the set of all blocks of $G-S$ which contain no cutvertex of $G-S$, and therefore have no block neighbour;
- $\mathcal{B}_1$ is the set of all blocks of $G-S$ which contain exactly one cutvertex of $G-S$, and therefore have at least one block neighbour;
- $\mathcal{B}_2$ is the set of all blocks of $G-S$ which contain exactly two cutvertices of $G-S$, *and* have exactly two block neighbours;
- $\mathcal{B}_{\geq 3}$ is the set of all the remaining blocks of $G-S$. A block of $G-S$ is in $\mathcal{B}_{\geq 3}$ if and only if it contains at least two cutvertices of $G-S$, *and* has at least three block neighbours.

In order to bound the number of vertices in $G-S$ it is enough to bound the number of blocks and the size of each block. For the rest of this section we will assume that Π is a strongly λ-extendible property that diverges on cliques.

Definition 5.22. Let v be a cutvertex of G and let C be a component of $G - \{v\}$ such that $G[V(C) \cup \{v\}]$ is a 2-connected almost-complete subgraph of G. Then we say that $G[V(C) \cup \{v\}]$ is a *dangling component* with root v.

To bound the number of isolated and leaf blocks in $G-S$, we require the following reduction rule.

Reduction Rule 5.23. *Let $G \in \mathcal{G}$ be a connected graph which is not 2-connected and let G' be a dangling component. Then if $\mathsf{ex}(G') = 0$, delete $G' - \{v\}$ (where v is the root of G') and leave k the same.*

Lemma 5.24. *Reduction Rule 5.23 is valid.*

Proof. Let G'' be the graph obtained after an application of the rule. By Lemma 5.5, $\mathsf{ex}(G) = \mathsf{ex}(G'') + \mathsf{ex}(G') = \mathsf{ex}(G'')$. Now, observe that in polynomial time it is possible to find the

[2]Note that a leaf block of G is simply a pendant block of $G-S$. For ease of discussion, though, a different name is used.

block decomposition of G and to check which blocks have an underlying graph which is almost-complete. Thus, in polynomial time we can find all dangling components. Now, we claim that in constant time it is possible to evaluate whether their excess is zero. In fact, it holds that $\mathsf{ex}(K_j) > \frac{1-\lambda}{2}$ for some j. Given a graph G' whose underlying graph is almost-complete, if G' has at least $j+1$ vertices Lemma 5.21 ensures that $\mathsf{ex}(G') > 0$. On the other hand, if G' has at most j vertices, a brute force algorithm which finds $\mathsf{opt}(G')$ runs in time $\mathcal{O}(2^{j^2})$, where j is a constant which only depends on Π. $\square$

Observe that every time Reduction Rule 5.23 is applied the number of vertices decreases, hence in polynomial time we can produce a graph which is reduced under this rule. In such a graph, the excess of every dangling component is strictly positive. We are interested in the *infimum* of these values.

Definition 5.25. We use AK^+ to denote the class of all graphs $G' \in \mathcal{G}$ such that $\mathcal{U}(G')$ is almost-complete and $\mathsf{ex}(G') > 0$. Let $\inf_{AK}$ denote the value $\inf_{(G \in AK^+)} \mathsf{ex}(G')$.

Note that the class AK^+ contains an infinite number of graphs in general. Hence, it could be the case that $\inf_{AK} = 0$. Nonetheless, next lemma shows that this is not the case.

Lemma 5.26. *The infimum $\inf_{AK}$ is strictly greater than 0.*

Proof. Since Π diverges on cliques, there exist $j \in \mathbb{N}^+$ and $a \in \mathbb{R}^+$ such that $\mathsf{ex}(K_j) = \frac{1-\lambda}{2} + a$. Then, by Lemma 5.21, for every graph $G' \in AK^+$ with at least $j+1$ vertices, $\mathsf{ex}(G') \geq a$. Now observe that $\{G' \in AK^+ : |V(G')| \leq j\}$ is a finite set (recall that the number of labels, if there are any, is finite), hence the minimum of $\mathsf{ex}(G')$ over this set is defined and is positive. So we have that $\inf_{AK} \geq \min(a, \min_{\{G' \in AK^+ : |V(G')| \leq j\}} \mathsf{ex}(G')) > 0$. $\square$

Now, we are able to bound the number of dangling components.

Lemma 5.27. *Let G be a connected graph reduced under Reduction Rule 5.23. Then the number of dangling components is bounded by $b_0 k$ (where b_0 is a constant depending only on Π) or the instance is a* YES*-instance.*

Proof. Let $B_1, \ldots, B_l$ be the dangling components of G. Since the graph is reduced under Reduction Rule 5.23, $\mathsf{ex}(B_i) > 0$ for every $1 \leq i \leq l$. In addition, Lemma 5.26 ensures that $\inf_{AK} > 0$. Let $G' = G - (\cup_{i=1}^{l}((B_i)_{\mathsf{int}})$.

By Lemma 5.5, $\mathsf{ex}(G) = \mathsf{ex}(G') + \sum_{i=1}^{l} \mathsf{ex}(B_i) \geq 0 + (\inf_{AK})l$. Then if $l \geq \frac{k}{\inf_{AK}}$ the instance is a YES-instance. Otherwise, $l \leq \frac{k}{\inf_{AK}} = b_0 k$. $\square$

Note that an isolated block always contains a neighbour of S, as G is connected, and a leaf block which does not contain a neighbour of S in its interior is a dangling component, whose number is bounded. Therefore, we now want to prove a bound on the number of neighbours that vertices in S can have in the interiors of blocks of $G - S$. This will lead to a bound on $|\mathcal{B}_0| + |\mathcal{B}_1|$ and on the number of connected components of $G - S$.

Theorem 5.28. *Let G be a connected graph reduced under Reduction Rule 5.23. If there exists $s \in S$ such that $\sum_{B\in\mathcal{B}} |N_G(B_{\mathsf{int}}) \cap \{s\}|$ is at least $(\frac{16}{1-\lambda} + \frac{2}{\inf_{AK}})k - 2$, then the instance is a* YES*-instance.*

Proof. Let $U = \{s\}$. For every block B of $G - S$ such that $|N_G(B_{\mathsf{int}}) \cap \{s\}| = 1$, pick a vertex in $N(s) \cap B_{\mathsf{int}}$ and add it to U. Since the vertices are chosen in the interior of different blocks of $G - S$, $G[U]$ is a tree and, therefore, by Lemma 5.6 it is in Π and $\mathsf{ex}(G[U]) = \frac{1-\lambda}{2}d$, where d is the degree of s in $G[U]$. Let c be the number of components of $G - U$, and assume that U is constructed such that d is maximal and c is minimal. By Lemma 5.18, $\mathsf{ex}(G) \geq \frac{1-\lambda}{2}(d - c)$.

We will now show that c is bounded. The number of components of $G - U$ which contain a vertex of $S \setminus \{s\}$ is bounded by $(|S| - 1) < \frac{6k}{1-\lambda} - 1$. In addition, the number of components of $G - U$ which contain at least two blocks from which a vertex has been added to U is at most $\frac{d}{2}$.

Now, let C be a component of $G - S$ such that, in the graph G, no vertex in $C - U$ has a neighbour in $S \setminus \{s\}$ and $|U \cap V(C)| = 1$. Firstly, suppose that C contains only one block B of $G - S$. Let $\{v\} = U \cap V(C)$. Note that, by the current assumptions, $N(S \setminus \{s\}) \cap V(C) \subseteq \{v\}$. If v is the only neighbour of s in C, then it is a cutvertex in G, hence B is a dangling component of G. If s has another neighbour v' in C and v has no neighbour in S different from s, then s is a cutvertex, therefore $G[V(B) \cup \{s\}]$ is a dangling component. Finally, if v has at least two neighbours in S and s has at least another neighbour v' in C, let U' be the set obtained by replacing v with v' in U, and observe that C is connected to $S \setminus \{s\}$ in $G - U'$, contradicting the minimality of c.

Now, suppose that C contains at least two blocks of $G - S$. In this case, every block except one block B does not contain neighbours of S. In particular, this holds for at least one leaf block B' in C. Hence, B' is a dangling component.

This ensures that carefully choosing the vertices of U we may assume that any component of $G - U$ still contains a neighbour of $S \setminus \{s\}$, or contains at least two blocks from which a vertex of U has been chosen, or contains part of a dangling component. Hence, the number of components of $G - U$ is at most $\frac{6k}{1-\lambda} - 1 + \frac{d}{2} + \frac{k}{\inf_{AK}}$.

Therefore, if $d \geq (\frac{16}{1-\lambda} + \frac{2}{\inf_{AK}})k - 2$, then $\mathsf{ex}(G) \geq k$. □

Corollary 5.29. *Let G be a connected graph reduced under Reduction Rule 5.23. Then $\sum_{s \in S} \sum_{B \in \mathcal{B}} |N_G(B_{\mathsf{int}}) \cap \{s\}|$ is bounded by $b_1 k^2$ (where b_1 is a constant depending only on Π), or the instance is a* `YES`*-instance.*

Proof. It follows from Theorem 5.28 and from the fact that $|S| \leq \frac{6k}{1-\lambda}$. □

Corollary 5.30. *Let G be a connected graph reduced under Reduction Rule 5.23. Then $|\mathcal{B}_0| + |\mathcal{B}_1|$ is bounded by $(b_0 + b_1)k^2$ (where b_0 and b_1 are as in Lemma 5.27 and Corollary 5.29 respectively), or the instance is a* `YES`*-instance.*

Proof. Note that every isolated or leaf block either has a neighbour of S in its interior or is a dangling component. The claim follows from Lemma 5.27 and Corollary 5.29. □

Corollary 5.31. *Let G be a connected graph reduced under Reduction Rule 5.23. Then either $G - S$ has at most $(b_0 + b_1)k^2$ components (where b_0 and b_1 are as in Lemma 5.27 and Corollary 5.29 respectively), or the instance is a* `YES`*-instance.*

Proof. Since a component of $G - S$ contains at least one block from $\mathcal{B}_0 \cup \mathcal{B}_1$, the result follows applying Corollary 5.30. □

The bound on the number of components of $G - S$ allows us to prove a bound on the number of blocks in $\mathcal{B}$ which have positive excess, as the next lemma shows.

Lemma 5.32. *Let G be a connected graph reduced under Reduction Rule 5.23. The blocks with positive excess of $G - S$ are at most $b_2 k^2$ (where b_2 is a constant depending only on Π), or the instance is a* `YES`*-instance.*

Proof. Let p be the number of blocks in $G - S$ with positive excess, and let G' be the union of all components of $G - S$ which contain a block with positive excess. Observe that by Corollary 5.31, we may assume that G' has at most $(b_0 + b_1)k^2$ components. Observe that by repeated use of Lemma 5.5, $\mathsf{ex}(G') \geq (\inf_{AK})p$. Now let $G'' = G - G'$, and observe that G'' has at most $|S| \leq \frac{6k}{1-\lambda}$ components. Then by Lemma 5.18, $\mathsf{ex}(G) \geq (\inf_{AK})p - \frac{1-\lambda}{2}((b_0 + b_1)k^2 + \frac{6k}{1-\lambda} - 1)$. Therefore if $p \geq \frac{1}{\inf_{AK}}\left(\frac{1-\lambda}{2}((b_0 + b_1)k^2 + \frac{6k}{1-\lambda} - 1) + k\right) = b_2 k^2$, the instance is a `YES`-instance. □

Now, we show that an instance which contains 'large' blocks is a `YES`-instance.

Lemma 5.33. *Let j, a be such that $\mathsf{ex}(K_j) = \frac{1-\lambda}{2} + a$, $a > 0$. If $|V(B)| \geq \lceil \frac{4k}{a} \rceil j = b_3 k$ for any block B of $G - S$, then the instance is a* `YES`*-instance.*

Proof. Let C be the component of $G - S$ containing B. Note that $G - V(C)$ has at most $|S| \leq \frac{6k}{1-\lambda}$ components, since every component of $G-S$ which does not contain B still contains a neighbour of S. In addition, by repeated use of Lemma 5.5, we get that $\mathsf{ex}(C) \geq \mathsf{ex}(B)$. Therefore, if $\mathsf{ex}(B) \geq \frac{1-\lambda}{2}\frac{6k}{1-\lambda} + k = 4k$, Lemma 5.18 ensures that we have a `YES`-instance.

By Lemma 5.20, if r is an integer such that $r \geq \lceil \frac{4k}{a} \rceil$, then $\mathsf{ex}(K_{rj}) \geq 4k + \frac{1-\lambda}{2}$. Furthermore, by Lemma 5.21, if $|V(B)| \geq rj$ then $\mathsf{ex}(B) \geq 4k$. Thus, if $|V(B)| \geq \lceil \frac{4k}{a} \rceil j$ we have a `YES`-instance. □

At this point, we are able to prove that a restricted case of the Above Poljak-Turzík (Π) problem admits a polynomial kernel. Note that this theorem is just an auxiliary result which is needed to prove the existence of a kernelization in the general case.

Theorem 5.34. *Let Π be a strongly λ-extendible property which diverges on cliques, and suppose $\mathsf{ex}(K_i) > 0$ for all $i \geq 2$. Then the* Above Poljak-Turzík (Π) *problem admits a kernel with at most $\mathcal{O}(k^2)$ vertices.*

Proof. By Corollary 5.14, $APT(\Pi)$ can be reduced in polynomial time to $SAPT(\Pi)$. Then it is enough to show that $SAPT(\Pi)$ admits a kernel with at most $\mathcal{O}(k^2)$ vertices. Let (G, S, k) be an instance of this problem and assume that G is reduced under Reduction Rule 5.23. Let $j \in \mathbb{N}$ be such that $\mathsf{ex}(K_j) = \frac{1-\lambda}{2} + a$, where $a > 0$. Let $V(G-S) = V' \cup V'' \cup V'''$, where V' is the set of vertices contained in blocks of $G-S$ with exactly one vertex, V'' is the set of vertices contained in blocks of $G - S$ with between 2 and $j-1$ vertices and V''' is the set of vertices contained in blocks of $G - S$ with at least j vertices (note that in general $V'' \cap V''' \neq \emptyset$). Observe that the blocks containing V' are isolated blocks, therefore by Corollary 5.30 we may assume that $|V'| \leq (b_0 + b_1)k^2$ for some constants b_0, b_1 depending only on Π. Moreover, by Lemma 5.32, there exists a constant b_2 depending only on Π such that $|V''| \leq b_2(j-1)k^2$, or the instance is a `YES`-instance.

Now, let $\mathcal{B}'''$ be the set of blocks of $G - S$ which contain at least j vertices. Given a block $B \in \mathcal{B}'''$, let $jr_B + l_B$ be the number of its vertices, where $0 \leq l_B < j$. Note that, by Lemma 5.20 and Lemma 5.18, $\mathsf{ex}(B) \geq r_B a$. Let $d = \sum_{B \in \mathcal{B}'''} r_B$ and let G''' be the union of all components of $G-S$ which contain a block in $\mathcal{B}'''$. By Corollary 5.31, we may assume that G''' has at most $(b_0+b_1)k^2$ components. Furthermore, by repeated use of Lemma 5.5, we get that $\mathsf{ex}(G''') \geq da$. Observe that $G-G'''$ has at most $|S| \leq \frac{6k}{1-\lambda}$ components: then, by Lemma 5.18,

$\mathsf{ex}(G) \geq da - \frac{1-\lambda}{2}((b_0+b_1)k^2 + \frac{6k}{1-\lambda} - 1)$. Therefore if $d \geq \frac{1}{a}\left(\frac{1-\lambda}{2}((b_0+b_1)k^2 + \frac{6k}{1-\lambda} - 1) + k\right)$, the instance is a YES-instance. Otherwise, $|V'''| \leq 2dj \leq bjk^2$, where b is a constant which depends only on Π, which means that $|V(G)| = |S| + |V(G-S)| \leq \frac{6k}{1-\lambda} + (b_0 + b_1 + b_2(j-1) + bj)k^2$. □

Before moving to the next section, we prove that even when we cannot use the strong result of Theorem 5.34, we are able to derive a bound on $|\mathcal{B}_{\geq 3}|$ using the bound on $|\mathcal{B}_1|$.

Lemma 5.35. *The number of blocks in $\mathcal{B}_{\geq 3}$ is bounded by $3|\mathcal{B}_1|$.*

Proof. The proof is by induction on $|\mathcal{B}|$. We may assume that $G - S$ is connected, otherwise we can prove the bound separately for every component. If $|\mathcal{B}| = 1$, then $|\mathcal{B}_{\geq 3}| = 0$ and the bound trivially holds. Suppose now that $|\mathcal{B}| = l + 1 \geq 2$ and that the bound holds for every tree of cliques with at most l blocks. Let $B \in \mathcal{B}$ be a leaf block and let v be its root. Let $G' = G - (V(B) \setminus \{v\})$. $G' - S$ is a tree of cliques with at most l blocks, so by induction hypothesis $|\mathcal{B}'_{\geq 3}| \leq 3|\mathcal{B}'_1|$. Now, note that only block neighbours of B can be influenced by the deletion of B: in other words, if a block B' is not a block neighbour of B and $B' \in \mathcal{B}_i$, then $B' \in \mathcal{B}'_i$ for every $i \in \{1, 2, \geq 3\}$.

We distinguish three cases.

Case 1 (*B has at least three block neighbours*)**:** In this case it holds that $Q = Q'$, which ensures that the deletion of B does not increase the number of leaf blocks, that is, $|\mathcal{B}'_1| = |\mathcal{B}_1| - 1$. In addition, if a block neighbour B' of B is in $\mathcal{B}_{\geq 3}$, then it is in $\mathcal{B}'_{\geq 3}$, which means that $|\mathcal{B}'_{\geq 3}| = |\mathcal{B}_{\geq 3}|$. Therefore in this case, using the induction hypothesis it follows that $|\mathcal{B}_{\geq 3}| = |\mathcal{B}'_{\geq 3}| \leq 3|\mathcal{B}'_1| = 3|\mathcal{B}_1| - 3$.

Case 2 (*B has two block neighbours*)**:** As in the previous case $Q = Q'$, hence $|\mathcal{B}'_1| = |\mathcal{B}_1| - 1$. On the other hand, if a block neighbour B' of B is in $\mathcal{B}_{\geq 3}$, it could be the case that B' is in $\mathcal{B}'_2$. Therefore, we only have $|\mathcal{B}'_{\geq 3}| \geq |\mathcal{B}_{\geq 3}| - 2$. Using the induction hypothesis it follows that $|\mathcal{B}_{\geq 3}| \leq |\mathcal{B}'_{\geq 3}| + 2 \leq 3|\mathcal{B}'_1| + 2 = 3|\mathcal{B}_1| - 1$.

Case 3 (*B has exactly one block neighbour*)**:** Let $\widetilde{B}$ be the only block neighbour of B. Again, we distinguish three cases. If $\widetilde{B} \in \mathcal{B}_1$, then B and $\widetilde{B}$ are the only blocks of $G - S$ and $|\mathcal{B}_{\geq 3}| = 0$, therefore the bound trivially holds. If $\widetilde{B} \in \mathcal{B}_2$, then $\widetilde{B}$ is a leaf block in $G' - S$, hence $|\mathcal{B}_1| = |\mathcal{B}'_1|$ and $|\mathcal{B}_{\geq 3}| = |\mathcal{B}'_{\geq 3}|$: the bound follows using the induction hypothesis.

Lastly, let $\widetilde{B} \in \mathcal{B}_{\geq 3}$. If $|V(\widetilde{B}) \cap Q| \geq 3$, then $|\mathcal{B}'_{\geq 3}| \geq |\mathcal{B}_{\geq 3}| - 1$ and $|\mathcal{B}'_1| = |\mathcal{B}_1| - 1$: therefore, by induction hypothesis, $|\mathcal{B}_{\geq 3}| \leq |\mathcal{B}'_{\geq 3}| + 1 \leq 3|\mathcal{B}'_1| + 1 \leq 3|\mathcal{B}_1| - 2$. Otherwise, $|V(\widetilde{B}) \cap Q| = 2$ and $\widetilde{B}$ is a leaf block in $G' - S$. In this case, $|\mathcal{B}'_1| = |\mathcal{B}_1|$ and $|\mathcal{B}'_{\geq 3}| = |\mathcal{B}_{\geq 3}| - 1$. Now, consider the graph $G'' = G' - (V(\widetilde{B}) \setminus \{v'\})$, where v' is the root of $\widetilde{B}$ in $G' - S$. Deleting $\widetilde{B}$ from G' corresponds either to Case 1 or 2, hence it holds that $|\mathcal{B}''_1| = |\mathcal{B}'_1| - 1$ and $|\mathcal{B}''_{\geq 3}| \geq |\mathcal{B}'_{\geq 3}| - 2$. Therefore, using the induction hypothesis on $G'' - S$ (which is a tree of cliques with $l-1$ blocks) it follows that $|\mathcal{B}_{\geq 3}| = |\mathcal{B}'_{\geq 3}| + 1 \leq |\mathcal{B}''_{\geq 3}| + 3 \leq 3|\mathcal{B}''_1| + 3 = 3|\mathcal{B}'_1| = 3|\mathcal{B}_1|$, which concludes the proof. ☐

Corollary 5.36. *Let G be a connected graph reduced under Reduction Rule 5.23. Then $|\mathcal{B}_0| + |\mathcal{B}_1| + |\mathcal{B}_{\geq 3}| \leq 4(b_0 + b_1)k^2$ (where $b_0 + b_1$ is the constant of Corollary 5.30), or the instance is a* `YES`*-instance.*

Proof. The bound follows from Corollary 5.30 and Lemma 5.35. ☐

Note that combining Corollary 5.36 and Lemma 5.33 produces a bound on the number of blocks which are not contained in a block path, and on the size of their interiors. As a matter of fact, bounding the blocks in $\mathcal{B}_2$ looks like the most difficult part. We will not show a straightforward bound as the one in Corollary 5.36: instead, we will tackle the problem on a case-by-case basis.

5.2.2 Kernel when $\lambda \neq \frac{1}{2}$ or $K_3 \in \Pi$

Here, we will assume that $\lambda \neq \frac{1}{2}$, or that Π contains every graph whose underlying graph is isomorphic to K_3. In this case, we are able to make use of Theorem 5.34. The next lemmata have the purpose of showing that the hypothesis of the theorem hold.

Lemma 5.37. *It holds that $\mathsf{ex}(K_3) \geq 1 - 2\lambda$ and, if $\lambda > \frac{1}{2}$, $\mathsf{ex}(K_3) = 2 - 2\lambda$. In particular, $\mathsf{ex}(K_3) > 0$ in every case.*

Proof. Note that $\mathsf{opt}(G) \geq 2$ for any connected graph $G \in \mathcal{G}$ with at least two edges, because any graph whose underlying graph is a path on three vertices is in Π by Lemma 5.6. Therefore, $\mathsf{opt}(K_3) \geq 2$, which ensures that $\mathsf{ex}(K_3) \geq 2 - (3\lambda + \frac{1-\lambda}{2}2) = 1 - 2\lambda$, which is strictly greater than zero if $\lambda < \frac{1}{2}$.

Now, assume $\lambda > \frac{1}{2}$, let $G \in \mathcal{G}$ be such that $\mathcal{U}(G) = K_3$ and let $V(G) = \{v_1, v_2, v_3\}$. Consider $U = \{v_1, v_2\}$ and $W = \{v_3\}$ and note that $G[U], G[W] \in \Pi$ by Inclusiveness. Then, by strong λ-subgraph extension, it holds that $G \in \Pi$, which ensures that $\mathsf{opt}(K_3) = 3$. This means that $\mathsf{ex}(K_3) = 3 - (3\lambda + \frac{1-\lambda}{2}2) = 2 - 2\lambda > 0$. ☐

Lemma 5.38. *If $\lambda \neq \frac{1}{2}$, then $\mathsf{ex}(K_3) > \frac{1-\lambda}{2}$ or $\mathsf{ex}(K_4) > \frac{1-\lambda}{2}$. In particular, Π diverges on cliques.*

Proof. If $\lambda > \frac{1}{2}$, then by Lemma 5.37 $\mathsf{ex}(K_3) = 2 - 2\lambda > \frac{1-\lambda}{2}$. If $\lambda < \frac{1}{3}$, then by Lemma 5.37 $\mathsf{ex}(K_3) \geq 1 - 2\lambda$, which is greater than $\frac{1-\lambda}{2}$. Lastly, if $\frac{1}{3} \leq \lambda < \frac{1}{2}$, let $G \in \mathcal{G}$ be such that $\mathcal{U}(G) = K_4$ and let $V(G) = \{v_1, v_2, v_3, v_4\}$. Consider $U = \{v_1, v_2\}$ and $W = \{v_3, v_4\}$ and note that $G[U], G[W] \in \Pi$ by Inclusiveness. By strong λ-subgraph extension, it holds that $\mathsf{opt}(G) \geq 4$, since $\lambda > \frac{1}{4}$. Therefore, $\mathsf{ex}(K_4) \geq 4 - 6\lambda - \frac{1-\lambda}{2}3 = \frac{5}{2} - \frac{9}{2}\lambda$ which is greater than $\frac{1-\lambda}{2}$. □

Lemma 5.39. *If $K_3 \in \Pi$, then $\mathsf{ex}(K_3) > \frac{1-\lambda}{2}$. In particular, Π diverges on cliques.*

Proof. If $K_3 \in \Pi$, then $\mathsf{opt}(K_3) = 3$, which means that $\mathsf{ex}(K_3) = 2 - 2\lambda > \frac{1-\lambda}{2}$. □

Lemma 5.40. *If $\lambda \neq \frac{1}{2}$ or $K_3 \in \Pi$, then $\mathsf{ex}(K_i) > 0$ for all $i \geq 2$.*

Proof. By Lemma 5.38 and Lemma 5.39, $\mathsf{ex}(K_3) > \frac{1-\lambda}{2}$ or $\mathsf{ex}(K_4) > \frac{1-\lambda}{2}$. In the first case, by Lemma 5.21, it holds that $\mathsf{ex}(K_j) > 0$ for all $j \geq 4$, while in the second case, using the same lemma, $\mathsf{ex}(K_j) > 0$ for all $j \geq 5$. In addition, by Lemma 5.37, $\mathsf{ex}(K_3) > 0$. Finally, $\mathsf{ex}(K_2) = 1 - (\lambda + \frac{1-\lambda}{2}) = \frac{1-\lambda}{2} > 0$. □

Theorem 5.41. *Let Π be a strongly λ-extendible property. If $\lambda \neq \frac{1}{2}$ or $K_3 \in \Pi$, then the* Above Poljak-Turzík *(Π) problem admits a kernel with $\mathcal{O}(k^2)$ vertices.*

Proof. By Lemma 5.38 or Lemma 5.39, Π diverges on cliques. Furthermore, by Lemma 5.40, $\mathsf{ex}(K_i) > 0$ for all $i \geq 2$. Then, by Theorem 5.34, $APT(\Pi)$ admits a kernel with at most $\mathcal{O}(k^2)$ vertices. □

5.2.3 Kernel when $\lambda = \frac{1}{2}$

It is left to consider when $\lambda = \frac{1}{2}$ and Π does not contain every graph whose underlying graph is isomorphic to K_3. In this case it is not true that every property diverges: for instance, the size of an optimal solution for Max Cut on complete graphs with an odd number of vertices is equal to the Poljak-Turzík bound, independently from how large the graphs are.

At the same time, in a sense, the situation with Max Cut is the 'worst possible'. We will show that every property Π, except 'being bipartite', diverges, and we will use this fact to produce a kernel for $APT(\Pi)$. Nonetheless, we are only able to do so for hereditary properties on simple or oriented graphs.

Definition 5.42. A graph property Π is *hereditary* if, for any graph G and vertex-induced subgraph G' of G, if $G \in \Pi$ then $G' \in \Pi$.

Observe that all λ-extendible properties which Poljak and Turzík [78] described in their paper are hereditary.

From now on, assume $\lambda = \frac{1}{2}$. The next theorem characterizes the hereditary properties which are equivalent to 'being bipartite', in terms of the complete graphs with three vertices they contain. Let $\mathcal{G}_{\mathcal{S}}$ contain all simple graphs.

Theorem 5.43. *Suppose* Π *is hereditary and* $G_3 \notin \Pi$ *for any* $G_3 \in \mathcal{G}$ *such that* $\mathcal{U}(G_3) = K_3$. *Then for every* $G \in \mathcal{G}$, $G \in \Pi$ *if and only if* $\mathcal{U}(G) \in \Pi_{BP} = \{G_S \in \mathcal{G}_{\mathcal{S}} : G_S$ *is bipartite*$\}$.

Proof. First, assume for the sake of contradiction that Π contains a non-bipartite graph H. Then H contains an odd cycle C_l. By choosing l as small as possible we may assume that C_l is a vertex-induced subgraph of H. Then, since Π is hereditary, C_l is in Π. Note that if $l = 3$, then $\mathcal{U}(C_3) = K_3$, so this is not the case. Consider the graph H' obtained from C_l adding a new vertex v and an edge from v to every vertex of C_l. Since both C_l and $K_1 = \{v\}$ are in Π, by STRONG λ-SUBGRAPH EXTENSION we can find a subgraph of H' which contains C_l, v and at least half of the edges between v and C_l. Since l is odd, for any choice of $\frac{l+1}{2}$ edges there are two of them, say $e_1 = vu$ and $e_2 = vw$, such that the edge uw is in $E(C_l)$. Therefore, since Π is hereditary, $H'[v, u, w] \in \Pi$, which leads to a contradiction, as $\mathcal{U}(H'[v, x, y]) = K_3$.

Now, we will show that all connected bipartite graphs are in Π, independently from any possible labelling and/or orientation. We will proceed by induction. The claim is true for $j = 1, 2$ by INCLUSIVENESS. Assume $j \geq 3$ and that every bipartite graph with $l < j$ vertices is in Π. Consider any connected bipartite graph H with j vertices, and let V_1, V_2 be a partition of $V(H)$ such that $E(H) = E(V_1, V_2)$. Consider a vertex v such that $H' = H - \{v\}$ is connected. By induction hypothesis, $H' \in \Pi$. Consider the graph H'' obtained from H' and G_2, where G_2 is any graph in $\mathcal{G}$ with $\mathcal{U}(G_2) = K_2$ (let $V(G_2) = \{v_1, v_2\}$), adding an edge from v_i, $i = 1, 2$, to $w \in V(H')$ if and only if in H there is an edge from v to w. Colour red the edges from v_1 and blue the edges from v_2.

Since $G_2 \in \Pi$ by INCLUSIVENESS and $H' \in \Pi$, by STRONG λ-SUBGRAPH EXTENSION there must exist a subgraph $\widetilde{H}$ of H'' which contains G_2, H' and at least half of the edges between them. Note that the red edges are exactly half of the edges and that if $\widetilde{H}$ contains all of them and no blue edges, then we can conclude that H is in Π by BLOCK ADDITIVITY. The same holds if $\widetilde{H}$ contains every blue edge and no red edge.

If, on the contrary, $\widetilde{H}$ contains one red and one blue edge, we will show that it contains a vertex-induced cycle of odd length, which leads to a contradiction according to the first part of the proof. First, suppose that both these edges contain $w \in V(H')$: if this happens, $\widetilde{H}$

contains a cycle of length 3 as a vertex-induced subgraph.

Now, suppose $\widetilde{H}$ contains a red edge v_1w_1 and a blue edge v_2w_2. Since H' is connected, there is a path from w_1 to w_2 and, since w_1 and w_2 are in the same side of the partition of $V(H)$ (that is, both in V_1 or both in V_2), the path has even length. Together with v_1w_1, v_2w_2 and v_1v_2, this gives a cycle of odd length. Choosing the shortest path between w_1 and w_2, we may assume that the cycle is vertex-induced.

Thus, we conclude that the only possible choices to make $\widetilde{H}$ are either picking the red edges or picking the blue edges, which concludes the proof. □

Theorem 5.43 ensures that when Π contains all and only bipartite graphs we can use the following result to compute a polynomial kernel (see Section 5.3 for a proof).

Theorem 5.76. *The* Above Poljak-Turzík *(Π_{BP}) problem admits a kernel with $\mathcal{O}(k^3)$ vertices.*

Simple case

Theorem 5.44. *Let $\mathcal{G}_S$ be the class of simple graphs, that is, without any labelling or orientation. Let $\Pi \subseteq \mathcal{G}_S$ be a strongly λ-extendible property, with $\lambda = \frac{1}{2}$, and suppose Π is hereditary. Then $APT(\Pi)$ admits a kernel with $\mathcal{O}(k^2)$ or $\mathcal{O}(k^3)$ vertices.*

Proof. Firstly, note that in this case there is only one graph, up to isomorphism, whose underlying graph is K_3 (namely, K_3 itself). If $K_3 \notin \Pi$, by Theorem 5.43 $\Pi = \Pi_{BP}$ and therefore by Theorem 5.76 it admits a kernel with $\mathcal{O}(k^3)$ vertices. On the other hand, if $K_3 \in \Pi$, then by Theorem 5.41 Π admits a kernel with $\mathcal{O}(k^2)$ vertices. □

Oriented case

When $\mathcal{G}$ is the class $\mathcal{G}_O$ of oriented graphs, the proof of the existence of a polynomial kernel is more involved. The difference is that in this case there are two different graphs whose underlying graph is isomorphic to K_3.

Definition 5.45. Let $\overrightarrow{K}_3 \in \mathcal{G}_O$ be such that $\mathcal{U}(\overrightarrow{K}_3)$ is isomorphic to K_3, $V(\overrightarrow{K}_3) = \{v_1, v_2, v_3\}$ and v_iv_{i+1} is oriented from v_i to v_{i+1} for $1 \leq i \leq 3$ (where subscripts are taken modulo 3). We will call $\overrightarrow{K}_3$ the *oriented* triangle.

Similarly, let $\overrightarrow{\overrightarrow{K}}_3 \in \mathcal{G}_O$ be such that $\mathcal{U}(\overrightarrow{\overrightarrow{K}}_3)$ is isomorphic to K_3, $V(\overrightarrow{\overrightarrow{K}}_3) = \{u_1, u_2, u_3\}$ and u_iu_j is oriented from u_i to u_j for $1 \leq i < j \leq 3$. We will call $\overrightarrow{\overrightarrow{K}}_3$ the *non-oriented* triangle.

It is not difficult to see that, up to isomorphism, $\overrightarrow{K}_3$ and $\overleftrightarrow{K}_3$ are the only graphs in $\mathcal{G}_{\mathcal{O}}$ with K_3 as underlying graph.

Lemma 5.46. *If* $\overrightarrow{K}_3 \in \Pi$, *then* $\overleftrightarrow{K}_3 \in \Pi$.

Proof. Consider the graph H obtained by adding a vertex v to $\overrightarrow{K}_3$ and an edge from v to $v_i \in V(\overrightarrow{K}_3)$, oriented from v to v_i $(i = 1, 2, 3)$. Since $\overrightarrow{K}_3 \in \Pi$, by STRONG λ-SUBGRAPH EXTENSION there exists a subgraph $H' \in \Pi$ of H which contains $\overrightarrow{K}_3$, v and at least two edges between $\overrightarrow{K}_3$ and v: without loss of generality, assume these edges are vv_1 and vv_2. Then since Π is hereditary $H'[v, v_1, v_2] \in \Pi$ and note that $H'[v, v_1, v_2]$ is isomorphic to $\overleftrightarrow{K}_3$. $\square$

Lemma 5.46 shows that there are only three possibilities: (i) $K_3 \in \Pi$, (ii) $\overrightarrow{K}_3 \notin \Pi$ and $\overleftrightarrow{K}_3 \in \Pi$, and (iii) $\overrightarrow{K}_3, \overleftrightarrow{K}_3 \notin \Pi$. The one that we cannot immediately solve using the results of the previous sections is the second one. The rest of this section is mainly devoted to study this case.

Lemma 5.47. *If* $\overleftrightarrow{K}_3 \in \Pi$, *then* $ex(K_4) > \frac{1}{4}$. *In particular,* Π *diverges on cliques.*

Proof. Let $H \in \mathcal{G}_{\mathcal{O}}$ be such that $\mathcal{U}(H)$ is isomorphic to K_4 and let $V(H) = \{w_1, w_2, w_3, w_4\}$. If $H[w_1, w_2, w_3]$ and $H[w_2, w_3, w_4]$ are isomorphic to $\overrightarrow{K}_3$, then $H[w_1, w_2, w_4]$ is isomorphic to $\overleftrightarrow{K}_3$. Hence, for any orientation on the edges of H, the graph contains $\overleftrightarrow{K}_3$ as a vertex-induced subgraph. Now, since $\overleftrightarrow{K}_3 \in \Pi$, by STRONG λ-SUBGRAPH EXTENSION there exists a subgraph of H which is in Π and contains at least 5 edges, which means that $\mathsf{opt}(H) \geq 5$. This ensures that $\mathsf{ex}(K_4) \geq 5 - (3 + \frac{3}{4}) = \frac{5}{4}$, which concludes the proof. $\square$

Lemma 5.48. *If* $\overleftrightarrow{K}_3 \in \Pi$, *then* $\mathsf{ex}(K_j) > 0$ *for every* $j \neq 3$.

Proof. Note that $\mathsf{ex}(K_4) > \frac{1}{4}$, then by Lemma 5.21 $\mathsf{ex}(K_j) > 0$ for every $j \geq 4$. In addition, $\mathsf{ex}(K_2) = \frac{1}{4}$. $\square$

When $\overrightarrow{K}_3 \notin \Pi$ and $\overleftrightarrow{K}_3 \in \Pi$, the property diverges by Lemma 5.47, but we cannot use Theorem 5.34 because $\mathsf{ex}(\overrightarrow{K}_3) = 0$. Nonetheless, by Corollary 5.36 we only need to bound $|\mathcal{B}_2|$.

Let $\mathcal{B}_2^0$ be the subset of $\mathcal{B}_2$ which contains all the blocks with excess zero which have no internal vertices in $N(S)$. Let Q_0 denote the set of cutvertices of $G - S$ which only appear in blocks in $\mathcal{B}_2^0$. Note that every vertex in Q_0 appears in exactly two blocks in $\mathcal{B}_2^0$.

Lemma 5.49. *Suppose $\overrightarrow{K}_3 \notin \Pi$ and $\overleftrightarrow{K}_3 \in \Pi$. Let (G, S, k) be an instance of SAPT(Π) reduced under Reduction Rule 5.23. For any $s \in S$, if $|Q_0 \cap N(s)| \geq (b_0 + b_1 + 4)k^2$ (where $b_0 + b_1$ is the constant of Corollary 5.30), then the instance is a* YES*-instance.*

Proof. First, note that all the blocks in $\mathcal{B}_2^0$ are isomorphic to $\overrightarrow{K}_3$ by Lemma 5.48. Observe that every vertex in Q_0 has at most two neighbours in Q_0. Since all vertices in Q_0 are cutvertices of $G - S$, it follows that $G[Q_0 \cap N(s)]$ is a disjoint union of paths. It follows that we can find a set $Q_0' \subseteq Q_0 \cap N(s)$ such that $|Q_0'| \geq \frac{|Q_0 \cap N(s)|}{2}$ and Q_0' is an independent set.

For each $v \in Q_0'$, let B_1, B_2 be the two blocks in $\mathcal{B}_2^0$ that contain v, and let v_i be the unique vertex in $(B_i)_{\mathsf{int}}$, for $i \in \{1, 2\}$. Then let $U = \{s\} \cup Q_0' \cup \{v_i : v \in Q_0', i \in \{1, 2\}\}$, and observe that $G[U]$ is a tree with $3|Q_0'|$ edges. By Lemma 5.6 it follows that $G[U] \in \Pi$ and $\mathsf{ex}(G[U]) = \frac{3|Q_0'|}{4}$. By Lemma 5.18, $\mathsf{ex}(G) \geq \frac{3|Q_0'| - c}{4}$, where c is the number of components of $G - U$.

Consider the components of $G - U$. Each component either contains a block in $\mathcal{B}_1 \cup \mathcal{B}_{\geq 3}$ or it is part of a block path of $G - S$ containing two vertices from Q_0': by Corollary 5.30 there are at most $(b_0 + b_1)k^2$ components of the first kind, while there are at most $|Q_0'|$ of the second kind.

Thus, if $2|Q_0'| - (b_0 + b_1)k^2 \geq 4k$ then we have a YES-instance; otherwise $|Q_0 \cap N(s)| \leq 2|Q_0'| \leq (b_0 + b_1)k^2 + 4k \leq (b_0 + b_1 + 4)k^2$. □

Lemma 5.49 is needed to bound the number of cutvertices adjacent to S. To ensure that there cannot be long block paths which do not contain neighbours of S we will need the following reduction rule:

Reduction Rule 5.50. *Let $B_1, B_2 \in \mathcal{B}_2$ be blocks isomorphic to $\overrightarrow{K}_3$ such that $V(B_1) \cap V(B_2) = \{v\}$, $\{v\} \cap N(S) = \emptyset$ and $(B_i)_{\mathsf{int}} \cap N(S) = \emptyset$ for $i = 1, 2$. Let $\{w_i\} = (B_i)_{\mathsf{int}}$ and $\{u_i\} = V(B_i) \setminus \{v, w_i\}$ for $i = 1, 2$. If $G - \{v\}$ is disconnected, delete v, w_1, w_2, identify u_1 and u_2 and set $k' = k$. Otherwise, delete v, w_1, w_2 and set $k' = k - \frac{1}{4}$.*

Lemma 5.51. *If $\overrightarrow{K}_3 \notin \Pi$, then Reduction Rule 5.50 is valid.*

Proof. Let G' be the graph which is obtained after an application of the rule. Initially, suppose that $G - \{v\}$ is disconnected and let G'' be the graph obtained from G deleting v, w_1 and w_2 and without identifying any vertices. Then, note that G'' has two connected components, one containing u_1 and the other containing u_2: hence, G' is connected. Additionally, observe that $\mathsf{pt}(G) = \mathsf{pt}(G'') + \mathsf{pt}(G[\{v, w_1, w_2, u_1, u_2\}])$, $\mathsf{opt}(G'') = \mathsf{opt}(G')$ (by BLOCK ADDITIVITY) and $\mathsf{pt}(G'') = \mathsf{pt}(G')$. If, on the other hand, $G - \{v\}$ is connected, then $\mathsf{pt}(G) = \mathsf{pt}(G') + \mathsf{pt}(G[\{v, w_1, w_2, u_1, u_2\}]) - \frac{1}{4}$. In both cases, $\mathsf{pt}(G[\{v, w_1, w_2, u_1, u_2\}]) = 4$. Let $\widetilde{G} = G''$ if $G - \{v\}$ is disconnected and $\widetilde{G} = G'$ otherwise: we are done if we show that $\mathsf{opt}(G) = \mathsf{opt}(\widetilde{G}) + 4$.

Let $\widetilde{H} \in \Pi$ be a subgraph of $\widetilde{G}$: we may assume that $\widetilde{H}$ is a spanning subgraph, otherwise we may extend it adding a spanning tree for every connected component of $\widetilde{G} - V(\widetilde{H})$ (note that the resulting graph is in Π by Inclusiveness and Block additivity). Moreover, $G[v, w_1, w_2]$ is a tree and is in Π by Lemma 5.6. Then, by strong λ-subgraph extension there exists a subgraph $H \in \Pi$ of G which contains $\widetilde{H}$, $G[v, w_1, w_2]$ and at least half of the edges between them. Note that these edges are exactly four: vu_1, w_1u_1, vu_2 and w_2u_2. If vu_1 and w_1u_1 are in $E(H)$, then since Π is hereditary it holds that $\overrightarrow{K}_3 \in \Pi$, which is a contradiction. Similarly if vu_2 and w_2u_2 are in $E(H)$. This means that exactly two edges among them are in $E(H)$, that is that $|E(H)| = |E(\widetilde{H})| + 4$, which ensures that $\mathsf{opt}(G) \geq \mathsf{opt}(\widetilde{G}) + 4$. On the other hand, if $H \in \Pi$ is a subgraph of G, then $H[V(\widetilde{G})]$ is a subgraph of $\widetilde{G}$ which is in Π because Π is hereditary. If $|E(H[\{v, w_1, w_2, u_1, u_2\}]) \geq 5$, then $\overrightarrow{K}_3$ is a vertex-induced subgraph of H and is therefore in Π, a contradiction. Thus, $\mathsf{opt}(G) \leq \mathsf{opt}(\widetilde{G}) + 4$.

Finally, observe that the rule can be applied in polynomial time, as the block decomposition of $G - S$ can be computed in polynomial time. □

Note that every time Rule 5.50 applies, the resulting graph contains less vertices, hence we can compute in polynomial time a graph which is reduced under this rule. We are finally able to describe the kernelization.

Theorem 5.52. *Let $\mathcal{G}_{\mathcal{O}}$ be the class of oriented graphs. Let $\Pi \subseteq \mathcal{G}_{\mathcal{O}}$ be a strongly λ-extendible property, with $\lambda = \frac{1}{2}$, and suppose Π is hereditary. Then $APT(\Pi)$ admits a kernel with $\mathcal{O}(k^2)$ or $\mathcal{O}(k^3)$ vertices.*

Proof. If $\overrightarrow{K}_3 \in \Pi$, by Lemma 5.46 $\overset{\nrightarrow}{K}_3 \in \Pi$. This means that $K_3 \in \Pi$ and, by Theorem 5.41, $APT(\Pi)$ admits a kernel with $\mathcal{O}(k^2)$ vertices. On the other hand, if $\overrightarrow{K}_3 \notin \Pi$ and $\overset{\nrightarrow}{K}_3 \notin \Pi$, then, by Theorem 5.43, $G \in \Pi$ if and only if $\mathcal{U}(G) \in \Pi_{BP}$ for every $G \in \mathcal{G}$; hence, by Theorem 5.76, $APT(\Pi)$ admits a kernel with $\mathcal{O}(k^3)$ vertices.

Lastly, suppose $\overset{\nrightarrow}{K}_3 \in \Pi$ and $\overrightarrow{K}_3 \notin \Pi$. By Corollary 5.14, $APT(\Pi)$ can be reduced in polynomial time to $SAPT(\Pi)$, so it is enough to describe a kernelization for the latter problem. By Lemma 5.47, Π diverges on cliques. Let (G, S, k) be an instance of $SAPT(\Pi)$ reduced by Reduction Rule 5.23 and 5.50: note that by Lemma 5.24 and Lemma 5.51 both rules are valid.

By Corollary 5.36, we may assume that $|\mathcal{B}_0| + |\mathcal{B}_1| + |\mathcal{B}_{\geq 3}| \leq 4(b_0 + b_1)k^2$, for some constants b_0, b_1 depending only on Π. We now need to consider different types of blocks in $\mathcal{B}_2$ separately. Let $\mathcal{B}_2^+$ be the blocks in $\mathcal{B}_2$ with positive excess. By Lemma 5.32, we may assume the number of such blocks is at most b_2k^2 for some constant b_2 depending only on Π.

Let $\mathcal{B}_2'$ be the blocks in $\mathcal{B}_2 \setminus \mathcal{B}_2^+$ which have an interior vertex in $N(S)$. By Corollary 5.29, we may assume the number of such blocks is at most b_1k^2.

Let $\mathcal{B}_2''$ be the blocks in $\mathcal{B}_2 \setminus (\mathcal{B}_2^+ \cup \mathcal{B}_2')$ which contain a vertex in $Q \cap N(S)$. Observe that these blocks must either contain a vertex of $Q_0 \cap N(S)$ or be adjacent to a block in $\mathcal{B}_1, \mathcal{B}_{\geq 3}, \mathcal{B}_2^+$ or $\mathcal{B}_2'$. Furthermore they must be in block paths between such blocks, from which it follows that $|\mathcal{B}_2''| \leq 2(|\mathcal{B}_1| + |\mathcal{B}_{\geq 3}| + |\mathcal{B}_2^+| + |\mathcal{B}_2'| + |Q_0 \cap N(S)|)$.

Finally let $\mathcal{B}_2''' = \mathcal{B}_2 \setminus (\mathcal{B}_2^+ \cup \mathcal{B}_2' \cup \mathcal{B}_2'')$. These are just the blocks in $\mathcal{B}_2$ with excess 0 which contain no neighbours of S. By Reduction Rule 5.50, no two such blocks can be adjacent. Therefore every block in $\mathcal{B}_2'''$ is adjacent to two blocks from $\mathcal{B}_1, \mathcal{B}_{\geq 3}, \mathcal{B}_2^+, \mathcal{B}_2'$ or $\mathcal{B}_2''$. It follows that $|\mathcal{B}_2'''| \leq |\mathcal{B}_1| + |\mathcal{B}_{\geq 3}| + |\mathcal{B}_2^+| + |\mathcal{B}_2'| + |\mathcal{B}_2''|$.

Note that by Lemma 5.49 and the fact that $|S| \leq 12k$, we may assume that $|Q_0 \cap N(S)| \leq b_4k^3$ for some constant b_4 depending only on Π. Then we may conclude from the above that $|\mathcal{B}_2'| + |\mathcal{B}_2''| + |\mathcal{B}_2'''| \leq b_5k^3$ for some constant b_5 depending only on Π.

Therefore the total number of blocks in $G - S$ is at most $4(b_0 + b_1)k^2 + b_2k^2 + b_5k^3$.

By Lemma 5.33, we may assume that the number of vertices contained in any block is at most b_3k, for some constant b_3 depending only on Π. It follows that the number of vertices in blocks from $\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_{\geq 3}$ or $\mathcal{B}_2^+$ is at most $b_3(b_0 + b_1 + b_2)k^3$. To bound the number of vertices in blocks from $\mathcal{B}_2' \cup \mathcal{B}_2'' \cup \mathcal{B}_2'''$, note that each of these blocks contains at most 3 vertices, by Lemma 5.48 and the fact that these blocks have excess 0 by definition. Therefore the number of vertices in blocks from $\mathcal{B}_2' \cup \mathcal{B}_2'' \cup \mathcal{B}_2'''$ is at most $3b_5k^3$. Finally, recalling that $|S| \leq 12k$, we have that the number of vertices in G is at most $\mathcal{O}(k^3)$, as required. □

5.3 Signed Max Cut

The Signed Max Cut problem is a generalization of Max Cut to signed graphs. A *signed* graph is a simple graph where every edge is labelled by $+$ or $-$. An edge is *positive* if it is labelled $+$ and *negative* otherwise: the labels are the *signs* of the corresponding edges.

Let $G = (V, E)$ be a signed graph and let $V = V_1 \cup V_2$ be a partition of V. We say that G

is (V_1, V_2)-*balanced* if an edge with both endpoints in V_1, or both endpoints in V_2, is positive, and an edge with one endpoint in V_1 and one endpoint in V_2 is negative; G is *balanced* if it is (V_1, V_2)-balanced for some partition V_1, V_2 of V (V_1 or V_2 may be empty). The problem we will consider in this section is defined as follows.

Signed Max Cut APT	
Input:	A pair (G, k) where $G = (V, E)$ is a connected signed graph, and k is an integer.
Parameter:	k
Question:	Is there a balanced subgraph H of G with at least $\mathsf{pt}(G) + k$ edges?

When a graph contains only negative edges, Signed Max Cut APT asks for a bipartite subgraph of $\mathcal{U}(G)$ with at least $\mathsf{pt}(G) + k$ edges, therefore the problem is equivalent to Max Cut APT. The problem, in general, has various applications and interesting theoretical properties [14, 25, 51, 86]. Previously, it has been studied from the parameterized point of view by Hüffner *et al.* [56], who considered the parameterization below the number of edges (that is, decide whether there exists a balanced subgraph with at least $|E| - k$ edges, where k is the parameter), and showed that it admits an FPT-algorithm.

In this section, we will show that Signed Max Cut APT admits a polynomial kernel, and we will use the result to derive a polynomial kernel for Max Cut APT, thus proving Theorem 5.76, which has been used in Section 5.2 to prove the existence of a generic kernelization for $APT(\Pi)$.

For an edge set F of a signed graph G, F^+ and F^- denote the set of positive and negative edges of F, respectively. Let $G = (V, E)$ be a signed graph. For a partition V_1, V_2 of the vertex set V of a signed graph $G = (V, E)$, the *balanced subgraph H induced by this partition* is the subgraph of G which contains all the positive edges in $G[V_1]$ and $G[V_2]$, and all the negative edges in $E(V_1, V_2)$.

We say that a cycle C in G is *positive* if the number of negative edges is even, and it is *negative* otherwise. A *triangle* is a cycle with three edges. If G is a signed graph, the *positive* neighbours of $W \subseteq V$ are the neighbours of W in $G^+ = (V, E^+)$; the set of positive neighbours is denoted $N^+_G(W)$. Similarly, for the *negative* neighbours and $N^-_G(W)$. The next theorem is a well-known characterization of the condition of 'being balanced'.

Theorem 5.53. *[51] A signed graph G is balanced if and only if every cycle in G is positive.*

For a subset $W \subseteq V$, the W-*switch* of G is the signed graph G_W obtained from G by switching the signs of the edges in $E(W, V \setminus W)$. Note that the sizes of the largest balanced subgraphs of G and of G_W are the same; in fact, the balanced subgraph H of G induced by a partition V_1, V_2 is isomorphic to the balanced subgraph H' of G_W induced by $(V_1 \setminus W) \cup (V_2 \cap W)$ and $(V_2 \setminus W) \cup (V_1 \cap W)$.

The next theorems, which we state without proof, will prove useful later.

Theorem 5.54. *[43] Let $G = (V, E)$ be a signed graph. Deciding whether G is balanced is solvable in polynomial time. Moreover, if G is balanced then, in polynomial time, we can find a subset W of V such that G_W has no negative edges.*

Theorem 5.55. *[21] Let $(G = (V, E), k)$ be an instance of* Signed Max Cut APT*, let $U \subseteq V$ and let $G[U]$ be a chordal graph which does not contain a positive triangle. Then there exists a set $W \subseteq U$, such that the instance (G_W, k) is equivalent to (G, k), and $G_W[U]$ does not contain positive edges.*

For the following, it will be useful to note that a forest of cliques is a chordal graph.

Now, we want to show that the notion of strongly λ-extendible property is general enough to capture the property of 'being balanced'.

Theorem 5.56. *Let $\mathcal{G}^+_-$ contain all signed graphs, let $\Pi_{BL} \subseteq \mathcal{G}^+_-$ contain all balanced graphs and let $\lambda = \frac{1}{2}$. Then Π_{BL} is a hereditary strongly λ-extendible property.*

Proof. It is hereditary because if a graph contains no negative cycles then the same is true for every subgraph. Now, observe that K_1 and K_2 are balanced for every labelling because they contain no cycles, hence Inclusiveness holds. Moreover, Block additivity holds because a graph does not contain a negative cycle if and only if its blocks do not contain a negative cycle, as every cycle is contained in one block. Finally, let $G = (V, E)$ be any signed graph and assume that $G[U]$ and $G[W]$ are balanced graphs for a partition U, W of V. Let $U_1 \subseteq U$ and $W_1 \subseteq W$ be such that $G[U]_{U_1}$ and $G[W]_{W_1}$ contain no negative edges (see Theorem 5.54). Let $X = U_1 \cup W_1 \subseteq V$. Let $F \subseteq E(U, W)$ contain all the edges which are positive in G_X, or all the edges which are negative (select the largest group). In the first case, $(G_X) \setminus (E(U, W) \setminus F)$ contains no negative edges; in the other, let $Y = (U \setminus U_1) \cup W_1$ and observe that $(G_Y) \setminus (E(U, W) \setminus F)$ contains no negative edges. In both cases we find a balanced subgraph of G containing $G[U]$ and $G[W]$ and at least half of the edges in $E(U, W)$. $\square$

Observe that Signed Max Cut APT is equivalent to the Above Poljak-Turzík (Π_{BL}) problem. Therefore, we can use the results of Section 5.1, in particular Theorem 5.13, to immediately obtain the following result.

Theorem 5.57. *Given an instance (G, k) of* Signed Max Cut APT*, in polynomial time it is possible either to answer* `YES`*, or to produce a set $S \subseteq V$ such that $G - S$ is a forest of cliques, and $|S| \leq 12k$.*

Making use of Theorem 5.57, we only need to bound the number of vertices in $G - S$. First of all, we need to modify the graph in such a way that $G - S$ contains only negative edges. Let $\mathcal{B}$ be the set of blocks of $G - S$ and recall the definitions of $\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_{\geq 3}$ from Section 5.2:

- $\mathcal{B}_0$ is the set of all blocks of $G - S$ which contain no cutvertex of $G - S$, and therefore have no block neighbour;
- $\mathcal{B}_1$ is the set of all blocks of $G - S$ which contain exactly one cutvertex of $G - S$, and therefore have at least one block neighbour;
- $\mathcal{B}_2$ is the set of all blocks of $G - S$ which contain exactly two cutvertices of $G - S$, *and* have exactly two block neighbours; and,
- $\mathcal{B}_{\geq 3}$ is the set of all the remaining blocks of $G - S$. A block of $G - S$ is in $\mathcal{B}_{\geq 3}$ if and only if it contains at least two cutvertices of $G - S$, *and* has at least three block neighbours.

We will need a reduction rule:

Reduction Rule 5.58. *Let $B \in \mathcal{B}_1$ be a leaf block which does not contain a neighbour of S in its interior. If B contains no positive triangles, then delete B_{int} and decrease k by $\frac{1}{4}$ if $|V(B)|$ is even, and leave it the same otherwise.*

Lemma 5.59. *Reduction Rule 5.58 is valid.*

Proof. It is not difficult to see that the rule can be applied in polynomial time. Now, let (G', k') be the instance obtained after an application of Rule 5.58. Then by Lemma 5.5, $\mathsf{ex}(G) = \mathsf{ex}(G') + \mathsf{ex}(B)$, as $B - \{v\}$ is a connected component of $G - \{v\}$, where v is the root of B. Now, observe that by Theorem 5.55 we may assume that B contains only negative edges, as it does not contain a positive triangle by hypothesis. Hence, $\mathsf{opt}(B)$ is equal to the maximum size of a bipartite subgraph of B, which is equal to $\mathsf{pt}(B) + \frac{1}{4}$ if $|V(B)|$ is even, and is equal to $\mathsf{pt}(B)$ otherwise. □

We will show that we can add the vertices of the positive triangles in $G - S$ to S, and that in doing so the size of S increases only by a constant, or the instance is a `YES`-instance.

Theorem 5.60. *Given an instance (G, k) of* Signed Max Cut APT *and a set $S \subseteq V$, such that $G - S$ is a forest of cliques and $|S| \leq 12k$, in polynomial time it is possible either to answer* `YES`*, or to find a set $S' \supseteq S$ of vertices such that $G - S'$ is a forest of cliques which does not contain positive edges, and $|S'| \leq 60k$.*

Proof. Start by exhaustively applying Reduction Rule 5.58: every time the rule applies, the number of vertices of G decreases, so in polynomial time we obtain a graph which is reduced under this rule. Then greedily construct a set $\mathcal{T}$ of edge-disjoint positive triangles in $G - S$, with the condition that triangles chosen in the same block must also be vertex-disjoint. We will show that either (G, k) is a `YES`-instance, or $|\mathcal{T}|$ is bounded. Let U be the set of vertices contained in triangles in $\mathcal{T}$ and, for every block B in $\mathfrak{B}(G[U])$, let $\mathcal{T}_B$ be the set of triangles in $\mathcal{T}$ which are contained in B. Note that every block of $G[U]$ is contained in a block of $G - S$. Hence, by repeated use of Lemma 5.5 and Lemma 5.18, we obtain that

$$\mathsf{ex}(G[U]) = \sum_{B \in \mathfrak{B}(G[U])} \mathsf{ex}(B) \geq \sum_{B \in \mathfrak{B}(G[U])} (|\mathcal{T}_B| - \frac{1}{4}(|\mathcal{T}_B| - 1))$$

Hence $\mathsf{ex}(G[U]) - \frac{1}{4}|\mathfrak{C}(G[U])| \geq \mathsf{ex}(G[U]) - \frac{1}{4}|\mathfrak{B}(G[U])| \geq \sum_{B \in \mathfrak{B}(G[U])} \frac{3}{4}|\mathcal{T}_B| = \frac{3}{4}|\mathcal{T}|$.

Now we consider the components of $G - U$. We will show by induction on the number of blocks of $G - S$ that they are at most $2|\mathcal{T}| + |S|$. If there is only one block B, if B does not contain a positive triangle then $U = \emptyset$ and $G - U$ is connected, otherwise $G - U$ has at most $|S| + 1$ components and we are done. Now assume that $G - S$ contains $r > 1$ blocks. If $\mathcal{B}_1 = \emptyset$, then $\mathcal{B} = \mathcal{B}_0$: in this case, $G - U$ has at most $|S| + |\mathcal{T}|$ components, as every block which is disconnected from S in $G - U$ contains a positive triangle. So assume that $\mathcal{B}_1 \neq \emptyset$ and let $B \in \mathcal{B}_1$. Since the graph is reduced under Reduction Rule 5.58, either B contains a positive triangle in $\mathcal{T}$ or $B_{\mathsf{int}} \setminus U$ contains a neighbour of S. The idea is that we want to delete B to use the induction hypothesis, but simply doing this is not enough, as the resulting graph may not be reduced under Reduction Rule 5.58.

Hence, let $B_0, B_1, \ldots, B_l, B_{l+1}$ be a sequence of blocks in $G - S$ such that the subgraph induced by $V(B_1) \cup \cdots \cup V(B_l)$ is a block path, $B_0 = B$ and $B_{l+1} \notin \mathcal{B}_2$ (note that it may be $l = 0$). Let $j \leq l + 1$ be the greatest index such that $B_1, \ldots, B_j$ do not contain a positive triangle in $\mathcal{T}$ and B_{j+1} does; if $B_1, \ldots, B_{l+1}$ do not contain a positive triangle in $\mathcal{T}$, simply let $j = l + 1$. Delete $\cup_{i=0}^{j-1} V(B_i) \cup (B_j)_{\mathsf{int}}$ and call G' the resulting graph. Observe that $\mathcal{B}'_1 \subseteq (\mathcal{B}_1 \cup \{B_{j+1}\}) \setminus \{B\}$ if $j \leq l$ and $\mathcal{B}'_1 = \mathcal{B}_1 \setminus \{B\}$ if $j = l + 1$. In both cases $G' - S$ is reduced under Reduction Rule 5.58 and we may apply the induction hypothesis, which says that $G' - U$ has at most $2|\mathcal{T}'| + |S|$ components, where $\mathcal{T}'$ is the set of positive triangles in

$\mathcal{T}$ which are contained in $G' - S$.

If B contains a positive triangle in $\mathcal{T}$, then $2|\mathcal{T}'| + |S| \leq 2|\mathcal{T}| - 2 + |S|$, and observe that $G - U$ has at most 2 components more than $G' - U$ (one containing $B - U$ and the other containing $V(B_1) \cup \cdots \cup V(B_j)$). If on the other hand B_{int} contains a neighbour of S and B does not contain a positive triangle in $\mathcal{T}$, then the component of $G - U$ containing $V(B_0) \cup \cdots \cup V(B_j)$ also contains a vertex in S, and there are at most $|S|$ of these components. This completes the induction step.

Finally, using Lemma 5.18 we can see that $\mathsf{ex}(G) \geq \mathsf{ex}(G[U]) + \mathsf{ex}(G - U) - \frac{1}{4}(|\mathfrak{C}(G[U])| + |\mathfrak{C}(G - U)| - 1) \geq \frac{1}{4}(|\mathcal{T}| - |S| + 1)$. Hence if $|\mathcal{T}| \geq |S| - 1 + 4k$, then the instance is a `YES`-instance. Otherwise, $|\mathcal{T}| \leq 16k$ and $|U| \leq 48k$. In this case, form S' adding U to S and note that $|S'| \leq 60k$ and $G - S'$ is a forest of cliques which does not contain a positive triangle (because of the way $\mathcal{T}$ was constructed). To conclude it is enough to apply Theorem 5.55 to $G - S'$, which is possible because a forest of cliques is a chordal graph. $\square$

From now on, we may assume that $(G = (V, E), k)$ is an instance of Signed Max Cut APT and $S \subseteq V$ is a set of at most $\mathcal{O}(k)$ vertices such that $G - S$ is a forest of cliques which does not contain positive edges.

Observe that solving the problem on subgraphs of $G - S$ is equivalent to solving Max Cut APT. We will use this equivalence in multiple occasions in our proofs, mainly to find largest solutions on complete graphs.

We now state the reduction rules that will be used to obtain the kernel.

Reduction Rule 5.61. *Let B be a block in $G - S$. If there exists $U \subseteq B_{int}$ such that $|U| > \frac{|V(B)| + |N_G(U) \cap S|}{2} \geq 1$, $N^+_G(u) \cap S = N^+_G(U) \cap S$ and $N^-_G(u) \cap S = N^-_G(U) \cap S$ for all $u \in U$, then delete two arbitrary vertices $u_1, u_2 \in U$ and set $k' = k$.*

Reduction Rule 5.62. *Let B be a block in $G - S$. If $|V(B)|$ is even and there exists $U \subseteq B_{int}$ such that $|U| = \frac{|V(B)|}{2}$ and $N_G(U) \cap S = \emptyset$, then delete a vertex $u \in U$ and set $k' = k - \frac{1}{4}$.*

Reduction Rule 5.63. *Let B be a block in $G - S$ with vertex set $\{u, v, w\}$, such that $N_G(w) = \{u, v\}$. If the edge uv is a bridge in $G - \{w\}$, delete $V(B)$, add a new vertex z, positive edges $\{zx : x \in N^+_{G-\{w\}}(\{u, v\})\}$, negative edges $\{zx : x \in N^-_{G-\{w\}}(\{u, v\})\}$ and set $k' = k$. Otherwise, delete w and the edge uv and set $k' = k - \frac{1}{4}$.*

Reduction Rule 5.64. *Let C be a connected component of $G - S$ only adjacent to a vertex $s \in S$. Form a* Max Cut with Weighted Vertices *instance on C (see Section 5.1.1) by* *defining* $w_0(v) = 1$ *if* $v \in N_G^+(s) \cap C$ *(*$w_0(v) = 0$ *otherwise) and* $w_1(v) = 1$ *if* $v \in N_G^-(s) \cap C$ *(*$w_1(v) = 0$ *otherwise). Let* $p \in \mathbb{N}$ *be the maximum integer such that* (C, w_0, w_1, p) *is a* YES*-instance. Then delete* C *and set* $k' = k - p + \mathsf{pt}(V(C) \cup \{s\})$*.*

To prove that these rules are valid, we need the following two lemmas.

Lemma 5.65. *Let* B *be a block in* $G - S$*. If there exists* $U \subseteq B_{\mathit{int}}$ *such that* $|U| \geq \frac{|V(B)|}{2}$*, then there exists a* (V_1, V_2)*-balanced subgraph* H *of* G *with* $\mathsf{opt}(G)$ *edges such that at least one of the following inequalities holds:*

- $|V_2 \cap V(B)| \leq |V_1 \cap V(B)| \leq |N_G(U) \cap S| + |V_2 \cap V(B)|$*;*
- $|V_2 \cap V(B)| \leq |V_1 \cap V(B)| \leq |V_2 \cap V(B)| + 1$*.*

Proof. Let H be a (V_1, V_2)-balanced subgraph of G with $\mathsf{opt}(G)$ edges for some partition V_1, V_2 of V. We may assume that $|V_1 \cap V(B)| \geq |V_2 \cap V(B)|$. Note that if $|V_1 \cap V(B)| > |V_2 \cap V(B)|$, then $U \cap V_1 \neq \emptyset$ (because $|U| \geq \frac{|V(B)|}{2}$).

First, if $N_G(U) \cap S = \emptyset$ and $|V_1 \cap V(B)| \geq |V_2 \cap V(B)| + 2$, then, for any $u \in U \cap V_1$, the balanced subgraph induced by the partition $V_1 \setminus \{u\}, V_2 \cup \{u\}$ has more edges than the balanced subgraph induced by (V_1, V_2), which is a contradiction.

Now, suppose that $N_G(U) \cap S \neq \emptyset$ and suppose also that $|V_1 \cap V(B)| - |V_2 \cap V(B)|$ is minimal. If $|V_1 \cap V(B)| \leq |V_2 \cap V(B)| + 1$ we are done, so suppose $|V_1 \cap V(B)| \geq |V_2 \cap V(B)| + 2$. Consider the partition $V_1' = V_1 \setminus \{u\}$, $V_2' = V_2 \cup \{u\}$, where $u \in V_1 \cap U$, and the balanced subgraph H' induced by this partition. Then $|E(H')| \geq |E(H)| + |E(V_1 \setminus \{u\}, \{u\})| - |E(V_2, \{u\})| \geq |E(H)| + (|V_1 \cap V(B)| - 1 - |N_G(U) \cap S| - |V_2 \cap V(B)|)$. Since $|V_1' \cap V(B)| - |V_2' \cap V(B)| < |V_1 \cap V(B)| - |V_2 \cap V(B)|$, it holds that $|E(H')| \leq |E(H)| - 1$. Therefore, $|V_1 \cap V(B)| \leq |N_G(U) \cap S| + |V_2 \cap V(B)|$. □

Lemma 5.66. *Let* B *be a block in* $G - S$*. If there exists* $U \subseteq B_{\mathit{int}}$ *such that* $|U| > \frac{|V(B)| + |N_G(U) \cap S|}{2}$*,* $N_G^+(u) \cap S = N_G^+(U) \cap S$ *and* $N_G^-(u) \cap S = N_G^-(U) \cap S$ *for all* $u \in U$*, then, for any* $u_1, u_2 \in U$*, there exists a* (V_1, V_2)*-balanced subgraph* H *of* G *with* $\mathsf{opt}(G)$ *edges such that* $u_1 \in V_1$ *and* $u_2 \in V_2$*.*

Proof. First, we claim that there exist vertices $u_1, u_2 \in U$ for which the result holds. Let H be a (V_1, V_2)-balanced subgraph of G with $\mathsf{opt}(G)$ edges as given by Lemma 5.65.

Suppose $N_G(U) \cap S = \emptyset$. Then, by Lemma 5.65 it holds that $|V_2 \cap V(B)| \leq |V_1 \cap V(B)| \leq |V_2 \cap V(B)| + 1$; in addition, $|U| > \frac{|V(B)|}{2}$. Hence, either we can find u_1 and u_2 as required, or $U = V_1 \cap V(B)$ and $|V_1 \cap V(B)| = |V_2 \cap V(B)| + 1$. In the second case, pick a vertex $u \in U$

and form the partition $V_1' = V_1 \setminus \{u\}$ and $V_2' = V_2 \cup \{u\}$. Consider the balanced subgraph H' induced by this partition. Observe that $|E(H')| = |E(H)| - |E(\{u\}, V_2)| + |E(\{u\}, V_1 \setminus \{u\})| = |E(H)| - |V_2 \cap V(B)| + |V_1 \cap V(B)| - 1 = |E(H)|$, so H' is a balanced subgraph of size $\mathsf{opt}(G)$ for which we can find u_1 and u_2 as required.

Now, suppose $N_G(U) \cap S \neq \emptyset$. Then by Lemma 5.65 it holds that $|V_2 \cap V(B)| \leq |V_1 \cap V(B)| \leq |N_G(U) \cap S| + |V_2 \cap V(B)|$. For the sake of contradiction, suppose $U \subseteq V_1 \cap V(B)$ or $U \subseteq V_2 \cap V(B)$: in both cases, this means that $|U| \leq |V_1 \cap V(B)|$. Note that $|V(B)| = |V_1 \cap V(B)| + |V_2 \cap V(B)| = 2|V_2 \cap V(B)| + t$, where $t \leq |N_G(U) \cap S|$. Hence, $|V_1 \cap V(B)| \geq |U| > \frac{|V(B)| + |N_G(U) \cap S|}{2} = |V_2 \cap V(B)| + \frac{t}{2} + \frac{|N_G(U) \cap S|}{2} \geq |V_2 \cap V(B)| + t = |V_1 \cap V(B)|$, which is a contradiction.

To conclude the proof, notice that for a (V_1, V_2)-balanced subgraph H of G with $\mathsf{opt}(G)$ edges and vertices $u_1, u_2 \in U$ such that $u_1 \in V_1$ and $u_2 \in V_2$, we have $|E(H)| = |E(H')|$, where H' is a balanced subgraph induced by $V_1' = V_1 \setminus \{u_1\} \cup \{u_2\}$ and $V_2' = V_2 \setminus \{u_2\} \cup \{u_1\}$: this is true because $N_G^+(u_1) \cap S = N_G^+(u_2) \cap S$ and $N_G^-(u_1) \cap S = N_G^-(u_2) \cap S$. □

Theorem 5.67. *Reduction Rules 5.61-5.64 are valid.*

Proof. It is not difficult to see that all rules can be applied in polynomial time (for Rule 5.64, this is true because Max Cut with Weighted Vertices can be solved in polynomial time on a forest of cliques, see Lemma 5.15). Moreover, each of them reduces the order of the graph, hence they can be applied at most $\mathcal{O}(|V|)$ times.

Rule 5.61: Let B, U be as in the description of Rule 5.61. Let $u_1, u_2 \in U$. By Lemma 5.66, there exists a (V_1, V_2)-balanced subgraph H of G with $\mathsf{opt}(G)$ edges such that $u_1 \in V_1$ and $u_2 \in V_2$. Now, let $G' = G - \{u_1, u_2\}$ and $H' = H - \{u_1, u_2\}$, and note that G' is connected. Since $N_G^+(u_1) \cap S = N_G^+(u_2) \cap S$ and $N_G^-(u_1) \cap S = N_G^-(u_2) \cap S$, it holds that $|E(H)| = |E(H')| + \frac{|E(G, \{u_1, u_2\})|}{2} + 1$, and so $\mathsf{opt}(G') + \frac{|E(G, \{u_1, u_2\})|}{2} + 1 \geq \mathsf{opt}(G)$. Conversely, by Lemma 5.18, $\mathsf{opt}(G) \geq \mathsf{opt}(G') + \frac{|E(G, \{u_1, u_2\})|}{2} + 1$. Finally, observe that $\mathsf{pt}(G) = \mathsf{pt}(G') + \frac{|E(G, \{u_1, u_2\})|}{2} + 1$, which implies that $\mathsf{ex}(G) = \mathsf{ex}(G')$.

Rule 5.62: Let B, U be as in the description of Rule 5.62. Let $u \in U$. By Lemma 5.65, there exists a (V_1, V_2)-balanced subgraph H of G with $\mathsf{opt}(G)$ edges, such that $|V_1 \cap V(B)| = |V_2 \cap V(B)|$. Consider the graph $G' = G - \{u\}$ formed by the application of the rule and the balanced subgraph $H' = H - \{u\}$, and note that G' is connected. Then $|E(H)| = |E(H')| + \frac{|V(B)|}{2}$, and thus $\mathsf{opt}(G') \geq \mathsf{opt}(G) - \frac{|V(B)|}{2}$. Conversely, by Lemma 5.18, $\mathsf{opt}(G) \geq \mathsf{opt}(G') + \frac{|V(B)|}{2}$. However, $\mathsf{pt}(G) = \mathsf{pt}(G') + \frac{|V(B)|}{2} - \frac{1}{4}$. Hence, $\mathsf{ex}(G) = \mathsf{ex}(G') + \frac{1}{4}$.

Rule 5.63: Let B and $\{u, v, w\}$ be as in the description of Rule 5.63, and let G' be the graph obtained after an application of the rule. Firstly consider the case when uv is a bridge in $G - \{w\}$. Note that the vertex z which replaces u and v is well-defined as in this case u and v have no common neighbour apart w; in addition, G' is connected. For any maximal balanced subgraph H of G (that is, a balanced subgraph which is not properly contained in a larger balanced subgraph of G), without loss of generality one may assume that $uw, vw \in E(H)$ and $uv \notin E(H)$. Suppose H is induced by a partition (V_1, V_2) and $u, v \in V_1$. Form a balanced subgraph of G' from $H - \{u, v, w\}$ by placing z in V_1. Therefore, $\mathsf{opt}(G) = \mathsf{opt}(G') + 2$. Since $\mathsf{pt}(G) = \mathsf{pt}(G') + \frac{3}{2} + \frac{2}{4} = \mathsf{pt}(G') + 2$, it follows that $\mathsf{ex}(G) = \mathsf{ex}(G')$.

Now consider the case when uv is not a bridge in $G - \{u\}$. Then the graph G' formed by deleting the vertex w and the edge uv is connected. Furthermore, regardless of whether u and v are in the same partition that induces a balanced subgraph H' of G', H' can be extended to a balanced subgraph H of G such that $|E(H)| = |E(H')| + 2$. This means that, as before, $\mathsf{opt}(G) = \mathsf{opt}(G') + 2$. But in this case $\mathsf{pt}(G) = \mathsf{pt}(G') + \frac{7}{4}$ and thus $\mathsf{ex}(G) = \mathsf{ex}(G') + \frac{1}{4}$.

Rule 5.64: Let C and $s \in S$ be as in the description of Rule 5.64. Solving Max Cut with Weighted Vertices gives $\mathsf{opt}(G[V(C) \cup \{s\}])$. Moreover, s is a cutvertex, hence by Lemma 5.5, $\mathsf{ex}(G) = \mathsf{ex}(G - C) + \mathsf{ex}(G[V(C) \cup \{s\}]) = \mathsf{ex}(G - C) + p - \mathsf{pt}(V(C) \cup \{s\})$. □

As in Section 5.2, we will bound the number of vertices in each block and the total number of blocks. The difference is that in this case we will make use of the specific properties of Signed Max Cut to locally solve the instance in an optimal way.

Henceforth, we assume that the instance (G, S, k) is such that G is reduced under Reduction Rules 5.61-5.64, $G - S$ is a forest of cliques which does not contain a positive edge and S contains at most $\mathcal{O}(k)$ vertices.

We begin with two results about the neighbours of S in the interior of blocks in $\mathcal{B}_0 \cup \mathcal{B}_1 \cup \mathcal{B}_2$. For a block $B \in \mathcal{B}$, let $B_{\mathsf{ext}} = V(B) \setminus B_{\mathsf{int}}$.

Lemma 5.68. *For every block $B \in \mathcal{B}_0 \cup \mathcal{B}_1$, $N_G(B_{\mathsf{int}}) \cap S \neq \emptyset$. Furthermore, if C is a component of $G - S$ and $|N_G(S) \cap V(C)| = 1$, then C consists of a single vertex.*

Proof. We start by proving the first claim. Note that if $B \in \mathcal{B}_0$ consists of a single vertex, then $N_G(B_{\mathsf{int}}) \cap S \neq \emptyset$ since G is connected, and if $B \in \mathcal{B}_1$, then it contains at least two vertices. So assume that B has at least two vertices. Suppose that $N_G(B_{\mathsf{int}}) \cap S = \emptyset$ and let $U = B_{\mathsf{int}}$. Then if $|B_{\mathsf{int}}| > |B_{\mathsf{ext}}|$, Rule 5.61 applies. If $|B_{\mathsf{int}}| = |B_{\mathsf{ext}}|$ then Rule 5.62 applies. Otherwise, $|B_{\mathsf{int}}| < |B_{\mathsf{ext}}|$, and, since $|B_{\mathsf{ext}}| \leq 1$, B has only one vertex, which contradicts our assumption above. For the second claim, first note that since $|N_G(S) \cap V(C)| = 1$, C consists of a single block. Let $N_G(S) \cap V(C) = \{v\}$ and $U = V(C) - \{v\}$. If $|U| > 1$, Rule 5.61 applies. If $|U| = 1$, Rule 5.62 applies. Hence $V(C) = \{v\}$. □

Let $\mathcal{B}_2'$ be the set of blocks B of $\mathcal{B}_2$ such that $|B_{\mathsf{ext}}| = 2$ and $|B_{\mathsf{int}}| = 0$.

Lemma 5.69. *If $B \in (\mathcal{B}_2 \setminus \mathcal{B}_2')$, then $N_G(B_{\mathit{int}}) \cap S \neq \emptyset$.*

Proof. Let $B \in \mathcal{B}_2$ and assume that $N_G(B_{\mathsf{int}}) \cap S = \emptyset$. Recall that by definition $|B_{\mathsf{ext}}| = 2$. If $|B_{\mathsf{int}}| > 2$, then Rule 5.61 applies. If $|B_{\mathsf{int}}| = 2$, then Rule 5.62 applies. If $|B_{\mathsf{int}}| = 1$, then Rule 5.63 applies. Hence it must hold that $|B_{\mathsf{int}}| = 0$ and $B \in \mathcal{B}_2'$. □

The next lemma provides a similar bound to the one given by Lemma 5.28 for diverging strongly λ-extendible properties. The proof is analogous.

Lemma 5.70. *If there exists a vertex $s \in S$ such that $\sum_{B\in\mathcal{B}} |N_G(B_{\mathit{int}}) \cap \{s\}| \geq 2(|S|-1+4k)$, then (G, k) is a* `YES`*-instance.*

Proof. Form $U' \subseteq N_G(s)$ by picking a vertex from each block B for which $|N_G(B_{\mathsf{int}}) \cap \{s\}| = 1$: if there exists a vertex $v \in B_{\mathsf{int}}$ such that $N_G(v) \cap S = \{s\}$, pick this, otherwise pick $v \in B_{\mathsf{int}}$ arbitrarily. Let $U = U' \cup \{s\}$ and $W = V \setminus U$.

Observe that $G[U]$ is connected and it is balanced because it is a tree. Thus $\mathsf{ex}(G[U]) = \frac{|U'|}{4}$. Now, consider a connected component C of $G - S$. By Rule 5.64, $|N_G(C) \cap S| \geq 2$ and by Lemma 5.68, if $|N_G(S) \cap V(C)| = 1$ then C consists of a single vertex. Otherwise, either $(N_G(S) \setminus N_G(s)) \cap V(C) \neq \emptyset$, or C has at least two vertices in U'. Moreover, note that the deletion of interior vertices does not disconnect the component itself. Hence $G[W]$ has at most $(|S| - 1) + \frac{|U'|}{2}$ connected components. Applying Lemma 5.18, $\mathsf{ex}(G) \geq \frac{|U'|}{4} - \frac{(|S|-1)+\frac{|U'|}{2}}{4} = \frac{|U'|}{8} - \frac{|S|-1}{4}$. Hence if $|U'| \geq 2(|S| - 1 + 4k)$, then (G, k) is a `YES`-instance. □

Corollary 5.71. *If $\sum_{B\in\mathcal{B}} |N_G(B_{\mathit{int}}) \cap S| \geq |S|(2|S| - 3 + 8k) + 1$, the instance is a* `YES`*-instance. Otherwise, $\sum_{B\in\mathcal{B}} |N_G(B_{\mathit{int}}) \cap S| \leq b_1k^2$ for some constant b_1.*

Proof. If $\sum_{B\in\mathcal{B}} |N_G(B_{\mathsf{int}}) \cap S| \geq |S|(2|S| - 3 + 8k) + 1$, then for some $s \in S$ we have $\sum_{B\in\mathcal{B}} |N_G(B_{\mathsf{int}}) \cap \{s\}| \geq 2|S| - 3 + 8k + \frac{1}{|S|}$ and, since the sum is integral, $\sum_{B\in\mathcal{B}} |N_G(B_{\mathsf{int}}) \cap \{s\}| \geq 2(|S| - 1 + 4k)$. Thus, by Lemma 5.70 (G, k) is a `YES`-instance. The second inequality of the corollary follows from the fact that $|S| \in \mathcal{O}(k)$. □

Corollary 5.71 can be used to bound the number of blocks in $\mathcal{B}_0 \cup \mathcal{B}_1 \cup (\mathcal{B}_2 \setminus \mathcal{B}_2')$, as by Lemma 5.68 and Lemma 5.69 all these blocks contain a neighbour of S in the interior. For the blocks in $\mathcal{B}_2'$ we will need the following lemma.

Lemma 5.72. *If in $G-S$ there exist vertices $U = \{u_1, u_2, \ldots, u_p\}$ such that $N_{G-S}(u_i) = \{u_{i-1}, u_{i+1}\}$ for $2 \le i \le p-1$, and $p \ge |S| + 4k + 1$, then (G,k) is* YES*-instance. Otherwise, $p \le b_2 k$ for some constant b_2.*

Proof. Observe that $G[U]$ is balanced since it is a tree. Thus $\mathsf{ex}(G[U]) = \frac{p-1}{4}$. Let $W = V \setminus U$ and observe that $G[W]$ has at most $|S|$ components, since by Lemma 5.68 for every vertex in $G-U$ there is a path to a vertex in S. Applying Lemma 5.18, $\mathsf{ex}(G) \ge \frac{p-1}{4} - \frac{|S|}{4}$. Hence if $p - 1 - |S| \ge 4k$, (G,k) is a YES-instance. The second inequality of the lemma follows from the fact that $|S| \in \mathcal{O}(k)$. □

Now we combine the results we have proved so far, and we also derive a bound on $|\mathcal{B}_{\ge 3}|$ using the structural result of Lemma 5.35.

Theorem 5.73. *$G-S$ contains at most $4b_1k^2$ blocks in $\mathcal{B} \setminus \mathcal{B}_2'$ and $4b_1b_2k^3$ blocks in $\mathcal{B}_2'$, or the instance is a* YES*-instance.*

Proof. By Lemma 5.68, Lemma 5.69 and Corollary 5.71, there are at most b_1k^2 blocks in $\mathcal{B}_0 \cup \mathcal{B}_1 \cup (\mathcal{B}_2 \setminus \mathcal{B}_2')$, or the instance is a YES-instance. Then, by Lemma 5.35, $|\mathcal{B}_{\ge 3}| \le 3b_1k^2$. As for the blocks in $\mathcal{B}_2'$, observe that each of them corresponds to a vertex of degree two in the block graph of $G-S$; in addition, by Lemma 5.72 there cannot be more than b_2k blocks in $\mathcal{B}_2'$ which correspond to adjacent vertices in the block graph, or the instance is a YES-instance. Hence there are at most $(4b_1k^2)(b_2k)$ blocks in $\mathcal{B}_2'$. □

It is only left to prove a bound on the size of a block. Next lemma will be used to this purpose.

Lemma 5.74. *For a block $B \in \mathcal{B}$, if $|V(B)| \ge 2|B_{ext}| + |N_G(B_{int}) \cap S|(2|S| + 8k + 1)$, then (G,k) is a* YES*-instance. Otherwise, $|V(B)| \le 2|B_{ext}| + b_3k|N_G(B_{int}) \cap S|$ for some constant b_3.*

Proof. Consider a fixed $s \in N_G(B_{\mathsf{int}}) \cap S$. We will show that we may assume that either $|N_G^+(s) \cap B_{\mathsf{int}}| \le \frac{4k+|S|}{2}$ or $|N_G^+(s) \cap B_{\mathsf{int}}| \ge |B_{\mathsf{int}}| - \frac{4k+|S|}{2}$, because otherwise (G,k) is a YES-instance.

Indeed, suppose $\lceil\frac{4k+|S|}{2}\rceil \leq |N_G^+(s) \cap B_{\mathsf{int}}| \leq |B_{\mathsf{int}}| - \lceil\frac{4k+|S|}{2}\rceil$. Let $U_1 \subseteq N_G^+(s) \cap B_{\mathsf{int}}$, $|U_1| = \lceil\frac{4k+|S|}{2}\rceil$, and let $U_2 \subseteq B_{\mathsf{int}} \setminus N_G^+(s)$, $|U_2| = \lceil\frac{4k+|S|}{2}\rceil$. Let $U = U_1 \cup U_2 \cup \{s\}$ and consider the subgraph H of $G[U]$ induced by the edges $E(U_1, U_2) \cup E(s, U_1) \cup E(s, U_2)$. Observe that H is $(U_1 \cup \{s\}, U_2)$-balanced and so $\mathsf{opt}(G[U]) \geq |U_1|^2 + |U_1| + |U_2 \cap N_G^-(s)|$. Furthermore, $\mathsf{pt}(G[U]) = |U_1|^2 + \frac{|U_1|}{2} + \frac{|U_2 \cap N_G^-(s)|}{2}$, and hence $\mathsf{ex}(G[U]) \geq \frac{|U_1| + |U_2 \cap N_G^-(s)|}{2} \geq \frac{4k+|S|}{4}$.

Now consider $W = V \setminus U$. Any connected component of $G - S$ is connected to two vertices in S, hence $G[W]$ has at most $|S| - 1$ components adjacent to vertices in $S \setminus \{s\}$ and one component corresponding to the block B. Applying Lemma 5.18, $\mathsf{ex}(G) \geq \frac{(4k+|S|)-|S|}{4}$, which means that (G, k) is a `YES`-instance.

Similarly, we can show that we may assume that either $|N_G^-(s) \cap B_{\mathsf{int}}| \leq \frac{4k+|S|}{2}$ or $|N_G^-(s) \cap B_{\mathsf{int}}| \geq |B_{\mathsf{int}}| - \frac{4k+|S|}{2}$, because otherwise (G, k) is a `YES`-instance.

Let $S_1^+ = \{s \in S : 0 < |N_G^+(s) \cap B_{\mathsf{int}}| \leq \frac{k+|S|}{2}\}$, $S_2^+ = (N_G^+(B_{\mathsf{int}}) \cap S) \setminus S_1^+$ and

$$U^+ = (B_{\mathsf{int}} \setminus N_G^+(S_1^+)) \bigcap (\cap_{s \in S_2^+} N_G^+(s))$$

Observe that for all $s \in S_2^+$, $|N_G^+(s) \cap B_{\mathsf{int}}| \geq |B_{\mathsf{int}}| - \frac{4k+|S|}{2}$, which means that $|U^+| \geq |B_{\mathsf{int}} \setminus N_G^+(S_1^+)| - |S_2^+|\frac{4k+|S|}{2}$. In addition, $|N_G^+(S_1^+) \cap B_{\mathsf{int}}| \leq |S_1^+|\frac{4k+|S|}{2}$, hence $|B_{\mathsf{int}} \setminus N_G^+(S_1^+)| \geq |B_{\mathsf{int}}| - |S_1^+|\frac{4k+|S|}{2}$. Therefore, $|U^+| \geq |B_{\mathsf{int}}| - (|S_1^+| + |S_2^+|)\frac{4k+|S|}{2} = |B_{\mathsf{int}}| - |N_G^+(B_{\mathsf{int}}) \cap S|\frac{4k+|S|}{2} \geq |B_{\mathsf{int}}| - |N_G(B_{\mathsf{int}}) \cap S|\frac{4k+|S|}{2}$.

With similar definitions and the same argument we obtain $|U^-| \geq |B_{\mathsf{int}}| - |N_G(B_{\mathsf{int}}) \cap S|\frac{4k+|S|}{2}$. Now let $U = U^+ \cap U^-$ and observe that and $|U| \geq |B_{\mathsf{int}}| - |N_G(B_{\mathsf{int}}) \cap S|(4k + |S|)$.

However, by Rule 5.61, $|U| \leq \frac{|V(B)| + |N_G(B_{\mathsf{int}}) \cap S|}{2}$. So, $|B_{\mathsf{int}}| \leq |N_G(B_{\mathsf{int}}) \cap S|(|S| + 4k + \frac{1}{2}) + \frac{|V(B)|}{2}$, and so $|V(B)| \leq 2|B_{\mathsf{ext}}| + |N_G(B_{\mathsf{int}}) \cap S|(2|S| + 8k + 1)$ as claimed. The second inequality of the lemma follows from the fact that $|S| \in \mathcal{O}(k)$. □

Finally, we can prove the main result of this section.

Theorem 5.75. Signed Max Cut APT *admits a kernel with $\mathcal{O}(k^3)$ vertices.*

Proof. Let $(G = (V, E), k)$ be an instance of Signed Max Cut APT. By Theorem 5.57 in polynomial time it is possible either to answer `YES` or to produce a set $S \subseteq V$ such that $|S| \leq 12k$ and $G - S$ is a forest of cliques. Apply Reduction Rule 5.58 as many times as possible (every time the number of vertices decreases, so it is possible to apply it at most $\mathcal{O}(n)$ times), and let (G', k') be the resulting instance. By Theorem 5.60, in polynomial time it is possible either to answer `YES`, or to find a set $S' \supseteq S$ of vertices such that $G' - S'$ is a forest of cliques which does not contain positive edges, and $|S'| \leq 60k$.

Now, apply Rules 5.61–5.64 exhaustively to (G', S', k') to obtain a new instance (G'', S', k''). If $k'' \leq 0$, then (G, k) is a YES-instance since Rules 5.61–5.64 are valid. Now let $G = G''$, $S = S'$ and $k = k''$. Check whether (G, k) is a YES-instance due to Corollary 5.71, Lemma 5.72 or Lemma 5.74. If this is not the case, by Theorem 5.73, $G - S$ contains at most $4b_1k^2$ blocks in $\mathcal{B} \setminus \mathcal{B}_2'$ and $4b_1b_2k^3$ blocks in $\mathcal{B}_2'$. Also, note that Theorem 5.73 also implies that $|Q| \in \mathcal{O}(k^3)$, as in every graph the number of cutvertices is bounded by the number of blocks. Hence,

$$|S| + 8b_1b_2k^3 + \sum_{B \in \mathcal{B}\setminus\mathcal{B}_2'} |V(B)| \leq |S| + 8b_1b_2k^3 + 2\sum_{B \in \mathcal{B}\setminus\mathcal{B}_2'} |B_{\mathsf{ext}}| + b_3k \sum_{B \in \mathcal{B}\setminus\mathcal{B}_2'} |N_G(B_{\mathsf{int}}) \cap S|$$

Now, observe that $\sum_{B \in \mathcal{B}\setminus\mathcal{B}_2'} |B_{\mathsf{ext}}|$ is bounded by the sum of the degrees of the vertices of the block graph of $G - S$, which is bounded by $2(|\mathcal{B}| + |Q|) \in \mathcal{O}(k^3)$, as the block graph is a tree. Therefore, applying again Corollary 5.71, we obtain that $|V| \in \mathcal{O}(k^3)$. □

The kernelization for Max Cut APT follows immediately.

Theorem 5.76. *The* Above Poljak-Turzík (Π_{BP}) *problem admits a kernel with* $\mathcal{O}(k^3)$ *vertices.*

Proof. Let (G, k) be an instance of Above Poljak-Turzík (Π_{BP}). Transform it into an instance of Signed Max Cut APT by labelling every edge negative. Note that this is possible as a balanced subgraph in a graph where every edge is negative is a bipartite subgraph. Now, since none of Rules 5.61–5.64 increase the number of positive edges in the graph, then the instance (G', k') produced using Theorem 5.75 only contains negative edges, hence $(\mathcal{U}(G'), k')$ is an instance of Above Poljak-Turzík (Π_{BP}) equivalent to (G, k) and such that $|V(G')| \in \mathcal{O}(k^3)$. □

Chapter 6

Discussion and Future Work

In Chapter 3 we studied the k-CHINESE POSTMAN problem and we showed that it admits a kernel with $\mathcal{O}(k^2 \log k)$ vertices and $\mathcal{O}(k^2 \log k)$ edges. There exists a similar version of this problem on directed graphs, where one is asked to find k directed closed walks which contain every arc of the digraph and whose total weight is at most p. Note that for $k = 1$ this problem corresponds to the DIRECTED CHINESE POSTMAN PROBLEM, which is polynomial time solvable [32]. For the general k, it can be shown that the problem is NP-complete [47].

The DIRECTED k-CHINESE POSTMAN PROBLEM appears to be more difficult to solve than its undirected version. Recently, it has been proved by Gutin *et al.* that this problem admits an FPT-algorithm [45]. Nonetheless, the running time of their algorithm is bounded by a function $f(k)$ which is a multiply iterated exponential, where the number of iterations is also a multiply iterated exponential, so this result is of no practical interest. It would be desirable to improve the bound on the running time and, at the same time, to explore the possibility of the existence of a polynomial kernel.

In Chapter 4, we considered the TEST COVER problem under different choices of the parameter. The main results are the existence of a polynomial kernel for TEST-r-COVER$(n-k, k)$ and TEST-r-COVER$(m-k, k)$. In fact, for the latter we proved a stronger fact: TEST-r-COVER$(m-k, k)$ admits a polynomial kernel for the parameter $k+r$ (that is, r is not required to be a constant). It would be interesting to find out whether the same holds for TEST-r-COVER$(n-k, k)$. Also, sometimes kernelizations do not produce the most efficient FPT-algorithms, hence it would be interesting to devise an FPT-algorithm for TEST-r-COVER$(n-k, k)$ which is not based on the kernelization (as it was done for TEST-r-COVER$(m-k, k)$).

In Chapter 5, we studied the theoretical class of strongly λ-extendible properties and the associated WEIGHTED ABOVE POLJAK-TURZÍK (Π) problem. In their 1986 paper, Poljak and Turzík described various graph properties which are strongly λ-extendible: apart from the ones we considered in this thesis, 'being bipartite' and 'being balanced', there are the properties of 'being acyclic' and 'having a homomorphism into a vertex-transitive graph G_0'.

The former is strongly λ-extendible for $\lambda = \frac{1}{2}$ [72]. The associated $WAPT(\Pi)$ asks for an acyclic subgraph of weight at least $\mathsf{pt}(G) + k$ of an oriented graph G. The latter, instead, is strongly λ-extendible for $\lambda = \frac{d}{n_0}$, where n_0 is the number of vertices of G_0, and d is the minimum number of edges of the given label and the given orientation incident to any vertex of G_0 over all allowed labels and orientations [72]. The associated $WAPT(\Pi)$ asks for a subgraph of an oriented and/or labelled graph G such that its weight is at least $\mathsf{pt}(G) + k$, and it has a homomorphism into G_0.

Considering only graphs with integral weights, we showed that for any strongly λ-extendible property Π we can either solve an instance (G, k) of $WAPT(\Pi)$, or find a small set of vertices S such that $G - S$ has a simpler structure. We used this result to prove that WEIGHTED MAX CUT APT on graphs with integral weights is FPT. In his doctoral thesis [60], Jones built on the same result to prove that WEIGHTED SIGNED MAX CUT APT can be solved in FPT time on graphs with integral weights. It is still an open question whether it is possible to apply the result to devise an FPT-algorithm for $WAPT(\Pi)$ when Π is the property of 'being acyclic' or 'having a homomorphism into a vertex-transitive graph G_0' (note that for the unweighted case the question was positively answered by Mnich *et al.* [72]).

We also proved that the unweighted version of the problem, namely $APT(\Pi)$, admits a polynomial kernel for every strongly λ-extendible property Π, unless Π is strongly λ-extendible for $\lambda = \frac{1}{2}$, does not contain some labelling of the triangle, and either is not hereditary or it is a property of labelled graphs. This ensures that $APT(\Pi)$ admits a polynomial kernel when Π is the property of 'being acyclic' (note, however, that this result had already been proved by Crowston *et al.* [19]), and when Π is the property of 'having a homomorphism into a vertex-transitive graph G_0': in fact, in the former case Π is a hereditary property on unlabelled graphs, and in the latter Π is hereditary, and when G_0 contains at least two edges labelled in a different way then $\frac{d}{n_0} < \frac{1}{2}$.

The natural extensions of this result are on one side removing, if possible, the condition that Π should be an unlabelled property for $\lambda = \frac{1}{2}$, and on the other generalising the algorithm to produce a polynomial kernel for $WAPT(\Pi)$ on graphs with integral weights.

Appendix A

List of the problems

Here we give the definitions of the classical and parameterized problems that we mention in the thesis, adding some comments on their complexity when necessary.

A.1 Classical Problems

Definition 1.

A Boolean variable is a variable that can take only two values, either TRUE or FALSE. A Boolean formula is built from Boolean variables through the use of conjunctions, disjunctions, negations and parentheses. A formula is satisfiable if there exists an assignment of values to its variables that makes it TRUE.

Boolean Satisfiability (SAT)	
Input:	A Boolean formula.
Question:	Is the formula satisfiable?

This is a central problem in Computer Science and it was the first problem that was shown to be NP-complete (this result is known as the Cook-Levin theorem [17, 68, 41]).

Definition 2.

Let C_3 denote the cycle with three edges.

3-Cycle Partitioning

Input: A graph $G = (V, E)$ with $|E| = m$ divisible by 3.

Question: Is it possible to partition E into sets $E_1, \dots, E_{\frac{m}{3}}$ in such a way that $G[E_i]$ is isomorphic to C_3 for $i \in [\frac{m}{3}]$?

This problem is an NP-complete problem due to a classical result by Holyer [55].

Definition 3.

3-Dimensional Matching

Input: A pair (V, T) where V is a set containing n elements and $T \subseteq V \times V \times V$.

Question: Is there a subset $M \subseteq T$ such that $|M| = n$ and $v_i \neq v'_i$, $i = 1, 2, 3$, for all $(v_1, v_2, v_3), (v'_1, v'_2, v'_3) \in M$?

The set M is called a 3-dimensional matching.

This problem is one of the 21 NP-complete problems described by Karp [62]. Note that the version we consider is a restriction of the more general problem where T is a subset of $W \times X \times Y$, with W, X, Y being three sets of possibly different cardinality.

Definition 4.

Set Cover (optimisation version)

Input: A hypergraph $H = (V, \mathcal{E})$.

Question: Find the smallest subset $\mathcal{S} \subseteq \mathcal{E}$, if it exists, such that for every $v \in V$ there exists $e \in \mathcal{S}$ with $v \in e$.

This problem admits a polynomial time $\mathcal{O}(\log |V|)$-approximation [59], but no polynomial time $o(\log |V|)$-approximation unless $P = NP$ [79] and no polynomial time $(1 - \varepsilon) \log |V|$-approximation for any $\varepsilon > 0$ unless $NP \subseteq DTIME(n^{\log \log |V|})$ [37].

Definition 5.

Let $r \in \mathbb{N}$ be a constant. Let V^r denote the cartesian product of r copies of a set V.

r-Dimensional Matching

Input: A pair (V, T) where V is a set containing n elements and $T \subseteq V^r$.

Question: Is there a subset $M \subseteq T$ such that $|M| = n$ and $v_i \neq v'_i$, $i \in [r]$, for all $(v_1, \dots, v_r), (v'_1, \dots, v'_r) \in M$?

The set M is called an r-dimensional matching.

The fact that this problem is NP-complete follows from a reduction from 3-Dimensional Matching to r-Dimensional Matching with $r \geq 4$. Let (V, T) be an instance of 3-Dimensional Matching: for every $(v_1, v_2, v_3) \in T$ and every $v \in V$, add $(v_1, v_2, v_3, v, \ldots, v)$ to T' (where the copies of v are $r-3$). Then (V, T') is an equivalent instance of r-Dimensional Matching.

Definition 6.

Let P_3 denote the path with two edges.

P_3-Packing

Input:	A graph $G = (V, E)$ with $\lvert V \rvert = n$ divisible by 3.
Question:	Is it possible to partition V into sets $V_1, \ldots, V_{\frac{n}{3}}$ in such a way that $G[V_i]$ is isomorphic to P_3 for $i \in [\frac{n}{3}]$?

This problem is NP-complete [41].

A.2 Parameterized Problems

Definition 7.

k-Set Cover

Input:	A pair (H, k) where $H = (V, \mathcal{E})$ is a hypergraph and k is an integer.
Parameter:	k
Question:	Is there a subset $\mathcal{S} \subseteq \mathcal{E}$ with $\lvert \mathcal{S} \rvert \leq k$ such that for every $v \in V$ there exists $e \in \mathcal{S}$ with $v \in e$?

The set $\mathcal{S}$ is called a *set cover*. This problem is $W[2]$-complete [30].

Definition 8.

$(m - k)$-Set Cover

Input:	A pair (H, k) where $H = (V, \mathcal{E})$ is a hypergraph with $\lvert \mathcal{E} \rvert = m$ and k is an integer.
Parameter:	k
Question:	Is there a subset $\mathcal{S} \subseteq \mathcal{E}$ with $\lvert \mathcal{S} \rvert \leq m - k$ such that for every $v \in V$ there exists $e \in \mathcal{S}$ with $v \in e$?

This problem is a dual of k-Set Cover with the size function being the size of the edge set. It is $W[1]$-complete [73, 44].

Definition 9.

A Boolean formula is in conjunctive normal form if it is a conjunction of clauses, where each clause consists of a disjunction of literals, which are negated or non-negated Boolean variables. The weight of a satisfying truth assignment is the number of variables which are set to TRUE.

Weighted 2-CNF-Satisfiability	
Input:	A Boolean formula in conjunctive normal form whose clauses have size at most 2 and an integer k.
Parameter:	k
Question:	Is there a satisfying truth assignment which has weight exactly k?

This problem is $W[1]$-complete. In fact, it is sometimes used to define the class $W[1]$ [75].

Definition 10.

See Definition 9 for the notions of Boolean formula in conjunctive normal form and weight of a satisfying truth assignment.

Weighted CNF-Satisfiability	
Input:	A Boolean formula in conjunctive normal form and an integer k.
Parameter:	k
Question:	Is there a satisfying truth assignment which has weight exactly k?

This problem is $W[2]$-complete. In fact, it is sometimes used to define the class $W[2]$ [75].

Definition 11.

k-Vertex Cover	
Input:	A pair (G, k) where $G = (V, E)$ is a graph and k is an integer.
Parameter:	k
Question:	Is there $V' \subseteq V$ such that $\|V'\| \leq k$ and $N_G[V'] = V$?

This is probably the most classical problem in Parameterized Complexity. It admits a kernel with $2k - c \log k$ vertices for any constant c [67].

Definition 12.

k-INDEPENDENT SET

Input: A pair (G, k) where $G = (V, E)$ is a graph and k is an integer.

Parameter: k

Question: Is there $V' \subseteq V$ such that $|V'| \geq k$ and $G[V']$ does not contain any edge?

This is a dual problem of VERTEX COVER, with the size function being the number of vertices. It is $W[1]$-complete [75].

Definition 13.

DISJOINT CYCLE PACKING

Input: A pair (G, k) where $G = (V, E)$ is a graph and k is an integer.

Parameter: k

Question: Is it possible to partition E into sets $E_1, \ldots, E_{k+1}$ such that $G[E_i]$ is a cycle for $i \in [k]$?

This problem admits a kernel with at most $\mathcal{O}(k \log k)$ vertices [10].

Bibliography

[1] F.N. Abu-Khzam, M.A. Langston, P. Shanbhag, and C.T. Symons. Scalable Parallel Algorithms for FPT Problems. Algorithmica 45 (3), pages 269-284 (2006). 10

[2] N. Alon, G. Gutin, E.J. Kim, S. Szeider, and A. Yeo. Solving MAX-k-SAT Above a Tight Lower Bound. Algorithmica 61 (3), pages 638-655 (2011). 12

[3] N. Alon, S. Hoory, and N. Linial. The Moore bound for irregular graphs. Graphs and Combinatorics 18 (1), pages 53-57 (2002). 40

[4] M. Basavaraju, M.C. Francis, M.S. Ramanujan, and S. Saurabh. Partially Polynomial Kernels for Set Cover and Test Cover. FSTTCS 2013, pages 67-78 (2013). 50

[5] N. Betzler, J. Guo, C. Komusiewicz, and R. Niedermeier. Average parameterization and partial kernelization for computing medians. Journal of Computer and System Sciences 77 (4), pages 774-789 (2011). 50

[6] H.L. Bodlaender, B.M.P. Jansen, and S. Kratsch. Cross-composition: A new technique for kernelization lower bounds. STACS 2011, pages 165-176 (2011). 19

[7] H.L. Bodlaender, B.M.P. Jansen, and S. Kratsch. Kernelization lower bounds by cross-composition. SIAM Journal on Discrete Mathematics 28 (1), pages 277-305 (2014). 23

[8] H.L. Bodlaender, E.D. Demaine, M.R. Fellows, J. Guo, D. Hermelin, D. Lokshtanov, M. Müller, V. Raman, J. van Rooij, and F.A. Rosamond. Open problems in parameterized and exact computation. IWPEC 2008 (2008). 13

[9] H.L. Bodlaender, R.G. Downey, M.R. Fellows, and D. Hermelin. On problems without polynomial kernels. Journal of Computer and System Sciences 75 (8), pages 423-434 (2009). 19, 23

[10] H.L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. Lecture Notes in Computer Science 5757, pages 635-646 (2009). 19, 39, 121

[11] B. Bollobàs and A. Scott. Better bounds for Max Cut. Contemporary Combinatorics, Bolyai Society Mathematical Studies 10, pages 185-246 (2002). 72

[12] J.Y. Cai, V.T. Chakaravarthy, L.A. Hemaspaandra, and M. Ogihara. Competing Provers Yield Improved Karp-Lipton Collapse Results. Lecture Notes in Computer Science 2607, pages 535-546 (2003). 22

[13] J. Chen, H. Fernau, I.A. Kanj and G. Xia. Parametric Duality and Kernelization: Lower Bounds and Upper Bounds on Kernel Size. Lecture Notes in Computer Science 3404, pages 269-280 (2005). 18, 50

[14] C. Chiang, A.B. Kahng, S. Sinha, X. Xu, and A.Z. Zelikovsky. Fast and efficient bright-field AAPSM conflict detection and correction. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 26 (1), pages 115-126 (2007). 103

[15] A. Cobham (1965). The intrinsic computational difficulty of functions. Studies in logic and the foundations of mathematics, North-Holland, pages 24-30 (1965). 7

[16] D. Cohen, J. Crampton, A. Gagarin, G. Gutin, and M. Jones. Engineering Algorithms for Workflow Satisfiability Problem with User-Independent Constraints. Lecture Notes in Computer Science Volume 8497, pages 48-59 (2014). 10

[17] S.A. Cook. The Complexity of Theorem-Proving Procedures. STOC 1971, pages 151-158 (1971). 117

[18] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, A. Schrijver. Combinatorial Optimization. Wiley (1997). 38

[19] R. Crowston, G. Gutin, and M. Jones. Directed Acyclic Subgraph Problem Parameterized above Poljak-Turzík Bound. FSTTCS 2012: LIPICS 18, pages 400-411 (2012). 116

[20] R. Crowston, G. Gutin, M. Jones, S. Saurabh, and A. Yeo. Parameterized Study of the Test Cover Problem. Lecture Notes in Computer Science 7464, 283-295 (2012). 30, 50, 51, 56

[21] R. Crowston, G. Gutin, M. Jones, and G. Muciaccia. Maximum balanced subgraph problem parameterized above lower bound. Theoretical Computer Science 513, pages 53-64 (2013). 31, 77, 104

[22] R. Crowston, G. Gutin, M. Jones, G. Muciaccia, and A. Yeo. Parameterizations of test cover with bounded test sizes. CoRR, abs/1209.6528 (2012). 31, 45

[23] R. Crowston, M. Jones, and M. Mnich. Max-Cut Parameterized above the Edwards-Erdős Bound. Automata, Languages, and Programming Lecture Notes in Computer Science 7391, pages 242-253 (2012). 72, 84

[24] R. Crowston, M. Jones, G. Muciaccia, G. Philip, A. Rai, and S. Saurabh. Polynomial Kernels for λ-extendible Properties Parameterized Above the Poljak-Turzík Bound. FSTTCS 2013, pages 43-54 (2013). 31

[25] B. DasGupta, G.A. Enciso, E.D. Sontag, and Y. Zhang. Algorithmic and complexity results for decompositions of biological networks into monotone subsystems. Biosystems 90 (1), pages 161-178 (2007). 103

[26] K.M.J. De Bontridder, B.V. Halldórsson, M.M. Halldórsson, C.A.J. Hurkens, J.K. Lenstra, R. Ravi, and L. Stougie. Approximation algorithms for the test cover problem. Mathematical Programming 98 (1-3), pages 477-491 (2003). 30, 43, 48, 51

[27] K.M.J. De Bontridder, B.J. Lageweg, J.K. Lenstra, J.B. Orlin, and L. Stougie. Branch-and-Bound Algorithms for the Test Cover Problem. Lecture Notes in Computer Science 2461, pages 223-233 (2002). 42

[28] H. Dell and D. van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. STOC 2010, pages 251-260 (2010). 23

[29] R. Diestel. Graph Theory, 4th Edition. Graduate texts in mathematics 173, Springer (2012). 25

[30] R. G. Downey and M. R. Fellows. Fundamentals of Parameterized Complexity. Springer (2013). 9, 10, 14, 119

[31] A. Drucker. New limits to classical and quantum instance compression. FOCS 2012, pages 609-618 (2012). 23

[32] J. Edmonds and E.L. Johnson. Matching Euler tours and the Chinese postman problem. Mathematical Programming 5 (1), pages 88-124 (1973). 33, 115

[33] C.S. Edwards. An improved lower bound for the number of edges in a largest bipartite subgraph. Recent Advances in Graph Theory, pages 167-181 (1975). 72

[34] C.S. Edwards. Some extremal properties of bipartite subgraphs. Canadian Journal of Mathematics 25, pages 475-485 (1973). 72

[35] P. Erdős. On some extremal problems in graph theory. Israel Journal of Mathemathics 3, pages 113-116 (1965). 72

[36] L. Euler. Solutio problematis ad geometriam situs pertinentis. Commentarii Academiae Petropolitanae 8, pages 128-140 (1741). 32, 33

[37] U. Feige. A threshold of $\ln n$ for approximating set cover. Journal of the ACM 45 (4), pages 634-652 (1998). 118

[38] H. Fernau, F.V. Fomin, D. Lokshtanov, D. Raible, S. Saurabh, and Y. Villanger. Kernel(s) for problems with no kernel: On out-trees with many leaves. STACS 2009, pages 421-432 (2009). 13

[39] J. Flum and M. Grohe. Parameterized Complexity Theory. Springer-Verlag (2006). 14, 15, 16

[40] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. Journal of Computer and System Sciences 77 (1), pages 91-106 (2011). 19

[41] M.R. Garey and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman & Co. New York, NY, USA © (1979). 43, 117, 119

[42] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM 42 (6), pages 1115-1145 (1995). 72

[43] N. Gülpınar, G. Gutin, G. Mitra, and A. Zverovitch. Extracting Pure Network Submatrices in Linear Programs Using Signed Graphs. Discrete Applied Mathematics 137, pages 359-372 (2004). 104

[44] G. Gutin, M. Jones, and A. Yeo. Kernels for Below-Upper-Bound Parameterizations of the Hitting Set and Directed Dominating Set Problems. Theoretical Computer Science 412, pages 5744-5751 (2011). 51, 120

[45] G. Gutin, M. Jones, B. Sheng, and M. Wahlström. Parameterized Directed k-Chinese Postman Problem and k Arc-Disjoint Cycles Problem on Euler Digraphs. CoRR, abs/1402.2137 (2014). 115

[46] G. Gutin, G. Muciaccia, and A. Yeo. (Non-)existence of polynomial kernels for the Test Cover problem. Information Processing Letters 113 (4), pages 123-126 (2013). 31

[47] G. Gutin, G. Muciaccia, and A. Yeo. Parameterized complexity of k-Chinese Postman Problem. Theoretical Computer Science 513, pages 124-128 (2013). 31, 115

[48] D.J. Haglin and S.M. Venkatesan. Approximation and intractability results for the maximum cut problem and its variants. IEEE Transactions on Computers 40 (1), pages 110-113 (1991). 72

[49] B.V. Halldórsson, J.S. Minden, and R. Ravi. PIER: Protein identification by epitope recognition. Currents in Computational Molecular Biology, pages 109-110 (2001). 30, 43

[50] B.V. Halldórsson, M.M. Halldórsson, and R. Ravi. On the approximability of the Minimum Test Collection problem. Lecture Notes in Computer Science 2161, 158-169 (2001). 30, 43, 46, 51

[51] F. Harary. On the notion of balance of a signed graph. Michigan Mathematical Journal 2 (2), pages 143-146 (1953). 103

[52] D. Harnik and M. Naor. On the compressibility of NP instances and cryptographic applications. SIAM Journal on Computing 39 (5), pages 1667-1713 (2010). 11

[53] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. Transactions of the American Mathematical Society 117, pages 285-306 (1965). 7

[54] C. Hierholzer. Ueber die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. Mathematische Annalen 6 (1), pages 30-32 (1873). 32

[55] I. Holyer. The NP-Completeness of Some Edge-Partition Problems. SIAM Journal on Computing 10 (4), pages 713-717 (1981). 118

[56] F. Hüffner, N. Betzler, and R. Niedermeier. Optimal edge deletions for signed graph balancing. Lecture Notes in Computer Science 4525, pages 297-310 (2007). 103

[57] F. Hüffner, R. Niedermeier, and S. Wernicke. Techniques for Practical Fixed-Parameter Algorithms. The Computer Journal 51 (1), pages 7-25 (2008). 10

[58] A. Itai, M. Rodeh. Finding a minimum circuit in a graph. SIAM Journal on Computing 7 (4), pages 413-423 (1978). 39, 40

[59] D.S. Johnson. Approximation algorithms for combinatorial problems. Journal of Computer and System Sciences 9, pages 256-278 (1972). 118

[60] M. Jones. Above And Below Guarantee Parameterizations For Combinatorial Optimisation Problems. PhD thesis, Royal Holloway, University of London, UK (2013). 84, 116

[61] S. Jukna. Extremal Combinatorics. Texts in Theoretical Computer Science, 2nd ed. (2011). 52

[62] R.M. Karp. Reducibility Among Combinatorial Problems. Complexity of Computer Computations. The IBM Research Symposia Series, pages 85-103 (1972). 7, 30, 71, 118

[63] S. Khot and R. O'Donnell. SDP gaps and UGC-hardness for Max-Cut-Gain. Theory of Computing 5, pages 83-117 (2009). 72

[64] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for Max Cut and other 2-variable CSPs?. SIAM Journal on Computing 37 (1), pages 319-357 (2007). 72

[65] S. Kratsch. Co-nondeterminism in compositions: a kernelization lower bound for a Ramsey-type problem. SODA 2012, pages 114-122 (2012). 23

[66] S. Kratsch, M. Pilipczuk, A. Rai, and V. Raman. Kernel lower bounds using co-nondeterminism: Finding induced hereditary subgraphs. Lecture Notes in Computer Science 7357, pages 364-375 (2012). 23

[67] M. Lampis. A kernel of order $2k - c\log k$ for vertex cover. Information Processing Letters 111 (23-24), pages 1089-1091 (2011). 120

[68] L. Levin. Universal search problems. Problems of Information Transmission 9 (3), pages 115-116 (1973). 117

[69] D. Lokshtanov. New Methods in Parameterized Algorithms and Complexity. PhD thesis, Universitetet i Bergen, Bergen, Norway (2009). 13

[70] M. Mahajan and V. Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. Electronic Colloquium on Computational Complexity 4 (36) (1997). 30, 72

[71] N. Misra, V. Raman, and S. Saurabh. Lower bounds on kernelization. Discrete Optimization 8, pages 110-118 (2011). 11

[72] M. Mnich, G. Philip, S. Saurabh, and O. Suchý. Beyond Max-Cut: λ-extendible properties parameterized above the PoljakTurzík bound. Journal of Computer and System Sciences 80 (7), pages 1384-1403 (2014). 76, 77, 116

[73] B.M.E. Moret and H.D. Shapiro. On minimizing a set of tests. SIAM Journal on Scientific and Statistical Computing 6, pages 983-1003 (1985). 30, 43, 48, 51, 120

[74] N.V. Ngoca and Zs. Tuza. Linear-time approximation algorithms for the max cut problem. Combinatorics, Probability and Computing 2 (2), pages 201-210 (1993). 72

[75] R. Niedermeier. Invitation to Fixed-Parameter Algorithms. Oxford University Press (2006). 9, 14, 17, 18, 120, 121

[76] R. Niedermeier. Ubiquitous Parameterization - Invitation to Fixed-Parameter Algorithms. Lecture Notes in Computer Science 3153, pages 84-103 (2004). 9

[77] W.L. Pearn. Solvable cases of the k-person Chinese postman problem. Operations Research Letters 16 (4), pages 241-244 (1994). 35

[78] S. Poljak and D. Turzìk. A polynomial time heuristic for certain subgraph optimization problems with guaranteed worst case bound. Discrete Mathematics 58 (1), pages 99-104 (1986). 30, 73, 74, 77, 97

[79] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. STOC 1997, pages 475-484 (1997). 118

[80] A. Schäfer, C. Komusiewicz, H. Moser, and R. Niedermeier. Parameterized Computational Complexity of Finding Small-Diameter Subgraphs. Optimization Letters 6 (5), pages 883-891 (2012). 13

[81] M. Sorge. Some algorithmic challenges in arc routing. Talk at NII Shonan Seminar No. 18 (2013). 30

[82] S. Thomassé, N. Trotignon, and K. Vušković. Parameterized algorithm for weighted independent set problem in bull-free graphs. CoRR, abs/1310.6205 (2013). 13

[83] C. Thomassen. On the complexity of finding a minimum cycle cover of a graph. SIAM Journal on Computing 26 (3), pages 675-677 (1997). 34

[84] R. van Bevern, R. Niedermeier, M. Sorge, M. Weller. Complexity of arc rooting problems. Arc Routing: Problems, Methods and Applications, Chapter 2, SIAM, Philadelphia, in press. 30

[85] C.K. Yap. Some consequences of non-uniform conditions on uniform classes. Theoretical Computer Science 26 (3), pages 287-300 (1983). 22

[86] T. Zaslavsky. Bibliography of signed and gain graphs. Electronic Journal of Combinatorics (1998). 103

[87] L. Zhang. Polynomial Algorithms for the k-Chinese Postman Problem. Information Processing 1992 1, pages 430-435 (1992). 35