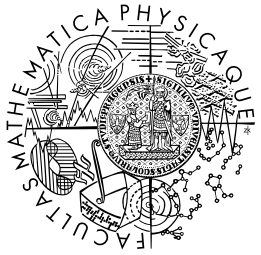

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Dušan Knop

Structural Properties of Graphs and Eficient Algorithms: Problems Between Parameters

Department of Applied Mathematics

Supervisor of the doctoral thesis:	Doc. RNDr. Jiří Fiala, Ph.D.
Study programme:	Computer Science
Study branch:	I4I – Discrete Models and Algorithms

Prague 2017

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature of the author

Title: Structural Properties of Graphs and Eficient Algorithms:
Problems Between Parameters

Author: Dušan Knop

Department: Department of Applied Mathematics

Supervisor: Doc. RNDr. Jiří Fiala, Ph.D., Department of Applied Mathematics

Abstract: Parameterized complexity became over last two decades one of the most important subfield of computational complexity. Structural graph parameters (widths) play important role both in graph theory and (parameterized) algoritmh design. By studying some concrete problems we exhibit the connection between structural graph parameters and parameterized tractability. We do this by examining tractability and hardness results for the Target Set Selection, Minimum Length Bounded Cut, and other problems.

In the Minimum Length Bounded Cut problem we are given a graph, source, sink, and a positive integer L and the task is to remove edges from the graph such that the distance between the source and the sink exceeds L in the resulting graph. We show that an optimal solution to the Minimum Length Bounded Cut problem can be computed in time $f(k)n$, where f is a computable function and k denotes the tree-depth of the input graph. On the other hand we prove that (under assumption that $\mathsf{FPT} \neq \mathsf{W}[1]$) no such algorithm can exist if the parameter k is the tree-width of the input graph. Currently only few such problems are known.

The Target Set Selection problem exibits the same phenomenon for the vertex cover number and neighborhood diversity. In its specialized variabnt (Majority Target Set Selection) for neighborhood diversity and twin-cover on one side and modular width on the other side. Currently we are not aware of other result of this type.

Keywords: graph decompositions, computational complexity, parameterized complexity

First of all my thanks goes to Doc. RNDr. Jiří Fiala, Ph.D. – you were and will be one of the most influencing teachers and friends in my life. I have to thank all my coauthors for all the moments we have spent together trying to resolve open questions in theoretical computer science. Especially, I am grateful for the time we spend on problems we have never successfully ansvered.

Last but not least I thank to my family, foremost to my wife, for their unstoppable continuous support and love.

Contents

Introduction

Besides the fact that efficient algorithms were found for large variety of problems still plenty of problems does not admit such algorithms. From what we know it may also be possible that for these hard problems there is no such efficient algorithm. On the other hand, in our day to day life we have to find solutions for these (NP-hard) problems. Scientist have developed two powerful ways which can deal with algorithmic hardness – namely *approximation algorithms* and *fixed-parameter algorithms*.

For the first group of algorithms an approximate solution is found in polynomial time. This means that the algorithm returns a solution for which one can prove that it is not too bad. Say, for a 3-approximation algorithm one proves that three times the value of optimal solution is an upper bound on the value of the solution returned by the algorithm (in the case of minimization problem). The best one can hope for is a so called *approximation scheme*. When a problem admits an approximation scheme there is a family of algorithms (for each guarantee) that runs in time polynomial in input size and the required approximation guarantee.

In the other group the task is to return an optimal solution in polynomial time, however the algorithm can run in any computable (exponential or even worse) time of the *parameter*. This leads to running time of a form $f(k)n^c$, where $f\colon \mathbb{N} \to \mathbb{N}$ is a computable function, c is constant, n is the length of input, and k is the parameter value for the input instance. A parameter can be a structural property of the input or a value of the optimal solution (in this case we usually say that this is a natural parameterization).

In both approaches there is also the need for hardness results. In the case of approximation algorithms this is the APX-hardness. If a problem is proven to be APX-hard this means that there is no approximation scheme for the problem, unless P=NP. In the case of parameterized algorithms we have the W[1]-hardness. If a problem is shown to be W[1]-hard, then (under some theoretic assumption) the best we can hope for is a so called XP algorithm, that is an algorithm that runs in polynomial time for every fixed value of the parameter. This means a running time of a form $n^{f(k)}$, where $f\colon \mathbb{N} \to \mathbb{N}$ is a computable function, n is the length of input, and k is the parameter value for the input instance.

We focus here solely on parameterization by the so called structural parameters. The most wide-spread and successful structural graph parameter is the tree-width. However, lot more structural graph parameters are known until today and we are sure that many are still to come. For a given problem we are interested in identifying the source of hardness of the instances. From the structural parameters point of view we are interested in which graph parameters allow the design of a fixed-parameter algorithm and which lead to W[1]-hardness. It is possible to find a relationship between structural parameters in a sense that e.g., the vertex cover number is an upper-bound for the tree-width of a graph. We give more detailed overview of the whole big picture of parameters in Section 1.3.

In this work we focus on fixed-parameter tractability of various graph problems and we find for some problems two close structural graph parameters for which the assumed problem has different behavior. We do this by finding an FPT algorithm for a given problem with respect to some structural parameter and

by designing a W[1]-hardness reduction for the problem with respect to a more general structural parameter.

Our Contribution

Recall that the most notable example of a structural graph parameter that become widely known among theorist and practitioners is tree-width. As this is the case the most notable of all the problems contained in this thesis is the Minimum Length Bounded Cut – for this problem we have shown that it admits a fixed parameterized algorithm for graphs of bounded tree-depth and remains hard on graphs with bounded tree-width. We establish these results in Chapter 2. Still only few problems have been discovered to have this property.

Beside its importance, tree-width and parameters related to it have small values only when the graph is sparse (i.e., the graph cannot contain large clique as a subgraph). We have focused mainly on dense graphs and thus we have to find suitable parameterizations for dense graph classes – fortunately there were many already in the literature. For these graphs the Target Set Selection problem (and its variants) seems to be an important problem. Here we have shown that the most general variant of the problem is fixed parameter tractable on graphs of bounded vertex cover (this was previously known) while the problem remains hard on graphs of bounded neighborhood diversity. The majority variant of the problem remains fixed parameter tractable on graphs of bounded neighborhood diversity while it remains hard on graphs of bounded modular width. We prove this formally in Chapter 3.

In chapters 4 and 5 we further study the neighborhood diversity as a structural graph parameter. We show that many problems previously known to admit an FPT algorithm with respect to vertex cover do admit such an FPT algorithm also on neighborhood diversity. Note that this does not hold for the Target Set Selection problem as is shown already in Chapter 3.

The work is based on content of the following articles

Chapter 2 Pavel Dvořák and Dušan Knop. Parametrized complexity of length-bounded cuts and multi-cuts. In R. Jain, S. Jain, and F. Stephan, editors, Theory and Applications of Models of Computation - 12th Annual Conference, TAMC 2015, Singapore, May 18-20, 2015, Proceedings, volume 9076 of Lecture Notes in Computer Science, pages 441–452. Springer, 2015. [26]

Chapter 3 Pavel Dvořák, Dušan Knop, and Tomáš Toufar. Target set selection in dense graph classes. CoRR, abs/1610.07530, 2016. [27]

Chapter 4 Jiří Fiala, Tomáš Gavenčiak, Dušan Knop, Martin Koutecký, and Jan Kratochvíl. Fixed parameter complexity of distance constrained labeling and uniform channel assignment problems - (extended abstract). In T. N. Dinh and M. T. Thai, editors, Computing and Combinatorics - 22nd International Conference, COCOON 2016, Ho Chi Minh City, Vietnam, August 2-4, 2016, Proceedings, volume 9797 of Lecture Notes in Computer Science, pages 67–78. Springer, 2016. [29]

Chapter 5 Dušan Knop. Partitioning graphs into induced subgraphs. In F. Drewes, C. Martín-Vide, and B. Truthe, editors, Language and Automata

Theory and Applications - 11th International Conference, LATA 2017, Umeå, Sweden, March 6-9, 2017, Proceedings, volume 10168 of Lecture Notes in Computer Science, pages 338–350, 2017. [54]

1. Parameterized Complexity and Graph Parameters

In this chapter we give an overview of basic theory of parameterized complexity. We define three most important parameterized complexity classes (FPT, XP, and W[1]) and review a framework for refuting polynomial sized kernels. The rest of the chapter is devoted to definitions of various structural graph parameters.

1.1 Parameterized Complexity

In what follows Σ is always a finite alphabet. For a positive integer n we denote the set of all strings of length n by Σ^n (i.e., the set of all n-tuples of elements of Σ). For $n = 0$ the set Σ^0 consists of the empty word λ only. Further, by Σ^* we denote the set of all words, i.e., all string composed of letters in Σ, formally $\Sigma^* = \bigcup_{n\in\mathbb{N}} \Sigma^n$. By a *(unparameterized) problem* we mean a set $L \subseteq \Sigma^*$. A *parameterized problem* is a set $P \subseteq \Sigma^* \times \mathbb{N}$. Here, the natural number associated with a word $x \in \Sigma^*$, also denoted $\kappa(x)$, is a *parameter* of the problem. Usually, we say that the function $\kappa\colon \sigma^* \to \mathbb{N}$ is a parameterization of the problem. Note that one decision (i.e., unparameterized) problem can have many various parameterizations. We say that (x, k) is a *yes*-instance of problem A if $(x, k) \in A$ and it is a *no*-instance otherwise.

As in classical computational complexity we describe the parameterized reduction here and follow it by definitions of some important classes of problems.

Definition 1 (Parameterized Reduction)**.** *Let $A, B \in \Sigma \times \mathbb{N}$ be two parameterized problems.* A parameterized reduction *from A to B is an algorithm that, given an instance (x, k) of A, outputs an instance (y, ℓ) of B with*

- *(x, k) is a* yes*-instance of A if and only if (y, ℓ) is a* yes*-instance of B,*
- *$\ell \leq g(k)$ for some computable function $g\colon \mathbb{N} \to \mathbb{N}$, and*
- *the running time of the algorithm is $f(k)\,\mathrm{poly}(|x|)$ for some computable function $f\colon \mathbb{N} \to \mathbb{N}$.*

We say that an algorithm *solves* parameterized problem P if on input (x, k) it always stops and it decides whether $(x, k) \in P$ or not. That is the algorithm outputs YES if $(x, k) \in P$ and NO otherwise. The class of tractable problems is a generalization of the well-known class P. All the classes presented here are closed with respect to parameterized reduction.

Definition 2 (FPT)**.** *A parameterized problem $P \subseteq \Sigma^* \times \mathbb{N}$ belongs to class* FPT *if there exists an algorithm that decides P and on input (x, k) runs in time $\mathcal{O}\big(f(k)\,\mathrm{poly}(|x|)\big)$. Here $f\colon \mathbb{N} \to \mathbb{N}$ is some computable function independent of the size of input $|x|$.*

We abuse the notation a bit and say that there is an FPT-algorithm for a problem if the problem belongs to the class FPT. It is well-know that there is an FPT-algorithm for a decidable problem if and only if there is a kernelization

algorithm. Roughly speaking, kernelization is a data reduction technique and it consist of the so caller reduction rules. The idea behind this technique is to reduce the input instance to an equivalent instance (of the same problem) so that it is possible to bound the size of the resulting instance in terms of parameter only. We say that instances $(x, k), (x', k')$ are equivalent (for a parametrized problem P) if it holds that $(x, k) \in P$ if and only if $(x', k') \in P$. We can formalize this notion as follows. A kernelization algorithm on instance (x, k) of a parametrized problem runs in polynomial time in $|x|$ and returns an equivalent instance (x', k') with $k' \leq f(k)$ and $|x'| \leq f(k)$ for some computable function f (independent of $|x|$). We say that a parameterized problem P admits a polynomial kernel if there exist a kernelization algorithm for P with f being polynomial function. In this work we do not design new kernelization algorithms however, for some problems we prove that a polynomial kernels do not exist, unless something bad happens. We formalize this in Section 1.2.

In a strict contrast to the class FPT the class XP mimics the class E of problems tractable in exponential time.

Definition 3 (XP)**.** *A parameterized problem $P \subseteq \Sigma^* \times \mathbb{N}$ belongs to class* XP *if there exists an algorithm that decides P and on input (x, k) runs in time $\mathcal{O}\Big(|x|^{f(k)}\Big)$. Here again $f \colon \mathbb{N} \to \mathbb{N}$ is some computable function independent of the size of input $|x|$.*

Problems as hard as Clique. Despite the effort of many researchers among the World, no FPT algorithm has been shown for the k-Multicolor Clique problem. It is common then to utilize such a problem to derive intractability results. The problem is as follows.

k-Multicolored Clique	
Input:	k-partite graph $G = (V_1 \dot\cup V_2 \dot\cup \ldots \dot\cup V_k, E)$, where V_i is independent set for every $i \in [k]$ and they are pairwise disjoint
Parameter:	k
Task:	find a clique of the size k

As the formal definition of the class W[1] is rather technical we omit it here. Here we just assume that the k-Multicolor Clique problem is W[1]-hard and work under theoretic assumption that W[1] $\neq$ FPT. This assumption is stronger than P $\neq$ NP. We use the fact that a parameterized problem P is W[1]-hard as an evidence that no FPT algorithm for P can exist.

1.2 Refuting Polynomial Kernels

Here we present a simplified review of a framework used to refute existence of polynomial kernel for a parameterized problem from Chapter 15 of a monograph by Cygan et al. [21].

In the following we denote by Σ a finite alphabet, by Σ^* we denote the set of all words over Σ and by $\Sigma^{\leq n}$ we denote the set of all words over Σ and length at most n.

Definition 4 (Polynomial equivalence relation)**.** *An equivalence relation $\mathcal{R}$ on the set Σ^* is called* polynomial equivalence relation *if the following conditions are satisfied:*

1. *There exists an algorithm such that, given strings $x, y \in \Sigma^*$, resolves whether $x \equiv_{\mathcal{R}} y$ in time polynomial in $|x| + |y|$.*

2. *Relation $\mathcal{R}$ restricted to the set $\Sigma^{\leq n}$ has at most $p(n)$ equivalence classes for some polynomial $p(\cdot)$.*

Definition 5 (AND-cross-composition)**.** *Let $L \subseteq \Sigma^*$ be an unparameterized language and $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized language. We say that L* cross-composes *into Q if there exists a polynomial equivalence relation $\mathcal{R}$ and an algorithm $\mathcal{A}$, called the AND-cross-composition, satisfying the following conditions. The algorithm $\mathcal{A}$ takes on input a sequence of strings $x_1, x_2, \ldots, x_r \in \Sigma^*$ that are equivalent with respect to $\mathcal{R}$, runs in polynomial time in $\sum_{i=1}^{r} |x_i|$, and outputs one instance $(y, k) \in \Sigma^* \times \mathbb{N}$ such that:*

1. *$k \leq p(\max_{i=1}^{r} |x_i|, \log r)$ for some polynomial $p(\cdot, \cdot)$, and*

2. *$(y, k) \in Q$ if and only if $x_i \in L$ for all i.*

We say that language L has a *polynomial kernel* if there is a kernelization algorithm that takes on input an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, runs in polynomial time in $|x|$ and k, and outputs an equivalent instance $(x', k') \in \Sigma^* \times \mathbb{N}$ with $|x'| \leq p(k')$ and $k' \leq q(k)$, where $p(\cdot), q(\cdot)$ are polynomials. With this framework, it is possible to refute even stronger data reduction techniques – namely polynomial compression:

Definition 6 (Polynomial compression)**.** *A* polynomial compression *of a parameterized language $Q \subseteq \Sigma^* \times \mathbb{N}$ into an unparameterized language $R \subseteq \Sigma^*$ is an algorithm that takes as input an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, works in polynomial time in $|x| + k$, and returns a string y such that:*

1. *$|y| \leq p(k)$ for some polynomial $p(\cdot)$, and*

2. *$y \in R$ if and only if $(x, k) \in Q$.*

It is easy to see that the polynomial kernelization is a special case of the polynomial compression. It is possible to refute existence of polynomial compression (and polynomial kernel) using AND-cross-composition with the help of use of the following theorem and a complexity assumption that is unlikely to hold – namely $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

Theorem 1 ([21, 5])**.** *Assume that an NP-hard language L AND-cross-composes to a parameterized language Q. Then Q does not admit a polynomial compression, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

1.3 Structural Graph Parameters

We give a formal definition of several graph parameters used in this work. For a better acquaint with these parameters, we provide a map of assumed parameters in Figure 1.1.

Significant amount of research work has been devoted to find efficient algorithms for graphs that admit embedding into a fixed surface or e.g., graphs with bounded tree-width. Both these classes of graphs together with many other classes are examples of the nowhere dense graphs, a generalization introduced by Nešetřil and Ossona de Mezdez [62]. A graph H' is an *r-subdivision* of a graph H if H' can be obtained from H by replacing edges by vertex disjoint paths of length at most $r+1$. Graph H is a *topological depth-r minor* of a graph G, denoted $H \preceq_r^t G$, if an r-subdivision of H is isomorphic to a subgraph of G.

Definition 7. *Let $\mathcal{G}$ be a class of graphs. $\mathcal{G}$ is* nowhere dense *if*

$$\limsup_{n\to\infty} \left\{ \frac{\log(|E(H)|)}{\log(|V(H)|)} \colon G \in \mathcal{G} \text{ with } |V(G)| \geq n, H \preceq_r^t G \right\} \leq 1.$$

Otherwise $\mathcal{G}$ is somewhere dense.

Here a parameter $\kappa \colon \mathcal{G} \to \mathbb{N}$ is suitable for nowhere dense graphs if all classes $\mathcal{G}_t := \{G \colon G \text{ is a graph}, \kappa(G) \leq t\}$ are nowhere dense. For example such a graph parameter can bound the clique number of the graph by some function. This is the case e.g. for the vertex cover number – any graph with vertex cover of size k cannot contain a clique of size $k+2$ or larger. On the other hand if this is not the case, usually the class of all cliques has the parameter value bounded by some constant – this is the case for e.g. the neighborhood diversity.

1.3.1 Nowhere Dense Graph Classes and Structural Parameters

One of the most restrictive graph parameters is called the *vertex cover number* and is defined as follows.

Definition 8 (Vertex cover)**.** *For a graph $G = (V, E)$ the set $U \subset V$ is called a* vertex cover *of G if for every edge $e \in E$ it holds that $e \cap U \neq \emptyset$. The* vertex cover number *of a graph, denoted as* $\mathrm{vc}(G)$*, is the least integer k for which there exists a vertex cover of size k.*

We say that the vertex cover number is very restrictive graph parameter, because for a fixed positive integer k the class of graphs with vertex cover number bounded by k does not contain large spectra of graphs.

As the vertex cover number is (usually) too restrictive, many authors focused on defining other structural parameters. Four most well-known parameters of this kind are the tree-depth, the path-width, the tree-width (introduced by Robertson and Seymour [69]), and the clique-width (introduced by Courcelle et al. [19]). We start with the definition of tree-depth:

Definition 9 (Tree-depth [63])**.** *The* closure *$Clos(F)$ of a forest F is the graph obtained from F by making every vertex adjacent to all of its ancestors. The* tree-depth $\mathrm{td}(G)$ *of a graph G is one more than the minimum height of a rooted forest F such that $G \subseteq Clos(F)$.*

In order to define tree-width and path-width we first introduce the notion of tree decomposition:

Definition 10. *A* tree decomposition *of a graph $G = (V, E)$ consists of bags and a rooted tree. Formally, it is a pair $\mathcal{T} = (\{B_X : X \in I\}, T = (I, F))$, where T is a rooted tree and $\{B_X : X \in I\}$ is a family of subsets of V, such that*

1. *for each $v \in V$ there exists an $X \in I$ such that $v \in B_X$,*
2. *for each $e \in E$ there exists an $X \in I$ such that $e \subseteq B_X$,*
3. *for each $v \in V, I_v = \{X \in I : v \in B_X\}$ induces a subtree of T.*

We call the elements of I the nodes, the elements of the set F the decomposition edges *and a set B_X is a bag of node X.*

We define a width of a tree decomposition $\mathcal{T} = (\{B_X : X \in I\}, T)$ as the value $\max_{X \in I} |B_X| - 1$ and the *tree-width* $\mathrm{tw}(G)$ of a graph G as the minimum width of a tree decomposition of the graph G. Moreover, if the tree T in the tree decomposition is a path we speak about the *path-width* of G, which we denote as $\mathrm{pw}(G)$.

Nice Tree Decomposition [53] For algorithmic purposes it is common to define a *nice tree decomposition* of the graph. We rooted the decomposition and naturally orient the decomposition edges towards the root. For an oriented decomposition edge (X, Y) from X to Y we call Y the *parent* of X and X a *child* of Y. If there is an oriented path from X to Y we say that X is a *descendant* of Y.

We also adjust a tree decomposition such that for each decomposition edge (X, Y) it holds that B_X and B_Y differ in at most one vertex. The in-degree of each node is at most 2 and if the in-degree of the node Z is 2 then for its children X, Y holds that $B_X = B_Y = B_Z$ (i.e., they represent the same vertex set).

We classify the nodes of a nice decomposition into four classes – namely *introduce nodes*, *forget nodes*, *join nodes* and *leaf nodes*. We call the node X an introduce node of the vertex v if it has a single child Y and $B_X \setminus B_Y = \{v\}$. We call the node X a forget node of the vertex v if it has a single child Y and $B_Y \setminus B_X = \{v\}$. If the node Z has two children X and Y, we call it a join node (of the nodes X and Y). Finally we call a node X a leaf node if it has no child.

Proposition 2 ([53])**.** *Given a tree decomposition of a graph G with n vertices that has width k and $O(n)$ nodes, we can find a nice tree decomposition of G that also has width k and $O(n)$ nodes in time $O(n)$.*

1.3.2 Dense Graph Parameters

There are (more recent) structural graph parameters which also generalize the vertex cover number but in contrary to the tree-width these parameters focus on dense graphs. First, up to our knowledge, of these parameters is the *neighborhood diversity* defined by Lampis [57]. We denote the neighborhood diversity of a graph $G = (V, E)$ as $\mathrm{nd}(G)$.

Neighborhood diversity. We say that two distinct vertices u, v are of the same *neighborhood type* if they share their respective neighborhoods, that is when $N(u) \setminus \{v\} = N(v) \setminus \{u\}$.

Definition 11 (Neighborhood diversity [57])**.** *A graph $G = (V, E)$ has* neighborhood diversity *at most w ($\mathrm{nd}(G) \leq w$) if there exists a partition of V into at most w sets (we call these sets* types*) such that all vertices in a type have the same neighborhood type.*

Note that every type induces either a clique or an independent set in G and two types are either joined by a complete bipartite graph or no edge between vertices of the two types is present in G. Thus, we use the notion of a *type graph* – that is a graph T_G representing the graph G and its neighborhood diversity decomposition in the following way. The vertices of type graph T_G are the neighborhood types of the graph G and two such vertices are joined by an edge if all the vertices of corresponding types are joined by an edge. We would like to point out that it is possible to compute the neighborhood diversity of a graph in linear time [57].

Twin cover. More recently, Ganian [39] defined the *twin cover number*. We begin with an auxiliary definition. If two vertices u, v have the same neighborhood type and $e = \{u, v\}$ is an edge of the graph, we say that e is a *twin edge*.

Definition 12 (Twin cover number [39])**.** *A set of vertices $T \subseteq V$ is a* twin cover *of a graph $G = (V, E)$, if for every edge $e \in E$ either*

1. *$T \cap e \neq \emptyset$, or*
2. *e is a twin edge.*

We say that G has twin cover number *k ($\mathrm{tc}(G) = k$) if the size of a minimum twin cover of G is k.*

Note that after removing T from a graph G the resulting graph consists of disjoint union of cliques. We denote these cliques as *twin cliques*.

Note that the twin cover can be upper-bounded by the vertex cover number. As the structure of graphs with bounded twin cover is very similar to the structure of graphs with bounded vertex cover number there is a hope that many of known algorithms for graphs with bounded vertex cover number can be easily turned into algorithms for graphs with bounded twin cover number.

Cluster vertex deletion. A *cluster vertex deletion number* of a graph G ($\mathrm{cvdn}(G)$) is the minimum number of vertices to delete from G to transform it into a union of disjoint cliques [47].

Observe that cluster vertex deletion is a generalization of the twin cover number in a sense that if we remove the covering vertices of the twin cover from a graph the result is a union of disjoint cliques, however the converse is not true. To see this take a graph G formed by a clique on n vertices and one extra vertex attached to $n/2$ clique vertices – this graph has $\mathrm{cvdn}(G) = 1$ but $\mathrm{tc}(G) = n/2$.

Modular-width. Both neighborhood diversity and twin cover number are generalized by a modular-width, defined by Gajarský et al. [38]. Here we deal with graphs created by an algebraic expression that uses four following operations:

1. Create an isolated vertex.
2. The *disjoint union* of two graphs, that is from graphs $G = (V, E), H = (W, F)$ create a graph $(V \cup W, E \cup F)$.
3. The *complete join* of two graphs, that is from graphs $G = (V, E), H = (W, F)$ create a graph with vertex set $V \cup W$ and edge set

   $$E \cup F \cup \{\{v, w\} \colon v \in V, w \in W\}.$$

   Note that the edge set of the resulting graph can be also written as $E \cup F \cup (V \times W)$.
4. The *substitution operation* with respect to a template graph T with vertex set $\{v_1, v_2, \ldots, v_k\}$ and graphs $G_1, G_2, \ldots, G_k$ created by algebraic expression. The substitution operation, denoted by $T(G_1, G_2, \ldots, G_k)$, results in the graph on vertex set $V = V_1 \cup V_2 \cup \cdots \cup V_k$ and edge set

   $$E = E_1 \cup E_2 \cup \cdots \cup E_k \cup \bigcup_{\{v_i, v_j\} \in E(T)} \Big\{\{u, v\} \colon u \in V_i, v \in V_j\Big\},$$

   where $G_i = (V_i, E_i)$ for all $i = 1, 2, \ldots, k$.

Definition 13 (Modular-width [38])**.** *Let A be an algebraic expression that uses only operations* 1–4*. The* width of expression A *is the maximum number of operands used by any occurrence of operation* 4 *in A. The* modular-width *of a graph G, denoted as* $\mathrm{mw}(G)$*, is the least positive integer k such that G can be obtained from such an algebraic expression of width at most k.*

When a graph H is constructed by the fourth operation, that is

$$G = T(G_1, G_2, \ldots, G_k),$$

we call the graph T the *template graph*. An algebraic expression of width $\mathrm{mw}(G)$ can be computed in linear time [70].

Restricted Modular-width. We would like to introduce here a restriction of the modular width that still generalizes both neighborhood diversity and twin cover number. The algebraic expression used to define graph G can contain the substitution operation at most once and if it contains the substitution operation it has to be the last operation in the expression. However, there is no limitation for the use of operations 1–3.

It is easy to see that this generalizes neighborhood diversity as it is possible to build a complete graph or independent set of arbitrary size using operations 1 and 3 only. A graph G with $\mathrm{nd}(G) \leq k$ can be constructed from a type graph T_G by replacing each vertex in $V(T_G)$ by an independent set or a clique. A graph G with $\mathrm{rmw}(G) \leq k$ can be constructed from a type graph T_G by replacing each vertex in $V(T_G)$ by arbitrary cograph – a graph H is a cograph if H can be constructed

by operations 1–3. Since cliques and independent sets are cographs, restricted modular-width is generalization of neighborhood diversity.

It is not hard to argue that this parameter generalizes twin cover number as well To see this divide the twin cliques according to their neighborhood in the set T. Now observe that it is possible to build disjoint union of cliques using operation 2.

One may ask, whether requiring the template operation to be used just once leads to the same class of graph as if we require it to be the last operation. However, this is not the case as is shown in the following lemma. Thus, we obtain sort of hierarchy leading to modular-width.

Lemma 3. *Requiring the substitution operation to be used as the last operation is more restrictive than requiring it to be used just once.*

Proof. To show this we build a family of graphs G_n on $n = 4, 5, \ldots$ vertices which admits a decomposition in which only one substitution operation of width 4 is used. We begin by setting $G_4 = P_4$. Note that P_4 is not a cograph and thus it has to be constructed using the substitution operation. We define for $n > 2$ graph G_{2n+1} as the graph G_{2n} plus an apex vertex and G_{2n+2} as the graph G_{2n+1} with an isolated vertex added.

Now observe that if we require the substitution operation to be the last operation of the decomposition of G_n, then its width is n. While if we allow to build P_4 using substitution operation of size 4, then what remains can be build only using the operations 1–3. □

The last parameter we have to define is the most general one:

Definition 14 (Clique-width [18])**.** *The* clique-width *of a graph G (denoted* $\mathrm{cw}(G)$*) is the minimum number of labels needed to construct G by the following operations:*

1. *creation of a new vertex v with label ℓ,*
2. *disjoint union of two labeled graphs H_1 and H_2,*
3. *joining by an edge every vertex labeled ℓ_1 to every vertex labeled ℓ_2, where $\ell_1 \neq \ell_2$, and*
4. *renaming label ℓ_1 to label ℓ_2.*

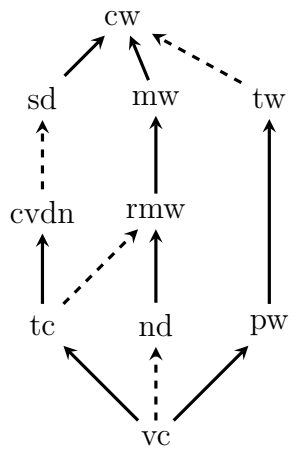

Figure 1.1: A map of assumed parameters. A full arrow stands for a linear upper bound, while a dashed arrow stands for an exponential upper bound. For example if a graph G has $\mathrm{vc}(G) \leq k$ then $\mathrm{nd}(G) \leq 2^k + k$.

2. Length Bounded Cuts in Sparse Graphs

2.1 Introduction

The study of network flows and cuts began in 1950s by the work of Ford and Fulkerson [36]. It has many generalizations and applications now. We are interested in a generalization of cuts related to the flows using only short paths.

Length-bounded Cuts Let $s, t \in V$ be two distinct vertices of a graph $G = (V, E)$ – we call them the source and the sink, respectively. We call a subset of edges $F \subseteq E$ of G an *L-bounded cut* (or *L-cut* for short), if the length of the shortest path between s and t in the graph $(V, E \setminus F)$ is at least $L + 1$. We measure the length of the path by the number of its edges. In particular, we do not require s and t to be in distinct connected components as in the standard cut, instead we do not allow s and t to be close to each other. We call the set *F a minimum L-cut* if it has the minimum size among all L-bounded cuts of the graph G. Throughout the paper we denote by n the number of vertices of input graph G and by m the number of edges of G.

We state the cut problem formally:

Minimum Length Bounded Cut (MLBC)	
Input:	graph $G = (V, E)$, vertices s, t and integer $L \in \mathbb{N}$
Task:	find a minimum L-bounded s, t cut $F \subset E$

Length-bounded flows were first considered by Adámek and Koubek [2]. They showed that the max-flow min-cut duality cannot hold and also that integral capacities do not imply integral flow. Finding a minimum length-bounded cut is NP-hard on general graphs for $L \geq 4$ as was shown by Itai et al. [49]. They also found algorithms for finding a minimum L-bounded cut with $L = 1, 2, 3$ in polynomial time by reducing it to the usual network cut in an altered graph. The algorithm of Itai et al. [49] uses the fact that paths of length $1, 2$ and 3 are edge disjoint from longer paths, while this does not hold for length at least 4.

Baier et al. [3] studied linear programming relaxation and approximation of MLBC together with inapproximability results for MLBC. They also showed instances of the MLBC having $O(L)$ integrality gap for their linear programming approach, which are series-parallel graphs and thus have constant bounded tree-width. The first parameterized complexity study of this and similar topics was made by Golovach and Thilikos [41] who studied parameterization by paths-length (that is in our setting the parameter L) and the size of the solution for cuts. They also proved hardness results – finding disjoint paths in graphs of bounded tree-width is a W[1]-hard problem. Very recently Fluschnik et al. [35] showed that, unless a collapse in the Polynomial Hierarchy occurs, there is no polynomial kernel with respect to parameters L and the size of the solution.

The MLBC problem has its applications in network design and in telecommunications. Huygens et al. [48] use a MLBC as a subroutine in the design of

2-edge-connected networks with cycles at most L long. The MLBC problem is called *hop constrained* in telecommunications and the number L is so called number of hops. The main interest is in the constant number of hops, see for example the article of Dahl and Gouveia [22].

Note that the standard use of Courcelle's theorem [16] gives for each fixed L a linear time algorithm for the decision version of the problem. But there is no apparent way of changing these algorithms into a single linear time algorithm. Moreover there is a nontrivial dependency between the formula (and thus the parameter L) and the running time of the algorithm given by Courcelle's theorem.

Now we give a formal definition of a rather new graph parameter, for which we give one of our results:

Definition 15 (Tree-depth [63])**.** *The closure $Clos(F)$ of a forest F is the graph obtained from F by making every vertex adjacent to all of its ancestors. The tree-depth* $\mathrm{td}(G)$ *of a graph G is one more than the minimum height of a rooted forest F such that $G \subseteq Clos(F)$.*

Our Contribution Our main contribution is an algorithm for the MLBC problem, its consequences and an algorithm for a more general multi-terminal version problem.

Theorem 4. *Let G be a graph of tree-width k. Let s and t be two distinct vertices of G. Then for any $L \in \mathbb{N}$ a minimum L-cut between s and t can be found in time $O\Big(L^{6k^2} \cdot n\Big)$.*

Corollary. Let G be a graph, $k = \mathrm{td}(G)$ and s and t be two distinct vertices of G. Then for any $L \in \mathbb{N}$ a minimum L-cut between s and t can be found in time $O\Big(\max\{2^{6k^3} \cdot n, nm\}\Big)$.

Proof. As k is the tree-depth of G it follows that the length of any path in G can be upper-bounded by 2^k (this follows from Proposition 6.2 in Nešetřil, de Mendez book [62]). It is a folklore fact, that k is also an upper-bound on the tree-width of G. Thus, we can use Theorem 4 for $L < 2^k$. If $L \geq 2^k$, then L-cut is standard minimum cut in G. For this case we use Orlin's algorithm [65] for max flow/min cut problem with running time $O(nm)$. □ □

Corollary. Let $G = (V, E)$ be a graph of tree-width k, $s \neq t \in V$ and $L \in \mathbb{N}$. A minimum L-cut between s and t can be found in time $O\Big(n^{6k^2+1}\Big)$.

Theorem 5. Minimum Length-Bounded Cut *parameterized by path-width is* W[1]-hard.

Path-width versus Tree-depth Admitting an FPT algorithm for a problem when parameterized by the path-width (tree-width) implies an FPT algorithm for the problem when parameterized by the tree-depth, as parameter-theoretic observation easily shows. On the other hand, the FPT algorithm parameterized by the path-width (tree-width) usually uses exponential (in the width) space, while the tree-depth version uses only polynomial space (in the tree-depth).

From this point of view, it is interesting to find problems that are "on the edge between path-width and tree-depth". That is problems that admit an FPT

algorithm when parameterized by the tree-depth, but being W[1]-hard when parameterized by the path-width.

The only other result of this type we are aware of, in the time of writing this article, is by Gutin et al. [44]. The Minimum Length-Bounded Cut problem is also a problem of this kind – as Theorem 5 and Corollary 2.1 demonstrate.

Theorem 4 gives us that the MLBC problem is fixed parameter tractable (FPT) when parameterized by the length of paths and the tree-width and that it belongs to XP when parameterized by the tree-width only (and is thus solvable in polynomial time for graph classes with constant bounded tree-width).

Theorem 6. *There is no polynomial kernel for the* Minimum Length Bounded Cut *problem parameterized by the tree-width of the graph and the length L, unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

We want to mention that our techniques apply also for a more general version of the MLBC problem.

Length-bounded Multicut We consider a generalized problem, where instead of only two terminals, we are given a set of terminals. For every pair of terminals, we are given a constraint – a lower bound on the length of the shortest path between these terminals.

More formally, let $S = \{s_1, \ldots, s_q\} \subseteq V$ be a subset of vertices of the graph $G = (V, E)$ and let $\mathbf{a} \in \mathbb{N}^{\binom{S}{2}}$ be a vector, where $\mathbb{N}^{\binom{S}{2}}$ is a set of all natural number vectors indexed by pairs of vertices in S. We call a subset of edges $F \subseteq E$ of G an $\mathbf{a}$-*bounded S-multicut* if the length of the shortest path between s_i and s_j in the graph $(V, E \setminus F)$ is at least $a_{s_i,s_j} + 1$ for every $s_i, s_j \in S, i < j$. Again if F has smallest possible size, we call it *minimum* $\mathbf{a}$-*bounded S-multicut*. We call the vertices $s_1, \ldots, s_q$ *terminals*. Let $L \geq \max_{s_i,s_j \in S:\, i<j} a_{s_i,s_j}$, we say that the problem is *L-limited*.

Minimum Length-Bounded MultiCut (MLBMC)	
Input:	graph $G = (V, E)$, set $S \subset V$ and $\mathbf{a} \in \mathbb{N}^{\binom{S}{2}}$
Task:	find a minimum $\mathbf{a}$-bounded S-multicut $F \subset E$

Theorem 7. *Let $G = (V, E)$ be a graph of tree-width k, $S \subseteq V$ with $|S| = q$ and let $p := q + k$. Then, for any $L \in \mathbb{N}$ and any L-limited length-vector $\mathbf{a}$ on S a minimum $\mathbf{a}$-bounded S-multicut can be computed in time $O\big(L^{4p^2} \cdot n\big)$.*

2.2 Preliminaries

In this section we introduce some minor changes of the tree decomposition specific for our algorithm. We proceed by the notion of auxiliary graphs used in proofs of our algorithm correctness.

Grafted Tree Decomposition So far we have described a standard nice tree decomposition. Now, we will describe how to change a nice tree decomposition to a *grafted tree decomposition*. Let X be an introduce node and Y its child. We add another two nodes X^c and X^s such that $B_{X^c} = B_{X^s} = B_X$. We remove

decomposition edge (Y, X) and add three decomposition edges $(Y, X^c), (X^s, X)$ and (X^c, X) (see Figure 2.1). Note that after this operation, X^s is a leaf of the decomposition, X^c is an introduce node and X is a join node. Note that by these further modifications we preserve linear number of nodes in the decomposition.

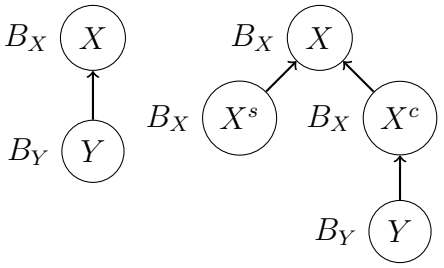

Figure 2.1: Change of Introduction nodes in the grafted tree decomposition.

Note that in the grafted tree decomposition for each edge e there is at least one leaf X^s of the decomposition satisfying $e \subseteq B_{X^s}$. By the definition of tree decomposition, we know there is a node X^c such that $e \subseteq B_{X^c}$. If X^c is not a leaf node, then we may suppose that X^c is an introduce node (for join or forget node choose its descendant). However, in the grafted decomposition any introduce node X^c has a sibling X^s that is a leaf node and $B_{X^c} = B_{X^s}$.

Auxiliary Subgraphs For every edge $e \in E(G)$ we choose an arbitrary leaf node X such that $e \in B_X$ and say that the edge e *belongs* to the leaf node X. By this process we have chosen set $E_X \subset E(G)$ for each leaf node X. Note that the sets E_X for all leaves X of the decomposition forms a partition of the set $E(G)$. We further use the notion of *auxiliary graph* G_X. For a leaf node X we set a graph $G_X = (B_X, E_X)$. For a non-leaf node Y we set a graph $G_Y = (V, E)$, where

$$V = B_Y \cup \bigcup_{X \text{ child of } Y} V(G_X)$$
$$E = \bigcup_{X \text{ child of } Y} E(G_X).$$

Vector Notation We often use integer vectors whose entries are indexed by pairs of vertices. We use bold characters for vectors $(\mathbf{a}, \mathbf{b})$ and italic characters for entries ($a_{x,y}, b_{x,y}$ are entries of $\mathbf{a}$, $\mathbf{b}$ respectively, for a pair of vertices x, y). Let S be a set of vertices and $\mathbf{a}, \mathbf{b} \in \mathbb{N}^{\binom{S}{2}}$. We write $\mathbf{a} \preceq \mathbf{b}$, if $a_{x,y} \leq b_{x,y}$ for all $\{x, y\} \in \binom{S}{2}$.

2.3 FPT Algorithm for the L-bounded Cut

In this section we present our approach to the L-bounded cut for the graphs of bounded tree-width. First, we give a more detailed study of the length constraints for the length-bounded multicut and the triangle inequalities. From this we derive Lemma 9 for merging solutions for edge-disjoint graphs. Second, we describe how to use dynamic programming in different nodes of the tree decomposition, which gives us the final algorithm.

Triangle Inequalities Let $\mathcal{I} = \big(G = (V, E), S, \mathbf{a}\big)$ be an instance of MLBMC and $F \subseteq E(G)$ be a solution of $\mathcal{I}$. Note that the distances between terminals in $G' = (V, E \setminus F)$ satisfy the triangle inequalities. This means that for any three terminals $s, t, u \in S$ and the distance function $\mathrm{dist}\colon V \times V \to \mathbb{N}$ in G' it holds that $\mathrm{dist}(s, u) + \mathrm{dist}(u, t) \geq \mathrm{dist}(s, t) \geq a_{s,t} + 1$. We say that a vector $\mathbf{a} \in \mathbb{N}^{\binom{S}{2}}$ *satisfies sharp triangle inequalities* if $a_{s,u} + a_{u,t} + 1 \geq a_{s,t}$ (i.e. $a_{s,u} + a_{u,t} > a_{s,t}$) for all distinct terminals $s, t, u \in S$. Thus, it makes sense to restrict instances of MLBMC problem only to those satisfying sharp triangle inequalities. We will formalize this idea in Observation 8.

Definition 16 (Length constraints)**.** *Let $G = (V, E)$ be a graph, $S \subset V$ and let $k = |S|$. We call a vector $\mathbf{a} = (a_{s_1,s_2}, \ldots, a_{s_{k-1},s_k})$ a* length constraint *if it satisfies sharp triangle inequalities.*

Observation 8. *Let $\mathcal{I} = \big(G = (V, E), S, \mathbf{a}\big)$ be an instance of MLBMC and $F \subseteq E$ be a solution of $\mathcal{I}$. Now, there exists a length constraint $\mathbf{b} \in \mathbb{N}^{\binom{S}{2}}$ such that $\mathbf{b} \succeq \mathbf{a}$ and F is a solution of $(G, S, \mathbf{b})$.*

Proof. Let $\mathrm{dist}\colon V \times V \to \mathbb{N}$ be a distance function in graph $(V, E \setminus F)$. We define $b_{s,t}$ as $\mathrm{dist}(s, t) - 1$ for all distinct terminals $s, t \in S$; thus F is $\mathbf{b}$-bounded S-multicut. It is straightforward to check the vector $\mathbf{b}$ is a length constraint because the function dist satisfies triangle inequality. Now, $\mathbf{b} \succeq \mathbf{a}$, since $a_{s,t} < \mathrm{dist}(s, t)$ for all distinct terminals $s, t \in S$. Since $\mathbf{b} \succeq \mathbf{a}$, the instance $(G, S, \mathbf{b})$ does not admit a solution of size smaller than $|F|$. Therefore, the set F is a solution of $(G, S, \mathbf{b})$. □

For our approach it is important to see the structure of the solution on a graph composed from two edge disjoint graphs.

Lemma 9. *Let $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ be edge disjoint graphs. Then for the graph $G = G_1 \cup G_2$ and $S = V_1 \cap V_2$ and an arbitrary length constraint $\mathbf{a} \in \mathbb{N}^{\binom{S}{2}}$ it holds that a minimum length $\mathbf{a}$-bounded S-multicut F for G is a disjoint union of the minimum length $\mathbf{a}$-bounded S-multicuts F_1 and F_2 for G_1 and G_2.*

Proof. First we prove that there cannot be smaller solution than $F_1 \cup F_2$. To see this observe that for every $\mathbf{a}$-bounded S-multicut F' on G it holds that $F' \cap E_1$ is an $\mathbf{a}$-bounded S-multicut on G_1 (and vice versa for G_2). Hence, if F' would be a cut of smaller size than F, we would get a contradiction with the minimality of choice of F_1 and F_2, because we would have $|F'| < |F| = |F_1| + |F_2|$.

Now we prove that $F = F_1 \cup F_2$ is a valid solution. To see this we prove that every path between two terminals is not shorter than L. Let P be a path in $(V(G), E(G) \setminus F)$ between terminals $s, t \in S$. We prove that the length of P is at least $a_{s,t} + 1$ by an induction over a number $h := |V(P) \cap S|$. If $h = 2$ then because G_1 and G_2 are edge disjoint, we may (by symmetry) assume that $P \subset G_1$. Therefore, the length of P is at least $a_{s,t} + 1$ because F_1 is a valid solution.

If $h > 2$ then there is a vertex $u \in S \setminus \{s, t\}$ such that the path P is composed from two segments P_1 and P_2, where P_1 is a path between s and u and P_2 is a path between u and t. Thus, by induction hypothesis and sharp triangle inequalities, we have $|P| = |P_1| + |P_2| \geq a_{s,u} + 1 + a_{u,t} + 1 \geq a_{s,t} + 1$, what was to be demonstrated. □

We use dynamic programming techniques on a grafted tree decomposition $\mathcal{T}$ of an input graph. First, we want to root the decomposition $\mathcal{T}$ in a node containing both source and sink of the L-cut problem. This can be achieved by adding the source to all nodes on the unique path in the decomposition tree between any node containing the source and any node containing the sink. Note that this may add at most 1 to the width of the decomposition.

We solve the L-cut by reducing it to simple instances of generalized MLBMC problem.

We reduce the problem to the $\mathbf{a}$-bounded S-multicut for k terminals, where $k = tw(G) + 1$ (the additional one is for changing the decomposition).

Let $X = \{x_1, \ldots, x_k\}$ be a set of vertices, $\mathbf{a}$ be a length constraint, let $I \subset [k]$ and let $Y = \{x_i \in X \colon i \in I\}$. By $\mathbf{a}|_Y$ we denote the length constraint $\mathbf{a}$ containing a_{x_i,x_j} if and only if both $i \in I$ and $j \in I$ (in an appropriate order) – in this case we say $\mathbf{a}|_Y$ is $\mathbf{a}$ *contracted* on the set Y.

Dynamic programming tables Recall that for each node X of a tree decomposition we have defined the auxiliary graph G_X (see Section 2.2 for the definition). With a node X we associate the table Tab_X. The table entry for length constraints $\mathbf{a} = (a_{x_1,x_2}, \ldots, a_{x_{k-1},x_k})$ of Tab_X (denoted by $Tab_X[\mathbf{a}]$) for the node $X = \{x_1, \ldots, x_k\}$ contains the size of a minimum $\mathbf{a}$-bounded X-multicut for the set X in the graph G_X. Note that for two length constraints $\mathbf{a} \preceq \mathbf{b}$ it holds that $Tab_X[\mathbf{a}] \leq Tab_X[\mathbf{b}]$.

2.3.1 Node lemmata

The leaf nodes are the only nodes bearing some edges. We use an exhaustive search procedure for building tables for these nodes. For this we need to compute the lengths of the shortest paths between all the vertices of the leaf node, for which we use the well known procedure due to Floyd and Warshall [34, 75]:

Proposition 10 ([34, 75])**.** *Let G be a graph with nonnegative length $f : G(E) \to \mathbb{N}$. It is possible to compute the table of lengths of the shortest paths between any pair $u, v \in V(G)$ with respect to f in time $O(|V(G)|^3)$.*

Lemma 11 (Leaf Nodes)**.** *For all L-limited length constraints and a leaf node X the table Tab_X of sizes of minimum length-bounded multicuts can be computed in time $O\left(L^{k^2} \cdot 2^{k^2} \cdot k^3\right)$, where $k = |X|$.*

Proof. Fix one L-limited length constraint $\mathbf{a}$. Let $G_X = (V, E)$ and $F \subseteq E$. We run the Floyd-Warshall algorithm (stated as Proposition 10) in graph $G_X^F = (V, E \setminus F)$. We check all $O(k^2)$ pairs of terminals if their distance is sufficiently large, i.e. if F is an $\mathbf{a}$-bounded S-multicut.

We iterate over all subsets of E and pick the minimum cut. As $|E| \leq \binom{k}{2}$ there are $O\left(2^{k^2}\right)$ choices for F. This gives us a running time $O\left(2^{k^2} \cdot k^3\right)$ for a single length constraint $\mathbf{a}$. We set the entry for $\mathbf{a}$ in Tab_X as

$$Tab_X[\mathbf{a}] := \min_{F \subseteq E \colon F \text{ is a } \mathbf{a}\text{-bounded } S\text{-multicut}} |F|.$$

Finally there are $O\left(L^{k^2}\right)$ L-limited length constraints, this gives our result. □

We now use Lemma 9 to prove time complexity of finding a dynamic programming table for join nodes from the table of its children.

Lemma 12 (Join Nodes)**.** *Let X be a join node with children Y and Z, let L be the limit on length constraints and let $k = |X|$. Then the table Tab_X can be computed in time $O\left(L^{k^2}\right)$ from the table Tab_Y and Tab_Z.*

Proof. Recall that graphs G_Y and G_Z are edge disjoint and that we store sizes of $\mathbf{a}$-bounded multicuts. Note also that $B_X = V(G_Y) \cap V(G_Z)$ and so we can apply Lemma 9 and set $Tab_X[\mathbf{a}] := Tab_Y[\mathbf{a}] + Tab_Z[\mathbf{a}]$, for each $\mathbf{a}$ satisfying the triangle inequalities. As there are $O\left(L^{k^2}\right)$ entries in the table Tab_X we have the complexity we wanted to prove. □

As the forget node represents forgetting a vertex, its table can be calculated by forgetting part of the table of the child node.

Lemma 13 (Forget Nodes)**.** *Let X be a forget node, Y its child, let L be the limit on length constraints and let $k = |X|$. Then the table Tab_X can be computed in time $O\left(L^{2k^2}\right)$ from the table Tab_Y.*

Proof. Fix one length constraint $\mathbf{a}$ and compute the set $\mathcal{A}(\mathbf{a})$ of all B_Y-augmented length constraints. Formally, $\mathbf{b} \in \mathcal{A}(\mathbf{a})$ if $\mathbf{b}$ is a length constraint for B_Y and $\mathbf{b}|_{B_X} = \mathbf{a}$. After this we set

$$Tab_X[\mathbf{a}] := \min_{\mathbf{b} \in \mathcal{A}(\mathbf{a})} Tab_Y[\mathbf{b}].$$

In the worst case for every entry in Tab_X we search whole Tab_Y, which gives us the claimed time. □

Also the introduce node (as the counter part for the forget node) only adds coordinates to the table of its child. It does no computation as there are no edges it can decide about – these nodes now only add isolated vertices to the auxiliary graph.

Lemma 14 (Introduce Nodes)**.** *Let X be an introduce node, Y its child, let L be the limit on length constraints and let $k = |X|$. Then the table Tab_X can be computed in time $O\left(L^{k^2}\right)$ from the table Tab_Y.*

Proof. Let v be the introduced vertex, i.e. $v \in B_X \setminus B_Y$. Let $\mathcal{T}'$ be a subtree of the grafted tree decomposition $\mathcal{T}$ rooted in X. Recall that each edge of the auxiliary graph G_X belongs to some leaf node of $\mathcal{T}'$. Since v is introduced in the root of $\mathcal{T}'$, there is no leaf node of $\mathcal{T}'$ containing v. Thus, there is no edge incident to v in G_X. Therefore, we can set $Tab_X[\mathbf{a}] := Tab_Y[\mathbf{a}|_{B_Y}]$, because v is arbitrarily far from any vertex in G_Y, especially from the set B_Y. □

2.3.2 Proofs of Theorems

We use Lemma 11, 12, 13 and 14 to prove Theorem 7.

Theorem 15. *Let $G = (V, E)$ be a graph of tree-width k, $S \subseteq V$ with $|S| = q$ and let $p := q + k - 1$. Then for any $L \in \mathbb{N}$ and any L-limited length constraint $\mathbf{b} \in [L]^{\binom{S}{2}}$ a minimum $\mathbf{b}$-bounded S-multicut can be computed in time $O(L^{3p^2} \cdot n)$.*

Proof. First, we compute all L-limited length-constraints in advance to work with constraints instead of vectors that do not have to fulfill triangle inequalities. This takes additional time $O\left(L^{k^2} \cdot k^3\right)$ which can be upper-bounded by $O\left(L^{2k^2}\right)$ for $L \geq 2$, which we can suppose because the problem for $L = 1$ can be trivially computed in linear time.

We create grafted tree decomposition $\mathcal{T}$ in linear time (see Section 2.2). We need that all terminals would be in the root of $\mathcal{T}$. Let S' be a set S without one vertex. We add the set S' to every bag of the decomposition $\mathcal{T}$ – this increases the width of $\mathcal{T}$ by at most $|S'| = q - 1$. Thus, there exists a node R of $\mathcal{T}$ such that $S \subseteq B_R$. We rooted the decomposition $\mathcal{T}$ in R.

We compute Tab_X for all nodes X of the decomposition $\mathcal{T}$ using Lemma 11, 12, 13 and 14. Now, we are able to read the value of the solution from Tab_R. Since $L \geq 2$ and size of every bag in $\mathcal{T}$ is at most $p = q + k$, we can upper-bound the processing time for any type of node by $O\left(L^{3p^2}\right)$. There are $O(n)$ nodes in $\mathcal{T}$ which gives us the claimed time. □

Theorem 4 and Theorem 7 are corollaries of Theorem 15.

Proof of Theorem 4. Since for two terminals the notions of length vector and length constraint are the same, we can use Theorem 15 to prove Theorem 4. For $\mathrm{tw}(G) = 1$ (G is a tree) can be the problem computed trivially in linear time. Otherwise for $\mathrm{tw}(G) = k >= 2$, we have running time $O\left(L^{3(k+1)^2} \cdot n\right)$ which can be upper-bounded by $O\left(L^{6k^2} \cdot n\right)$ as claimed. □

Proof of Theorem 7. Let $(G, S, \mathbf{a})$ be an input and $p = \mathrm{tw}(G) + |S|$. By Observation 8 we know that there exists a length constraints $\mathbf{b}$ such that instances $(G, S, \mathbf{a})$ and $(G, S\mathbf{b})$ has the same solution. Therefore, it suffice to try all length constraints $\mathbf{c} \succeq \mathbf{a}$ and select the smallest computed cut. By Theorem 15, the processing for one length constraint is $O\left(L^{3p^2} \cdot n\right)$. There are at most $L^{|S|^2}$ length constraints which gives us the claimed time $O\left(L^{4p^2} \cdot n\right)$. □

2.4 Hardness of the L-bounded Cut

In this section we prove that Decision version of Length-Bounded Cut parameterized by path-width is W[1]-hard by FPT-reduction from k-Multicolor Clique.

Decision version of Length-Bounded Cut (DLBC)	
Input:	graph $G = (V, E)$, vertices s, t, positive integers L, K
Task:	Find an L-bounded cut of size at most K

Notation In this section, sets $V_1, \ldots, V_k$ are always partites of the k-partite graph G. We denote edges between V_i and V_j by E_{ij}. The problem is W[1]-hard [21] even if every independent set V_i has the same size and the number of edges between every V_i and V_j is the same. Throughout this section we denote the size of an arbitrary V_i by N and the size of an arbitrary E_{ij} by M. For an FPT-reduction from k-Multicolor Clique to DLBC parameterized by path-width we need the following steps:

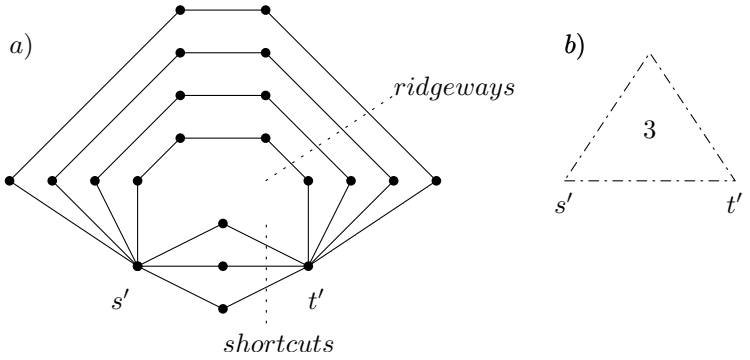

Figure 2.2: a) Example of a butte for $h = 3$ and $Q = 4$. b) Simple diagram for a butte of height 3.

1. Create an DLBC instance $G' = (V', E'), s, t, L, K$ from the k-Multicolor Clique instance $G = (V_1 \dot\cup V_2 \dot\cup \ldots \dot\cup V_k, E)$ in time $f(k)|G|^{O(1)}$ for a computable function f.

2. Prove that G contains a k-clique if and only if G' contains an L-bounded cut of the size K.

3. Prove the path-width of G' is smaller than $g(k)$ where g is a computable function.

Our ideas were inspired by work of Michael Dom et al. [24]. They proved W[1] hardness of Capacitated Vertex Cover and Capacitated Dominating Set parameterized by the tree-width of the input graph. We remark that their reduction also proves W[1] hardness of these problems parameterized by path-width.

2.4.1 Basic gadget

In the k-Multicolor Clique problem we need to select exactly one vertex from each independent set V_i and exactly one edge from each E_{ij}. Moreover, we have to make certain that if $e \in E_{ij}$ is a selected edge and $u \in V_i, v \in V_j$ are selected vertices. then $e = \{u, v\}$. The idea of the reduction is to have a basic gadget for every vertex and edge. We connect gadgets g_v for every v in V_i into a path P_i. The path P_i is cut in the gadget g_v if and only if the vertex $v \in V_i$ is selected into the clique. The same idea will be used for selecting the edges.

Definition 17. *Let $h, Q \in \mathbb{N}$. Butte $B(s', t', h, Q)$ is a graph which contains h paths of length 2 and Q paths of length $h + 2$ between the vertices s' and t'. The short paths (of length 2) are called shortcuts, the long paths are called ridgeways and the parameter h is called height.*

A butte for $h = 3, Q = 4$ is shown in Figure 2.2 part a. In our reduction all buttes will have the same parameter Q (it will be computed later). For simplicity we depict buttes as a dash-dotted line triangles with their height h inside (see Figure 2.2 part b), or only as triangles without the height if it is not important.

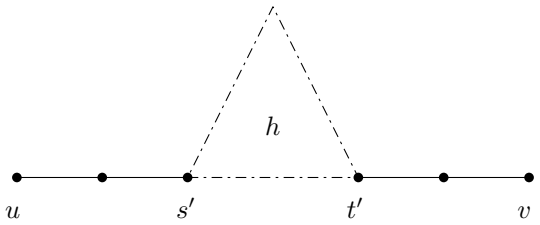

Figure 2.3: Example of a path going through a butte.

Let $B(s', t', h, Q)$ be a butte. We denote by $s(B), t(B), h(B), Q(B)$ the parameters of butte B s', t', h and Q, respectively. We state an easy but important observation about the butte path-width:

Observation 16. *Path-width of an arbitrary butte B is at most 3.*

Proof. If we remove vertices $s(B)$ and $t(B)$ from B we get $Q(B)$ paths from ridgeways and $h(B)$ isolated vertices from shortcuts. This graph certainly has path-width 1. If we add $s(B)$ and $t(B)$ to every node of the path decomposition we get a proper path decomposition of B with width 3. □

Let butte $B(s', t', h, Q)$ be a subgraph of a graph G. Let u, v be vertices of G such that all paths between u and v going through B enter into B in s' and leave it in t' (see Figure 2.3). The important properties of the butte B are:

1. By removing one edge from all h shortcuts of B, we extend the distance between u and v by h. If a cut C contains one edge of every shortcut of butte B we say the cut C *ridges* the butte B.

2. Suppose the size of a cut C is bounded by $K \in \mathbb{N}$ and C contains only edges in B. If we increase Q to be bigger than K then C cannot separate u and v (if C ridges B, then distance between u and v is only extended).

2.4.2 Butte path

In this section we define how we connect buttes into a path, which we call highland. The main idea is to have highland for every pair $(i, j), i \neq j \in [k]$. In the highland for (i, j), there are buttes for every vertex $v \in V_i$ and every edge $e \in E_{i,j}$. We connect vertex buttes and edge buttes into a path. Then we set the butte heights and limit the size of the cut in such a way that:

1. Exactly one vertex butte and exactly one edge butte have to be ridged.

2. If a butte for a vertex v is ridged, then only buttes for edges incident with v can be ridged.

The formal description of a highland is in the following definition.

Definition 18. *Let $X, Y \in \mathbb{N}$. A highland $H(X, Y, s, t)$ is a graph containing 2 vertices s and t and $Z = X + Y$ buttes $B_1, \ldots, B_Z$ where:*

1. *$s = s(B_1), t = t(B_Z)$ and $t(B_i) = s(B_{i+1})$ for every $1 \leq i < Z$.*

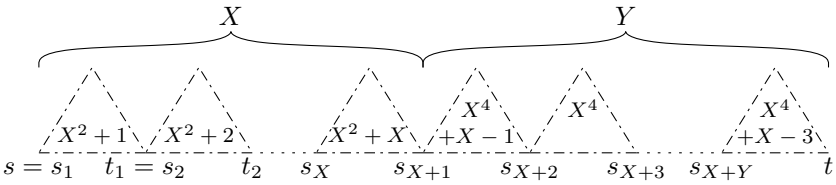

Figure 2.4: Example of a highland $H(X, Y, s, t)$.

2. $h(B_i) = X^2 + i$ *for* $1 \leq i \leq X$.

3. $h(B_i) \in \{X^4, \ldots, X^4 + X - 1\}$ *for* $X + 1 \leq i \leq Z$.

4. $Q(B_i) = X^4 + X^2$ *for every* i.

Let $H(X, Y, s, t)$ be a highland. We call buttes $B_1, \ldots, B_X$ from H low and buttes $B_{X+1}, \ldots, B_{X+Y}$ high (low buttes will be used for the vertices and high buttes for the edges). The vertex $t(B_X) = s(B_{X+1})$, where low and high buttes meet, is called the *center* of highland H. Note that there can be more buttes with the same height among high buttes and they are not ordered by height as the low buttes. An example of a highland is shown in Figure 2.4.

Proposition 17. *Let $H(X, Y, s, t)$ be a highland. Let $L = 2(X + Y) + X^4 + X^2 + X - 1$. Let C be an L-cut of size $X^4 + X^2 + X$, which cuts all paths of length L and shorter between s and t then:*

1. *The cut C ridges exactly two buttes B_i, B_j, such that B_i is low and B_j is high.*

2. *Let B_i be the ridged low butte and B_j be the ridged high butte. Then, $h(B_j) = X^4 + X - i$.*

Proof. Every butte has at least X^2+1 shortcuts and X^4+X^2 ridgeways. Therefore, C can not cut all paths in H between s and t and it is useless to add edges from ridgeways to the cut C. Note that the shortest st-path in H has the length $2(X + Y)$.

1. If the cut C ridges every low butte then the shortest st-path is extended by $\sum_{i=1}^{X}(X^2 + i) = X^3 + \frac{X^2}{2} + \frac{X}{2}$. However, it is not enough and at least one high butte has to be ridged. Two high buttes cannot be ridged otherwise the cut would be bigger than the bound. No high butte can extend the shortest st-path enough, therefore at least one low butte has to be ridged. However, two low buttes and one high butte cannot be ridged because the cut C would be bigger than the bound.

2. The height of ridged low butte B_i is $X^2 + i$. Therefore, the length of the shortest st-path when the edges in C are removed is $2(X+Y)+X^2+i+h(B_j)$ and the size of C is $X^2 + i + h(B_j)$. If $h(B_j) < X^4 + X - i$ then shortest st-path is strictly shorter then $2(X + Y) + X^4 + X^2 + X$. Thus, C is not an L-cut. If $h(B_j) > X^4 + X - i$ then $|C| > X^4 + X^2 + X$ which is bigger than the bound.

□

2.4.3 Reduction

In this section we present our reduction. Let $G = (V_1 \dot\cup V_2 \dot\cup \ldots \dot\cup V_k, E)$ be the input for k-Multicolor Clique. As we stated in the last section, the main idea is to have a low butte B_v for every vertex $v \in V(G)$ and a high butte B_e for every edge $e \in E(G)$. Vertex v and edge e is selected into the k-clique if and only if the butte B_v and the butte B_e are ridged. From G we construct MLBC input G', s, t, L (the construction is quite technical, for better understanding see Figure 2.5):

1. For every $1 \leq i, j \leq k, i \neq j$ we create highland $H^{i,j}(N, M, s, t)$ of buttes $B_1^{i,j}, \ldots, B_{N+M}^{i,j}$.

2. Let $V_i = \{v_1, \ldots, v_N\}$. The vertex $v_\ell \in V_i$ is represented by the low butte $B_\ell^{i,j}$ of the highland $H^{i,j}$ for every $j \neq i$. Thus, we have $k-1$ copies of buttes (in different highlands) for every vertex. Hence, we need to be certain that only buttes representing the same vertex are ridged. Note that buttes representing the same vertex have the same height and the same distance from the vertex s.

3. Let $E_{ij} = \{e_1, \ldots, e_M\}, i < j$. Edge $e_\ell = \{u, v\} \in E_{ij} (u \in V_i, v \in V_j)$ is represented by the high butte $B_{N+\ell}^{i,j}$ of the highland $H^{i,j}$ and by the high butte $B_{N+\ell}^{j,i}$ of the highland $H^{j,i}$. Note that two buttes representing the same edge have same distance from the vertex s. Let h_i, h_j be the heights of buttes representing the vertices u and v, respectively. We set the buttes heights:

   (a) $h(B_{N+\ell}^{i,j}) = N^4 + N - h_i$

   (b) $h(B_{N+\ell}^{j,i}) = N^4 + N - h_j$

4. We add edge $\left\{t(B_\ell^{i,j}), t(B_\ell^{i,j+1})\right\}$ for every $1 \leq i \leq k, 1 \leq j < k, i \neq j$ and $1 \leq \ell < N$.

5. We add paths of length $N-1$ connecting $t(B_\ell^{i,j})$ and $t(B_\ell^{j,i})$ for every $1 \leq i, j \leq k, i \neq j$ and $N+1 \leq \ell < N+M$.

6. We set L to $2(N+M) + N^4 + N^2 + N - 1$.

We call paths between highlands in Items 4 and 5 the *valley paths*.

Observation 18. *Graph G' can be computed in polynomial time in the size of graph G.*

Theorem 19. *If graph G has a clique of size k then (G', s, t) has an L-cut of size $k(k-1)(N^4 + N^2 + N)$.*

Proof. Suppose G has a k-clique $\{v_1, \ldots, v_k\}$ where $v_i \in V_i$ for every i and $e_{ij} = \{v_i, v_j\} \in E_{ij}$. We create an L-cut C. For every i the cut C ridges all $k-1$ buttes representing the vertex v_i in G'. And for every $i < j$ the cut C ridges both buttes representing the edge e_{ij}.

We claim that the set C is an L-cut. Let $H^{i,j}$ be an arbitrary highland. We show there is no st-path of length at most L in $H^{i,j}$. Let $h(B_v) = N^2 + \ell$ where

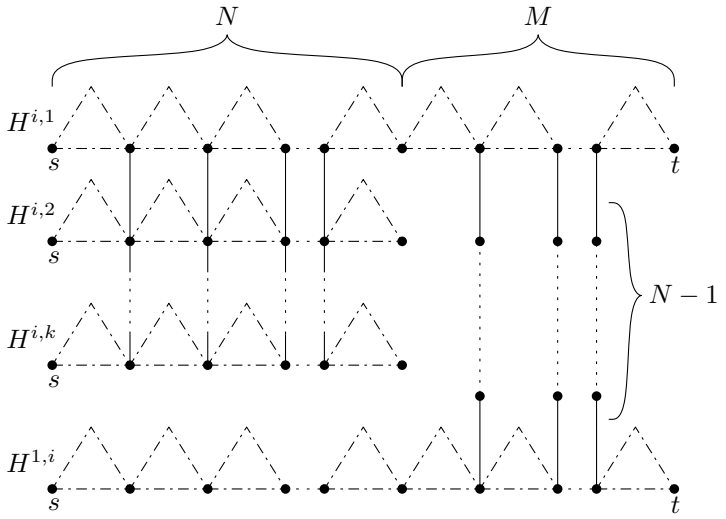

Figure 2.5: Some part of the graph G'. All vertices labeled s and t are actually two vertices s and t in the graph G'. We divided them for better illustration. Highlands $H^{i,2}$ and $H^{i,k}$ have also high buttes, but we omitted them.

B_v is an arbitrary butte representing the vertex v_i. By construction of G', the high butte representing the edge e_{ij} in $H^{i,j}$ has height $N^4 + N - \ell$. Thus, ridged buttes in $H^{i,j}$ extend the shortest st-path by $N^4 + N^2 + N$ and it has length $2(M + N) + N^4 + N^2 + N$. Buttes representing the vertex v_i have the same height. Thus, a path through the low buttes of highlands using some valley path is always longer than a path going through low buttes of only one highland. Therefore, it is useless to use valley paths among low buttes for the shortest st-path.

The remaining paths to consider are those using valley paths among high buttes, because buttes representing the same edge have different heights. The butte B_v representing the vertex v_i extends the shortest path at least by $N^2 + 1$. The butte B_e representing the edge $e_{i,j}$ extends the shortest at least by N^4. However, if $h(B_v) + h(B_e) < N^4 + N^2 + N$ then B_v and B_e have to be in different highlands. Therefore, the st-path going through B_v and B_e has to use a valley path between high buttes, which has length $N - 1$. Hence, any st-path has length at least $2(N + M) + N^4 + N^2 + N$.

We remove $N^4 + N^2 + N$ edges from each highland and there are $k(k-1)$ highlands in G'. Therefore, G' has L-cut of the size $k(k-1)(N^4 + N^2 + N)$. □

Theorem 20. *If (G', s, t) has an L-cut of size $k(k-1)(N^4 + N^2 + N)$ then G has a clique of size k.*

Proof. Let C be an L-cut of G'. Every shortest st-path going through every highland has to be extended by $N^4 + N^2 + N$. By Proposition 17 (Item 1), exactly one low butte and exactly one high butte of each highland has to be ridged. By Proposition 17 (Item 2) we remove $(N^4 + N^2 + N)$ from every highland in G'. Therefore, there can be only edges from ridged buttes in C.

For fixed i, highlands $H^{i,j}$ are the highlands whose low buttes represent vertices from V_i. We claim that ridged low buttes of $H^{i,1}, \ldots, H^{i,k}$ represent the same vertex. Suppose for contradiction, there exist two low ridged buttes B_ℓ of $H^{i,\ell}$ and B_m of $H^{i,m}$ which represent different vertices from V_i. It follows that there

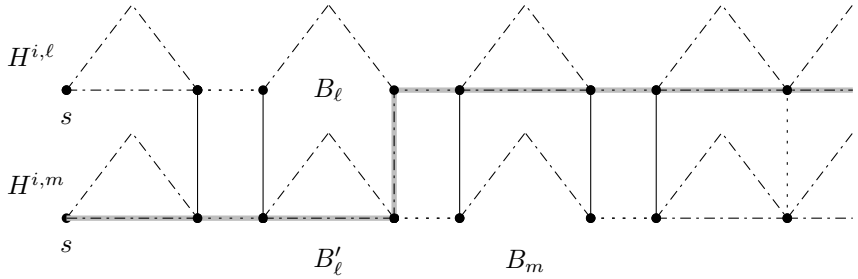

Figure 2.6: How to miss every ridged low butte if there are ridged two low buttes representing two different vertices from one color class. Ridged butte is depicted as triangle without hypotenuse.

have to be two highlands $H^{i,p}, H^{i,p+1}$ such that their ridged low buttes represent different vertices. Thus, we can suppose that $H^{i,\ell}$ and $H^{i,m}$ are next to each other (i.e. $|\ell - m| = 1$) and the distance from s to $s(B_\ell)$ is smaller than the distance from s to $s(B_m)$. Let B'_ℓ be a butte of $H^{i,m}$ such that it has the same distance from s as the butte B_ℓ (see Figure 2.6). The path s–$t(B'_\ell)$–$t(B_\ell)$–t does not go through any ridged low butte. Therefore, this path is shorter than L, which is contradiction. We can use the same argument to show that there are not two high ridged buttes of highland $H^{i,j}$ and $H^{j,i}$ which represent different edges from E_{ij}.

We put into the k-clique $K \subset V(G)$ the vertex $v_i \in V_i$ if and only if an arbitrary butte representing the vertex v_i is ridged. We proved in the previous paragraph that exactly one vertex from V_i can be put into the clique K. Let $e_{ij} \in E_{ij}$ be an edge represented by ridged high buttes. We claim that $v_i \in e_{ij}$. Let $B \in H^{i,j}$ be a butte representing v_i with height $N^2 + \ell$. Then by Proposition 17 (Item 2), butte $B' \in H^{i,j}$ of height $N^4 + N - \ell$ has to be ridged. By construction of G', only buttes representing edges incident with v_i have such height. Therefore, chosen edges are incident with chosen vertices and they form the k-clique of the graph G. □

Observation 21. *Graph G' has path-width in $O(k^2)$.*

Proof. Let H be a graph created from G by replacing every butte by a single edge and contract the valley paths between high buttes into single edges, see Figure 2.7 transformation a. Let U be a vertex set containing s, t and every highland center. Let H' be a graph created from H by removing all vertices from U, see Figure 2.7 transformation b.

Graph H' is unconnected and it contains k grids of size $k \times (N-2)$ and $\binom{k}{2}$ grids of size $2 \times (M-2)$. Path-width of $(k-1) \times (N-2)$ grids is in $O(k)$, therefore $\mathrm{pw}(H') \in O(k)$. If we add set U to every node of a path decomposition of H' we get proper path decomposition of H. Since $|U| \in O(k^2)$, path-width of H is in $O(k^2)$. The edge subdivision does not increase path-width. Moreover, replacing edges by buttes does not increase it either (up to multiplication constant) because butte has the constant path-width (Observation 16). Therefore, $\mathrm{pw}(G) = c\,\mathrm{pw}(H)$ for some constant c and $\mathrm{pw}(G) \in O(k^2)$.

□

Thus Theorem 5 easily follows from Observations 18 and 21 and Theorems 19 and 20.

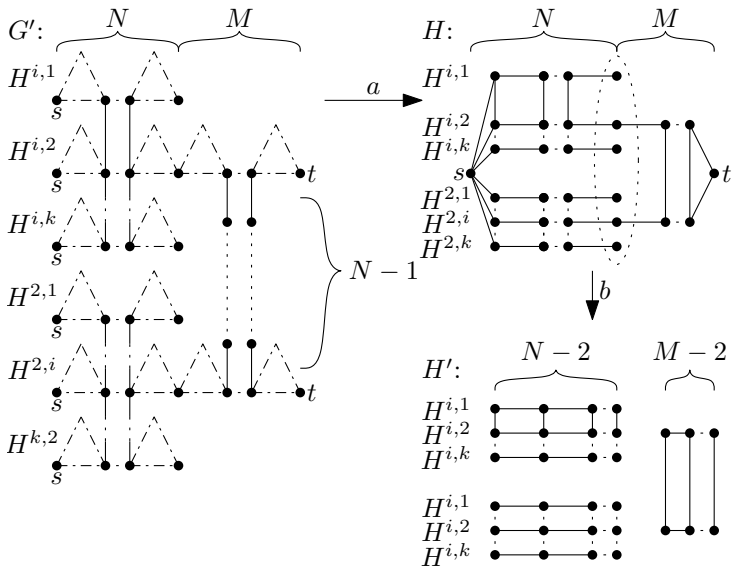

Figure 2.7: The transformation a replaces all buttes in G' by single edges and contract long valley paths into single edges. The transformation b removes vertices s and t and all highland centers (highlighted by dotted ellipse) from H.

2.5 Polynomial kernel is questionable

In this section, we prove that the Minimum Length Bounded Cut problem is unlikely to admit a polynomial kernel when parameterized by the length L and the path-width (tree-width) of the input graph. We will prove this fact by the use of an AND-cross-composition framework – that is by designing an AND-cross-composition algorithm from the Multicolor Clique problem:

Multicolor Clique	
Input:	k-partite graph G and positive integer k
Task:	find a clique of size k as a subgraph of the graph G

It is unparameterized version of k-Multicolor Clique from the previous section. The problem is NP-hard by well known reduction from the Clique problem [21]. Note that in the input graph G all cliques have size at most k. I.e., an instance (G, k) is an *yes*-instance if and only if the largest clique of G has the largest possible size k.

We define the polynomial equivalence relation $\mathcal{R}$ as follows. Two instances of the Multicolor Clique problem $(G, k), (G', k')$ are equivalent if $|V(G)| = |V(G')|, |E(G)| = |E(G')|$ and $k = k'$. It is clear that $\mathcal{R}$ is a polynomial equivalence relation.

AND-cross-composition We take the instances $(G_1, k), (G_2, k), \ldots, (G_r, k)$ that are equivalent under the relation $\mathcal{R}$. To every instance (G_i, k) we apply our reduction described in the previous section and get an instance $(G'_i, s_i, t_i, L_i, K_i)$

of DLBC problem. Let $n = |V(G_i)|, m = |E(G_i)|$. By the reduction,

$$L_i = 2(n+m) + n^4 + n^2 + n - 1$$
$$K_i = k(k-1)(n^4 + n^2 + n).$$

Since all instances of Multicolor Clique are equivalent under $\mathcal{R}$, for all $i, j \leq r$ holds that $L_i = L_j$ and $K_i = K_j$. Thus, all instances of DLBC has a form (G'_i, s_i, t_i, L, K). We take the disjoint union of graphs $G'_1, G'_2, \ldots, G'_r$ and unify all sources s_i to a vertex s and sinks t_i to a vertex t of the resulting graph and denote the graph as G. As we build the AND-cross-composition we set the budget of the resulting instance to $r \cdot K$, i.e., we have an instance (G, s, t, L, rK) of DLCB problem.

By the reduction, the minimum L-bounded cut in every graph G'_i has size at least K. Therefore, the graph G has the minimum L-bounded cut of size at least rK. If all instances (G_i, k) of Multicolor Clique are *yes*-instances, then there is an L-bounded cut in the graph G of size rK. If there is an *no*-instance (G_j, k) of Multicolor Clique, then every L-bounded cut in G'_j has size at least $K + 1$. Thus, every L-bounded cut in the graph G has size at least $rK + 1$.

So far we created an instance $\mathcal{I}$ of MLBC from r instances $(G_1, k), \ldots, (G_r, k)$ of Multicolor Clique and the instance $\mathcal{I}$ is *yes*-instance if and only if all instances (G_i, k) are *yes* instances. It remains to bound the parameters path-width of G and L. It is discussed above that L is polynomial in size of G_i. It is easy to see that the path-width of the graph G of our construction is at most

$$\max_{i=1,2,\ldots,r} |V(G'_i)| - 1.$$

We can put each graph G'_i into a bag B_i and connect them into a path. Only two common vertices among bags are the vertices s and t, which arise by unifying vertices s_i, t_i respectively. Thus, s and t are in all bags and we described the correct path decomposition such that $|B_i| = |V(G'_i)|$.

Thus, we have AND-cross-composition from NP-hard problem to DLBC parameterized by path-width and L. By Theorem 1, we can refute existence of polynomial kernel for DLBC parameterized by path-width and L and this finishes the proof of Theorem 6.

3. Target Set Selection in Dense Graph Classes

3.1 Introduction

We study the Target Set Selection problem, with notation according to by Kempe et al. [51], from the area of computational social choice from a parameterized complexity perspective. Let $G = (V, E)$ be a graph, $S \subseteq V$, and $f\colon V \to \mathbb{N}$ be a *threshold function*. The *activation process* arising from the set $S_0 = S$ is an iterative process with resulting sets $S_0, S_1, \ldots$ such that for $i \geq 0$

$$S_{i+1} = S_i \cup \{v \in V : |N(v) \cap S_i| \geq f(v)\},$$

where by $N(v)$ we denote the set of vertices adjacent to v. Note that after at most $n = |V|$ rounds the activation process has to stabilize – that is, $S_n = S_{n+i}$ for all $i > 0$. We say that the set S is a *target set* if for the activation process $S = S_0, \ldots, S_n$ holds that $S_n = V$.

Target Set Selection	
Input:	graph $G = (V, E)$, $f\colon V \to \mathbb{N}$ and a positive integer $b \in \mathbb{N}$
Task:	find a target set $S \subseteq V$ of size at most b or report that there is no such set

We call the input integer b the *budget*. The problem interpretation and computational complexity clearly may vary depending on the input function f. There are three important setting studied – namely constant, majority, and general function. If the threshold function f is the majority (i.e., $f(u) = \lceil \deg(u)/2 \rceil$ for every vertex $u \in V$) we denote the problem as Majority Target Set Selection.

Distance to Triviality. There are many natural parameterizations considered nowadays in the parameterized complexity studies – among these the size of the solution set and the structural parameters play the most significant role. Sometimes another very important parameter – distance to triviality – is considered. There are two major examples of this parameter use either the parameter of value k expresses that after removal of k vertices the input graph is turned in a graph belonging to a class of graphs on which the problem under consideration becomes trivial (polynomial time solvable) or it may be viewed as the distance from guarantee, as for example the guarantee given by rounding a relaxation of the integer linear program [60]. In this work we use the structural parameters suitable for dense graphs, however Chopin et al. [12] already observed that the Target Set Selection problem can be trivially solved on cliques and thus our structural parameters may be viewed as a distance to triviality in this context.

Motivation. The Target Set Selection problem was introduced by Domingos and Richardson [25] in order to allow study of influence of direct marketing on a social network. It is noted therein that it captures e.g. *viral marketing* [68]. The

Target Set Selection problem is important also from the graph theoretic viewpoint as it generalizes many well known NP-hard problems on graphs.

Previous Results. The Target Set Selection problem received an attention of researchers in theoretical computer science in the past years. A general lower bound on the number of selected vertices under majority constraints is $|V|/2$ [1]. The Target Set Selection problem admits an FPT algorithm when parameterized by the vertex cover number [64]. An $t^{\mathcal{O}(w)}\,\mathrm{poly}(n)$ algorithm is known where w is the tree-width of the input graph and t is an upper-bound on the threshold function [4], that is $f(v) \leq t$ for every vertex v. This is essentially optimal, as the Target Set Selection problem parameterized by path-width is W[1]-hard for majority [12] and general functions [4]. The Target Set Selection problem is solvable in linear time on trees [10] and more general on block-cacti graphs [11]. The optimization variant of the Target Set Selection problem is hard to approximate [10] within a polylogarithmic factor. For more and not recent results we refer the reader to a survey by Peleg [66]. Cicalese et al. [14, 13], considered versions of the problem in which the number of rounds of the activation process is bounded. For graphs of bounded clique-width, given parameters a, b, ℓ, they gave polynomial-time algorithms to determine whether there exists a target set of size b, such that at least a vertices are activated in at most ℓ rounds.

Our Results. In this work we generalize some results obtained by Nichterlein et al. [64]. Chopin et al. [12] essentially proved that in sparse graph classes (such as graphs with bounded tree-width) parameterized complexity of the Majority Target Set Selection problem is the same as for the Target Set Selection problem. For these graph classes, it is not hard to see that e.g. if the threshold for vertex v is set above the majority (i.e., $f(v) > \lceil \deg(v)/2 \rceil$), then we may add $2\Big(f(v) - \lceil \deg(v)/2 \rceil\Big)$ vertices neighboring with v only and the parameter stays unchanged. However, this is not true in general for dense graph classes. We demonstrate this phenomenon for the parameterization by neighborhood diversity. We show parameterized algorithms for a function which generalizes constant and majority functions. We call this function uniform (and the corresponding problem Uniform Target Set Selection), see the next section for a proper definition. Roughly speaking all vertices belonging to a same class of a graph decomposition must possess the same value of threshold function. In slight contrast to previous results, we derive an FPT algorithm that, instead of the maximal threshold value t, depends on the size of the image of the threshold function for graphs having bounded neighborhood diversity.

Theorem 22. *There is an* FPT *algorithm for the* Uniform Target Set Selection *problem parameterized by the neighborhood diversity of the input graph.*

Theorem 23. *The* Target Set Selection *problem is* W[1]-hard *parameterized by the neighborhood diversity of the input graph.*

The complexity of Majority Target Set Selection problem is not resolved for parameterization by the cluster vertex deletion number (the number of vertices whose removal from the graph results in a collection of disjoint cliques). We have a positive result in this direction that also assumes that each vertex we remove is completely adjacent to the whole clique or completely nonadjacent. This

result also suggest that various weighted variants of the Target Set Selection problem may be in FPT when parameterized by the vertex cover number.

Theorem 24. *There is an* FPT *algorithm for the* Uniform Target Set Selection *problem parameterized by the size of the twin cover.*

In contrast, previous results [12] imply that the parameterized complexity of the Target Set Selection and Majority Target Set Selection problems is the same in graphs with bounded clique-width. We show that this is already the case for parameterization by the (restricted) modular-width that generalizes both neighborhood diversity and twin cover number.

Theorem 25. *The* Majority Target Set Selection *problem is* W[1]-hard *parameterized by the modular-width of the input graph.*

Theorem 26. *The* Majority Target Set Selection *problem is* NP-hard *on class of graphs having shrub-depth* 3.

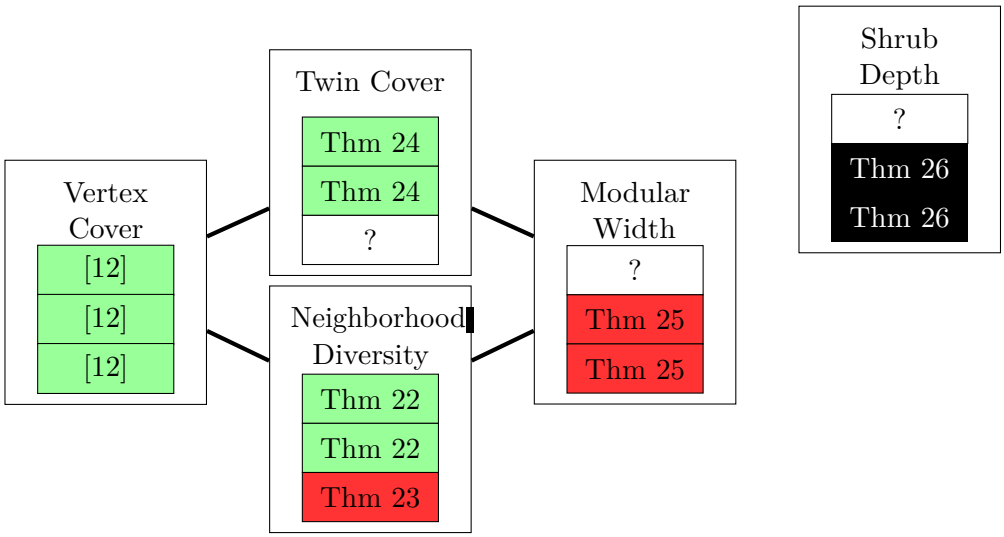

Figure 3.1: A map of considered parameterizations and specializations of the Target Set Selection problem. The three boxes represent from the top the constant threshold Target Set Selection, Majority Target Set Selection, and Target Set Selection problems. Green boxes represent FPT algorithms, red boxes represent W[1]-hardness result, and finally black boxes represent NP-hardness for constant value of the parameter.

3.2 Positive Results

In this section we give proofs of Theorem 22 and 24. In the first part we discuss the crucial property of dense structural parameters – uniformity of neighborhood. This, opposed to e.g. cluster vertex deletion number, allows us to design a parameterized algorithm.

Lemma 27. *Let $G = (V, E)$ be a graph, $S \subseteq V$ and f be a uniform function. Now let C be a twin clique or neighborhood diversity type. Let $S_0 = S, S_1, \ldots$ be the activation process arising from S. For each round $i \in \mathbb{N}_0$ one of the following holds:*

1. *$S_i \cap C = S_0 \cap C$, or*
2. *$S_i \cap C = C$.*

Moreover, there exist j with $j \in \mathbb{N}_0$ such that for C the first item applies in rounds $0, \ldots, j$ and the second in rounds $j + 1, \ldots$.

Proof. Since f is uniform, it is constant on C. The proof is by induction on the round number i. The statement clearly holds for $i = 0$. Suppose that lemma is valid for all $i' < i$ but not for i – this means that in the i-th round there are two vertices $u, v \in C$ such that $u \in S_i \setminus S_{i-1}$ but $v \notin S_i$. This is impossible as both u and v have the same threshold and the same neighborhood type. Thus if u gets activated, then v must be activated as well. The moreover part also follows easily. □

Let C be a twin clique or type of neighborhood diversity. For a threshold function f which is constant on C we define $f'(C)$ as $f(v)$ for arbitrary vertex v in C. By Lemma 27, we say that C is *activated in round i* if $S_i \cap C = C$ and $S_j \cap C = S_0 \cap C$ for every $j < i$. We denote $a_i^S(v)$ the number $|S_{i-1} \cap N(v)|$, i.e., the number of active neighbors of v in the round i in the activation process arising from the set S. Thus, a vertex v is activated in the first round i when holds $a_i^S(v) \geq f(v)$.

3.2.1 Uniformity and Twin Cover

In this subsection we present an algorithm for Uniform Target Set Selection parameterized by twin cover.

Trivial Bounds on the Minimum Target Set. Let $G = (V, E)$ be a graph with twin cover T of size t and let $C_1, C_2, \ldots, C_q$ be twin cliques of G. For a twin clique C by $N(C)$ we denote the common twin cover neighborhood, that is $N(v) \cap T$ for any $v \in V(C)$. We show that there are only small number of possibilities how the optimal target set can look like. Let $b_C = \max(f'(C) - |N(C)|, 0)$ for a twin clique C. The meaning of b_C is that we need to select at least b_C vertices of the twin clique C to any target set.

Observation 28. *Suppose the minimum target set of G has size s. For $b' = \sum_{i=1}^{q} b_{C_i}$ holds that $b' \leq s \leq b' + t$.*

Proof. Suppose there is a twin clique C such that $|S \cap C| = p < b_C$. It means that $b_C > 0$. Let $v \in V(C) \setminus S$. Note that $p < |V(C)|$, thus the vertex v exists. For a vertex v it holds that $a_i^S(v) < p + |N(C)|$ for every round i of the process. Thus, the vertex v is never activated because $p + |N(C)| < b_C + |N(C)| = f'(C)$ and S is not a target set. On the other hand, if we put b_C vertices from each twin clique C into a set S', then the set $S' \cup T$ is a target set because every vertex not in S' is activated in the first round. □

Structure of the Solution. Let (G, f, b) be an instance of Uniform Target Set Selection with $\mathrm{tc}(G) = t$. By Observation 28, if $b < \sum b_C$, then we automatically reject. On the other hand, if $b \geq t + \sum b_C$, then we automatically accept. Let $w = b - \sum b_C$. Thus, to find a target set of size b we need to select w *excess* vertices from twin cliques and twin cover. We will show there are at most $g(t)$ interesting choices how to select these w excess vertices for some computable function g. Since we can check if a given set $S \subseteq V(G)$ is a target set in polynomial time, we will have an FPT-algorithm for Uniform Target Set Selection.

We start with an easy preprocessing. Let C be a twin clique and $b_C > 0$. We select b_C vertices $V' \subseteq V(C)$ and remove them from the graph G. We also decrease the threshold value by b_C of every vertex which was adjacent to V' (recall that vertices in V' has the same neighborhood type, thus any vertex adjacent to some vertex in V' is adjacent to all vertices in V'). Formally, we get new instance $(G_1, f_1, b - b_C)$ where G_1 is G without vertices V' and

$$f_1(v) = \begin{cases} f(v) & v \notin N(V') \\ f(v) - b_C & v \in N(V'). \end{cases}$$

It is easy to see that instances (G, f, b) and $(G_1, f_1, b - b_C)$ are equivalent, because any target set of G need at least b_C vertices in the twin clique C due to Observation 28. Note that the function f_1 is uniform as well. From now we suppose that the instance (G, f, b) is already preprocessed. Thus, for every clique C holds that $b_C = 0$ and $f'(C) \leq N(C) \leq t$.

We say that a twin clique C is of *type* (Q, r) for $Q \subseteq T, r \leq t$ if $Q = N(C)$ and $f'(C) = r$. Two twin cliques C and D are of the same type if $N(C) = N(D)$ and $f'(C) = f'(D)$. Note that there are at most $(t + 1) \cdot 2^t$ distinct types of twin cliques.

We start to create a possible target set S of size b. We add w_1 (for some $w_1 \leq w$) vertices from the twin cover to S (at most 2^t choices). Now we need to select $w_2 = w - w_1$ excess vertices from twin cliques to S.

The number of twin cliques of one type is big. Thus, for twin cliques we need some more clever way than try all possibilities. The intuition is that if we want to select some excess vertices from a clique of type (Q, r) it is "better" choice to select the vertices from large cliques of type (Q, r). We assign each type (Q, r) a number $w_{(Q,r)}$ how many excess vertices would be in twin cliques of type (Q, r). We prove that it suffices distribute $w_{(Q,r)}$ excess vertices among $w_{(Q,r)}$ largest twin cliques of type (Q, r).

Definition 19. *Let $C_1, \ldots, C_p$ be all twin cliques of type (Q, r) ordered descendently by size, i.e., for all $i < p$ holds that $|V(C_i)| \geq |V(C_{i+1})|$. We say that a target set has a hole (C_i, C_j) for $j > i$ if $|S \cap V(C_i)| = 0$ and $|S \cap V(C_j)| \geq 1$. A target set is* (Q, r)-leaky *if it has a hole and it is* (Q, r)-compact *otherwise.*

Our goal is to prove that if there is a target set S which is (Q, r)-leaky, then there is also a target set R which is (Q, r)-compact and $|R| \leq |S|$.

Delayed Activation Process. For a better analyzing the activation process we introduce a little bit more general notion. Besides the threshold function we also have a delay function $d : V(G) \to \mathbb{N}$. Let $S \subseteq V(G)$. The *delayed activation*

process arising from S is $S = S_0, S_1, \dots$ where

$$S_{i+1} = S_i \cup \{v \in V : |N(v) \cap S_i| \geq f(v), i \geq d(v)\}.$$

Thus, a vertex v cannot be activated before round $d(v)$. We say the set S is a *delayed target set* if there is some j such that $S_j = V(G)$. Note that a target set is a special case of delayed target set for a delay function $d(v) = 0$ for all vertices $v \in V(G)$.

Observation 29. *Let S be a delayed target set for a graph G, a threshold function f and a delay function d. Then, S is a target set for graph G and threshold function f.*

Proof. The delayed condition of vertex activation is more restrictive than the condition for the standard activation process. Suppose vertex v is activated in a round i during a delayed activation process. Then, the vertex v is certainly activated during the round i of the standard activation process at the latest. □

Lemma 30. *Suppose there is a target set S for a graph G with a threshold function f and S is (Q, r)-leaky for some twin clique type (Q, r). Then, there is a delayed target set R such that:*

1. *It holds that $|R| \leq |S|$.*
2. *The sets R and S differ only at twin cliques of type (Q, r).*
3. *The set R is (Q, r)-compact.*

Proof. Let T be a twin cover of the input graph. Let $C_1, \dots, C_p$ be order of all twin cliques of type (Q, r) as in Definition 19 and the target set S has a hole (C_i, C_j). Let $\mathcal{S} = (S = S_0, S_1, \dots)$ be an activation process arising from S and the clique C_j is activated in a round j' and the clique C_i in a round i' of the process $\mathcal{S}$.

First, suppose that $i' \leq j'$. Since C_i is activated in the round i' and $S \cap C_i = \emptyset$, in the round i' there is $f'(C_i)$ active vertices in $Q \subseteq T$. Since the cliques C_i and C_j have the same type, the clique C_j is activated in the round i' as well. Moreover, C_j is activated in the round i' even if we remove vertices in $S \cap C_j$ from S, because $f'(C_i) = f'(C_j)$ and the cliques C_i and C_j have the same neighborhood in the twin cover T. We concluded that $R = S \setminus V(C_j)$ is a target set as well.

Now suppose that $i' > j'$. We create a delayed target set R by removing vertices from C_j and adding the same number of vertices from C_i. Formally,

$$R = \Big(S \setminus V(C_j)\Big) \cup X$$

where $X \subseteq V(C_i)$ and $|X| = |S \cap V(C_j)|$. Let $Y \subseteq V(C_i) \setminus X$ such that $|Y| = |V(C_j) \setminus S|$. We set a delay function d as follows

$$d(v) = \begin{cases} j' & v \in Y \\ i' & v \in \Big(V(C_i) \setminus (X \cup Y)\Big) \cup V(C_j) \\ 0 & \text{otherwise} \end{cases}$$

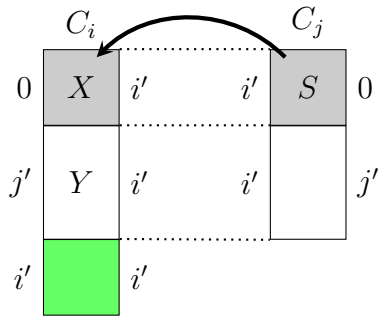

Figure 3.2: A depiction how we create the set R from the set S. There are rounds when the parts of cliques are activated during the process $\mathcal{S}$ or $\mathcal{R}$ on the right, or left respectively.

See Figure 3.2 for better understanding how we create the set R from the set S and how we set the delay. Let $\mathcal{R} = (R = R_0, R_1, \dots)$ be an activation process arising from the set R. The meaning of the delay is that part of the clique C_i (the sets X and Y) will behave in the process $\mathcal{R}$ same as the clique C_j in the process $\mathcal{S}$. We will prove the following claim.

Claim 1.

1. For every k holds that $S_k \cap T \subseteq R_k \cap T$ and $S_k \cap V(C) \subseteq R_k \cap V(C)$ for every twin clique C different from C_i and C_j.

2. For every vertex v in the cliques C_i and C_j holds that $v \in R_{d(v)}$.

Immediate corollary of Claim 1 is that R is a delayed target set. We prove Claim 1 by induction on the number of round k. It is clear it holds for $k = 0$ because of the construction of set R_0.

Now suppose it holds for $k \geq 0$ and we will prove it for $k + 1$. Recall that for every vertex v not in the cliques C_i and C_j holds that $d(v) = 0$. Let $v \in (S_{k+1} \setminus S_k) \cap T$. Let $\mathcal{C}_v$ be twin cliques C such that $v \in N(C)$ and $T_v = T \cap N(v)$, i.e., the neighbors of v in T. Then,

$$a^S_{k+1}(v) = \sum_{C \in \mathcal{C}_v} |S_k \cap V(C)| + |S_k \cap T_v| \geq f(v).$$

Suppose $v \notin Q$. Then by induction hypothesis (Item 1 in the claim), it holds that

$$\begin{aligned} f(v) \leq a^S_{k+1}(v) &= \sum_{C \in \mathcal{C}_v} |S_k \cap V(C)| + |S_k \cap T_v| \\ &\leq \sum_{C \in \mathcal{C}_v} |R_k \cap V(C)| + |R_k \cap T_v| = a^R_{k+1}(v). \end{aligned}$$

Therefore, $v \in R_{k+1} \cap T$.

Now suppose $v \in Q$. We distinguish 3 cases: $k + 1 \leq j'$, $j' < k + 1 \leq i'$ and $i' < k + 1$. Suppose $k + 1 \leq j'$. Thus, the vertex v is activated at most in the same round as the clique C_j in the process $\mathcal{S}$. Note that in the process $\mathcal{R}$ the only active neighbors of the vertex v in the clique C_i and C_j are those vertices in the set X (which has the same size as $S_0 \cap V(C_j)$. Then,

$$\begin{aligned} f(v) \leq a^S_{k+1}(v) &= \sum_{C \in \mathcal{C}_v} |S_k \cap V(C)| + |S_k \cap T_v| \\ &= \sum_{C \in \mathcal{C}_v \setminus \{C_i, C_j\}} |S_k \cap V(C)| + |S_0 \cap C_j| + |S_k \cap T_v| \\ &\leq \sum_{C \in \mathcal{C}_v \setminus \{C_i, C_j\}} |R_k \cap V(C)| + |X| + |R_k \cap T_v| = a^R_{k+1}(v). \end{aligned}$$

Thus, $v \in R_{k+1}$. The other cases are similar, now suppose that $j' < k+1 \leq i'$, i.e. the vertex v is activated in the process $\mathcal{S}$ after the clique C_j but before the clique C_i. In the process $\mathcal{R}$ the vertices in X and Y are the only active vertices in the cliques C_i and C_j (recall that $|X| + |Y| = |V(C_j)|$.

$$\begin{aligned} f(v) \leq a^S_{k+1}(v) &= \sum_{C \in \mathcal{C}_v} |S_k \cap V(C)| + |S_k \cap T_v| \\ &= \sum_{C \in \mathcal{C}_v \setminus \{C_i, C_j\}} |S_k \cap V(C)| + |C_j| + |S_k \cap T_v| \\ &\leq \sum_{C \in \mathcal{C}_v \setminus \{C_i, C_j\}} |R_k \cap V(C)| + |X| + |Y| + |R_k \cap T_v| = a^R_{k+1}(v). \end{aligned}$$

The last case $i' < k+1$ is when the vertex v is activated after the activation of cliques C_i and C_j in both processes.

$$\begin{aligned} f(v) \leq a^S_{k+1}(v) &= \sum_{C \in \mathcal{C}_v} |S_k \cap V(C)| + |S_k \cap T_v| \\ &= \sum_{C \in \mathcal{C}_v \setminus \{C_i, C_j\}} |S_k \cap V(C)| + |C_j| + |C_i| + |S_k \cap T_v| \\ &\leq \sum_{C \in \mathcal{C}_v \setminus \{C_i, C_j\}} |R_k \cap V(C)| + |C_j| + |C_i| + |R_k \cap T_v| = a^R_{k+1}(v). \end{aligned}$$

In all three cases the vertex v has more than $f(v)$ active neighbors in the round k of process $\mathcal{R}$. Therefore, $v \in R_{k+1}$.

Now let $v \in (S_{k+1} \setminus S_k) \cap V(C)$ for some twin clique $C \neq C_i, C_j$. Let (Q', r') be a type of C. Thus by induction hypothesis,

$$f(v) \leq a^S_{k+1}(v) = |S_0 \cap V(C)| + |S_k \cap Q'| \leq |R_0 \cap V(C)| + |R_k \cap Q'| = a^R_{k+1}(v).$$

Therefore, $v \in R_{k+1}$.

It remains to prove the claim about cliques C_i and C_j. It is clear that for vertex $v \in X$ holds that $v \in R_0$.

Let $v \in V(C_j)$, thus $d(v) = i'$. In the process $\mathcal{S}$ the clique C_i is activated in the round i', thus $f'(C_i) \leq |S_{i'-1} \cap Q|$. The cliques C_i and C_j have the same type, thus $f(v) = f'(C_i)$. Therefore by induction hypothesis,

$$f(v) \leq |S_{i'-1} \cap Q| \leq |R_{i'-1} \cap Q| = a^R_{i'}(v)$$

and the clique C_j is activated in the round i' of the process $\mathcal{R}$ because of the delay.

The proof for vertex $u \in V(C_i) \setminus R_0$ is similar. Again we use that $f(u) = f'(C_j)$. Let $u \in Y$, i.e. $d(u) = j'$. If $V(C_j) \setminus S = \emptyset$, then the set Y is empty as well and there is nothing to prove for this case. The clique C_j was activated in the round j' of the process $\mathcal{S}$. Thus,

$$f(u) = f'(C_j) \leq |S_0 \cap V(C_j)| + |S_{j'-1} \cap Q| \leq |R_0 \cap V(C_i)| + |R_{j'-1} \cap Q| = a^R_{j'}(u).$$

The vertex u is activated in the round j' of the process $\mathcal{R}$. Let $u' \in V(C_i) \setminus (X \cup Y)$, i.e., $d(u') = i'$. The clique C_i (included the vertex u') was activated in the round i' of the process $\mathcal{S}$. Thus,

$$f(u') \leq a^S_{i'} = |S_{i'-1} \cap Q| \leq |R_{i'-1} \cap Q| = a^R_{i'}(u').$$

Thus, vertex u is activated in the round $d(u') = i'$, which ends the proof of Claim 1.

We create a delayed target set R such that $|R| \leq |S|$ and the set R has one less holes than the set S. Therefore, if we repeat this procedure we eventually get (Q, r)-compact target set. The property of R_0 (Items 1–3 in the lemma) are clear from the construction. □

By Lemma 30 and Observation 29 we know that if there is a (Q, r)-leaky target set S then there is a (Q, r)-compact set R. Moreover, the set S and R differs only at twin cliques of type (Q, r). Thus, if we repeat the procedure for all types we get a target set without any hole. To summarize how to distribute w excess vertices:

1. Pick w_1 vertices from the twin cover T, in total 2^t choices.
2. Distribute $w_2 = w - w_1$ excess vertices among $t \cdot 2^t$ types of twin cliques, in total $(t \cdot 2^t)^t = 2^{\mathcal{O}(t^2)}$ choices.
3. Distribute $w_{(Q,r)}$ excess vertices among $w_{(Q,r)}$ largest big cliques of type Q, in total t^t choices.

Thus, we create $2^{\mathcal{O}(t^2)}$ candidates for a target set. For each candidate we test whether it is a target set or not. If any candidate is a target set, then we find a target set of size b. If no candidate is a target set, then by argumentation above we know the graph G has no target set of size b. This finishes the proof of Theorem 24.

3.2.2 Neighborhood diversity

In this section we prove the Uniform Target Set Selection problem admits an FPT algorithm on graphs of bounded neighborhood diversity. We again use Lemma 27. Note that, in each round of the activation process at least one type has to be activated. This implies that in this setting there are at most $\mathrm{nd}(G)$ rounds of the activation process. We use this fact to model the whole activation process as an integer linear program which is then solved using Lenstra's celebrated result:

Proposition 31 ([58, 37])**.** *Let p be the number of integral variables in a mixed integer linear program and let L be the number of bits needed to encode the program. Then it is possible to find an optimal solution in time $\mathcal{O}(p^{2.5p}\,\mathrm{poly}(L))$ and a space polynomial in L.*

Let C be a type and n_C be the number of vertices in C. Since we know when C is activated, we know how many active vertices are in C in each round. There are x_C vertices before the activation of C and n_C after the activation. To formulate the integer linear program we denote the set of types by $\mathcal{T}$ and we write $D \in N(C)$ if the two corresponding vertices in the type graph T_G are joined by an edge.

ILP Formulation.

$$\begin{aligned}
&\text{minimize} && \sum_{C \in \mathcal{T}} x_C && \\
&\text{subject to} && f'(C) \leq \sum_{D \prec C, D \in N(C)} n_D + \sum_{D \succ C, D \in N(C)} x_C && \forall C \in \mathcal{T} \\
&\text{where} && 0 \leq x_C \leq n_C && \forall C \in \mathcal{T}
\end{aligned}$$

As there are at most $t!$ orders of the set $\mathcal{T}$, this implies that the Uniform Target Set Selection problem can be solved in time $t!t^{\mathcal{O}(t)}\,\mathrm{poly}(n) = t^{\mathcal{O}(t)}\,\mathrm{poly}(n)$. Thus, we have proven Theorem 22.

3.3 Hardness Reductions

In this section we prove that Target Set Selection is W[1]-hard on graphs of bounded neighborhood diversity and a general threshold function. We use an FPT-reduction from k-Multicolored Clique.

Let G be an input of k-Multicolored Clique. We refer to a set V_c as to a *color class* of G and to a set E_{cd} as to edges between color classes V_c and V_d. The problem is W[1]-hard [21] even if every color class V_c has the same size and the number of edges between every V_c and V_d is the same. For easier notation during the reduction, we denote the size of arbitrary color class V_c by $n+1$ and the size of arbitrary set E_{cd} by $m+1$. We describe how to create from the graph G an instance $(G', f\colon V \to \mathbb{N}, b)$ of Target Set Selection such that:

1. The reduction runs in time $\mathrm{poly}(|G|)$.
2. The graph G has a clique of size k if and only if the graph G' has a target set of size b.
3. The neighborhood diversity of G is $\mathcal{O}(k^2)$. Moreover, all types of G' are independent set.

In the k-Multicolored Clique problem we need to select exactly one vertex from each color class V_c and exactly one edge from each set E_{cd}. Moreover, we have to make certain that if $\{u, v\} \in E_{cd}$ is a selected edge, then $u \in V_c$ and $v \in V_d$ are selected vertices. As the proof is quite long and technical we overview main ideas contained in the proof here.

Overview of Proof of Theorem 23. We present a way of encoding a vertex v in a color class V_c of graph G by two numbers v-pos and v-neg with v-pos + v-neg $= n$. We proceed with encoding of edges by multiples of sufficiently large number q. This we do in such a way that sum of the encoding of a vertex and an incident edge is unique. Finally, we add an incidence check that has a vertex for each possible incidence between a vertex and an edge. Thus, we do this in both -pos and -neg parts. However, in this encoding all edges preceding the selected edge have their threshold also fulfilled – this happens in the -pos part, while in the -neg part all edges following the selected edge have their threshold fulfilled. The core of the proof relies on a fact that the threshold for the selected edge is fulfilled in both (-pos and -neg) parts if and only if the selected vertex is incident with it. It follows that there are only two possibilities – either $m+1$ or $m+2$ thresholds are fulfilled. Thus, we can test the incidence using threshold.

3.3.1 Proof of Theorem 23

Selection Gadget. First, we describe gadgets of the graph G' for selecting vertices and edges of the graph G. For an overall figure of the gadget please refer

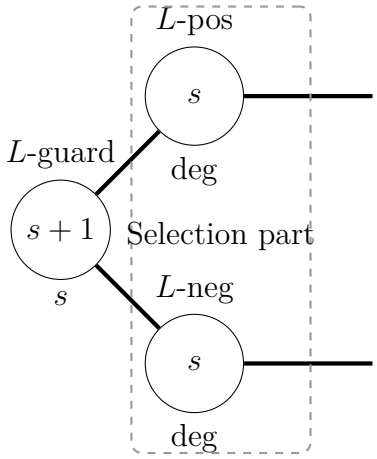

Figure 3.3: An overview of the selection gadget $L(s)$. Numbers in circles denote numbers of vertices in each type and numbers under circles denote thresholds of vertices in each type.

to Figure 3.3. The gadget $L(s)$ is formed by two types L-neg and L-pos of equal size s (the number s will be determined later); we refer to these two types as the *selection part*. For a vertex v in the selection part we set the value $f(v)$ of the threshold to the degree of v. It means that if some vertex v from the selection part is not selected into the target set then all neighbors of v have to be active before the vertex v can be activated by the activation process. The selection gadget L is connected to the rest of the graph using only vertices from the selection part.

The last part of the gadget L is formed by type L-guard of $s + 1$ vertices connected to both types in the selection parts. For each vertex v in L-guard type we set $f(v) = s$.

Lemma 32. *Suppose there is a selection gadget $L(s)$ in the input graph G' of* Target Set Selection. *We claim that exactly s vertices of the gadget L are needed to be selected in the target set S to activate the vertices in the L-guard type. Moreover, these s vertices have to be selected from the selection part of L.*

Proof. Let $S' = V(L) \cap S$, i.e., vertices of the target set S in the gadget L. First, suppose $|S'| < s$ or $|S'| = s$ and some vertex u of L-guard is in S'. Since there are $s + 1$ vertices in L-guard, there is a vertex v in L-guard type in the gadget L such that $v \notin S'$. Let V^p be vertices of the selection part of L. The vertex v has neighbors only in V^p and the threshold of v is s. Note that $|V^p \cap S'| < s$. Thus, at least one vertex $w \in V^p \setminus S'$ need to be activated during the activation process before the vertex v is activated. However, $f(w) = \deg(w)$. Therefore, the vertex w have to be activated after the vertex v is activated. That is a contradiction and $|V^p \cap S'| \geq s$ must hold. When S' contains s vertices from the selection part of L, then it is easy to see that the all vertices in L-guard type are activated in the first round of the process. □

Numeration of Vertices and Edges. Now, we describe how we use the selection gadget. Let $V_c = \{v_0, \ldots, v_n\}$. By Lemma 32, we can encode selection of vertices and edges of the graph G to the multicolor clique. For every color class V_c we create a selection gadget $L_c = L(n)$. We select a vertex $v_i \in V_c$ to the multicolor clique if i vertices in the L_c-pos type and $n - i$ vertices in the L_c-neg type of the gadget L_c are selected into the target set.

The selection of edges is similar, however complicated. Let $q \in \mathbb{N}$ and $E_{cd} = \{e_0, \ldots, e_m\}$. For every set E_{cd} we create a selection gadget $L_{cd}(qm)$. We select

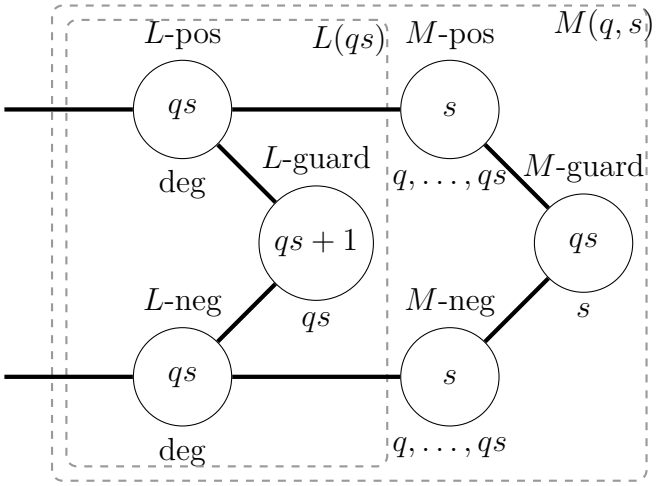

Figure 3.4: An overview of the multiple gadget $M(q, s)$.

an edge $e_j \in E_{cd}$ to the multicolor clique if qj vertices in the L_{cd}-pos type of the gadget L_{cd} are selected into the target set (and $q(m-j)$ vertices in the L_{cd}-neg are selected into the target set). Suppose s vertices in the L_{cd}-pos type are selected into the target set. If s is not divisible by q, then it is an invalid selection. We introduce a new gadget such that s has to be divisible by q.

Multiple Gadget. A multiple gadget $M(q, s)$ contains a selection gadget $L(qs)$ and 3 other types M-pos, M-neg of s vertices and M-guard of qs vertices. The type M-pos is connected to the type L-pos and the type M-neg is connected to the type L-neg. The type M-guard is connected to the types M-pos and M-neg. Still, the rest of graph G' is connected only to types L-pos and L-neg. Let $\{u_1, \ldots, u_s\}$ and $\{w_1, \ldots, w_s\}$ be vertices in M-pos type and M-neg type, respectively. We set thresholds $f(u_i) = f(w_i) = qi$. For each vertex v in M-guard we set $f(v) = s$. For an example of multiple gadget see Figure 3.4.

Lemma 33. *Suppose there is a multiple gadget $M(q, s)$ in the input graph G' of* Target Set Selection. *Let L be a selection gadget in M. We claim that exactly qs vertices of the gadget L are needed to be selected in the target set S to activate the types L-guard, M-pos, M-neg and M-guard. Moreover, these qs vertices have to be selected from the selection part of L and the numbers of vertices selected in L-pos and L-neg types are divisible by q.*

Proof. By Lemma 32, we know that qs selected vertices in the types L-pos and L-neg are needed to activate L-guard type. Suppose there is z vertices in the L-pos type selected into a target set and z is not divisible by q. It follows that there are $qs - z$ selected vertices in L-neg. Thus, $z = qa + r, r \neq 0$ and $qs - z = q(s-a) - r$. Let $\{u_1, \ldots, u_s\}$, $\{w_1, \ldots, w_s\}$ be vertices in the M-pos type, in the M-neg type respectively. Recall that $f(u_i) = f(w_i) = qi$. Thus, vertices $u_1, \ldots, u_a$ and $w_1, \ldots, w_{s-a-1}$ are activated in the first round of the activation process.

We claim that no other vertices in the gadget M would be activated during the process. Vertices in M-guard type have only $s-1$ activated vertices among their neighbors and have thresholds s. Vertices in L-pos and L-neg have thresholds their degrees. Thus, they have to be activated after all vertices in M-pos and M-neg are activated. Vertices $u_{a+1}, \ldots, u_s$ in the M-pos type and $w_{s-a}, \ldots, w_s$ in the M-neg type cannot be activated unless some of their neighbors are activated.

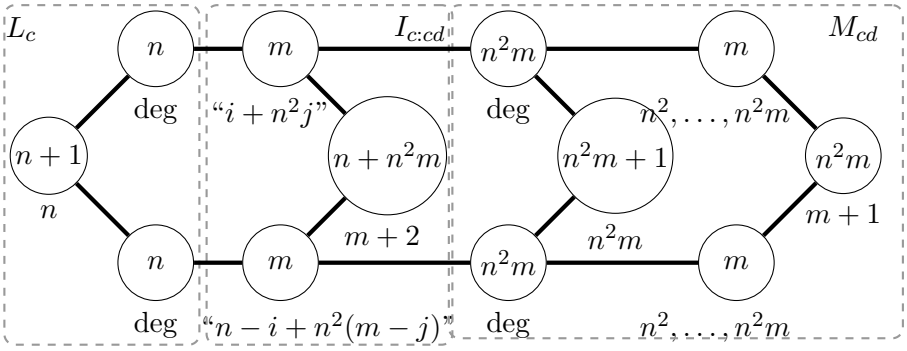

Figure 3.5: An overview of the reduction. The number inside a type is the number of vertices of the type. There is a threshold of vertices beneath each type.

Now suppose that $r = 0$, i.e., $z = qa$ and $qs - z = q(s - a)$. Vertices $u_1, \ldots, u_a$ and $w_1, \ldots, w_{s-a}$ are activated in the first round. All vertices in the M-guard type are activated in the second round because they have s activated vertices among their neighbors. Recall that the maximum threshold in the M-pos and the M-neg type is qs. Since there are qs vertices in M-guard, every vertex in the types M-pos and M-neg has at least qs activated vertices among its neighbors. Therefore, the rest of vertices in the types M-pos and M-neg are activated in the third round. □

Incidence Gadget. So far we described how we encode in graph G' selecting vertices and edges to multicolor clique. It remains to describe how we encode the correct selection, i.e., if $v \in V_c$ and $e \in E_{cd}$ are selected vertex and edge to multicolor clique, then $v \in e$. We create $L_c(n)$ selection gadget for a color class V_c. We set the number q to n^2 and create a multiple gadget $M_{cd}(n^2, m)$ (with selection gadget L_{cd}) for a set E_{cd}. We join gadgets L_c and M_{cd} through an incidence gadget $I_{c:cd}$. See Figure 3.5, for better understanding how the incidence gadget is connected to the selection and multiple gadgets. The incidence gadget $I_{c:cd}$ has three types $I_{c:cd}$-pos and $I_{c:cd}$-neg of $m + 1$ vertices and $I_{c:cd}$-guard of $n + n^2m$ vertices. We connect the $I_{c:cd}$-guard type to the types $I_{c:cd}$-pos and $I_{c:cd}$-neg. Furthermore, we connect the type $I_{c:cd}$-pos to the types L_c-pos and L_{cd}-pos. Similarly, we connect the type $I_{c:cd}$-neg to the types L_c-neg and L_{cd}-neg.

We set thresholds of all vertices in the $I_{c:cd}$-guard type to $m + 2$. Recall there are $m + 1$ edges in the set E_{cd}. Thus, we can associate edges in E_{cd} with vertices in $I_{c:cd}$-pos ($I_{c:cd}$-neg respectively) one-to-one. I.e., $V(I_{c:cd}\text{-pos}) = \{u_\ell : e_\ell \in E_{cd}\}$ and $V(I_{c:cd}\text{-neg}) = \{w_\ell : e_\ell \in E_{cd}\}$. Let $v_i \in V_c, e_j \in E_{cd}$ and $v_i \in e_j$. Recall that selecting v_i and e_j into a multicolor clique is encoded as selecting i vertices in L_c-pos type and $n^2 j$ vertices in L_{cd}-pos type into a target set. We set threshold of u_j to $i + n^2 j$ and threshold of w_j to the "opposite" value $n - i + n^2(m - j)$.

Since we set the coefficient q to n^2, for each edge $e_j \in E_{cd}$ and each vertex $v_i \in V_c$ the sum $i + n^2 j$ is unique. Thus, every vertex in $I_{c:cd}$-pos ($I_{c:cd}$-neg) has a unique threshold. We will use this number to check the incidence.

Reduction Correctness. We described how from the graph G with k color classes (instance of k-Multicolored Clique) we create the graph G' with the threshold function f (input for Target Set Selection):

1. For every color class V_c we create a selection gadget L_c.
2. For every edge set E_{cd} we create a multiple gadget M_{cd}.
3. We join the gadgets L_c and M_{cd} by an incidence gadget $I_{c:cd}$ (gadgets L_d and M_{cd} are joint by a gadget $I_{d:cd}$).

It is easy to see the following observations by constructions of G'.

Observation 34. *The graph G' has polynomial size in the size of the graph G.*

Observation 35. *Neighborhood diversity of the graph G' is $\mathcal{O}(k^2)$.*

To finish the instance of Target Set Selection we set budget for target set $b = kn + \binom{k}{2}n^2m$.

Theorem 36. *If the graph G contains a clique of size k, then G' with the threshold function f contains a target set of size b.*

Proof. Let K be a k-clique in the graph G. We construct a set $S \subseteq V(G')$. Let $v_i \in V(K) \cap V_c$. We add i vertices in the L_c-pos type and $n-i$ in the L_c-neg type into the set S. Let $e_j \in E(K) \cap E_{cd}$. For the set E_{cd} we have a multiple gadget M_{cd} and there is a selection gadget L_{cd} inside M_{cd}. We add n^2j vertices in the L_{cd}-pos type and $n^2(m-j)$ vertices in the L_{cd}-neg into the set S. We have n vertices in S for every color class V_c and n^2m vertices in S for every edge set E_{cd}. Thus,

$$|S| = kn + \binom{k}{2}n^2m = b.$$

We claim that the set S is a target set. We analyze the selection gadget L_c, the multiple gadget M_{cd} (with the L_{cd} selection gadget) and the incidence gadget $I_{c:cd}$. All vertices in the types L_c-guard and L_{cd}-guard are activated in the first round (see proof of Lemma 32). All vertices in the types M_{cd}-neg, M_{cd}-pos and M_{cd}-guard are activated during the first three rounds – for details see proof of Lemma 33.

Recall $V(I_{c:cd}\text{-pos}) = \{u_\ell : e_\ell \in E_{cd}\}$ and $V(I_{c:cd}\text{-neg}) = \{w_\ell : e_\ell \in E_{cd}\}$. The threshold of $u_\ell \in V(I_{c:cd}\text{-pos})$ is $n^2\ell + \ell'$ for some $\ell' \in \{0, \ldots, n\}$. There are $n^2j + i$ vertices activated in the types L_{cd}-pos and L_c-pos. Vertices $u_0, \ldots, u_{j-1}$ are activated in the first round because their thresholds are strictly smaller than n^2j. The threshold of u_j is n^2j+i because this vertex corresponds to the incidence $v_i \in e_j$. Thus, the vertex u_j is activated in the first round as well. Vertices $u_{j+1}, \ldots, u_m$ have thresholds bigger than $n^2(j+1)$ and cannot be activated in the first round. By the same analysis we get that vertices $w_j, \ldots, w_m$ in the $I_{c:cd}$-neg type are activated in the first round.

In the first round there are $m+2$ activated vertices in the types $I_{c:cd}$-pos and $I_{c:cd}$-neg. All vertices in the $I_{c:cd}$-guard type are activated in the second round because they have threshold $m+2$. The maximum threshold in the $I_{c:cd}$-pos ($I_{c:cd}$-neg) type is $n + n^2m$. Thus, the rest of vertices in the types $I_{c:cd}$-pos and $I_{c:cd}$-neg are activated in the third round because they have $n + n^2m$ activate neighbors in the $I_{c:cd}$-guard type.

All vertices outside the types L_c-pos, L_c-neg, L_{cd}-pos and L_{cd}-neg are activated during the first three rounds. Let U be a set of vertices which are not activated

during the first three rounds. Note that U is an independent set and for every $u \in U$ holds that $f(u) = \deg(u)$. Therefore, vertices in U are activated in the fourth round. □

Theorem 37. *If the graph G' with the threshold function f contains a target set of size b, then G contains a clique of size k.*

Proof. Let S be a target set of the graph G of size b. There are k selection gadgets $L(n)$ in G'. By Lemma 32, the set S has to contain at least n vertices in the selection part of every gadget $L(n)$. There are also $\binom{k}{2}$ selection gadgets $L(n^2m)$ in multiple gadgets in G'. By Lemma 33, the set S has to contain at least n^2m vertices in the selection part of every gadget $L(n^2m)$. Since $|S| = b = kn + \binom{k}{2}n^2m$, there is not any other vertex in S.

Now, for every V_c and E_{cd} we select a vertex (or an edge, respectively). We select a vertex $v_i \in V_c$ if $|V(L_c\text{-pos}) \cap S| = i$. We select an edge $e_j \in E_{cd}$ if $|V(L_{cd})\text{-pos} \cap S| = qj$. By Lemma 32 and 33, we know the selection is correct. We claim that if $v_i \in V_c$ is the selected vertex and $e_j \in E_{cd}$ is the selected edge, then $v_i \in e_j$.

For a contradiction suppose $v_i \notin e_j$. We analyze the incidence gadget $I_{c:cd}$. Let $V(I_{c:cd}\text{-pos}) = \{u_0, \ldots, u_m\}$ and $V(I_{c:cd}\text{-neg}) = \{w_0, \ldots, w_m\}$. Vertices in the type $I_{c:cd}$-pos have $i + n^2j$ active neighbors. Vertices in the type $I_{c:cd}$-neg have $n - i + n^2(m - j)$ active neighbors. As we say in the proof of Theorem 36, vertices $u_0, \ldots, u_{j-1}$ and $w_{j+1}, \ldots, u_m$ are activated in the first round and vertices $u_{j+1}, \ldots, u_m$ and $w_0, \ldots, w_{j-1}$ are not activated.

It remains to analyze the vertices u_j and w_j. Suppose u_j is activated in the first round. Thus, $f(u_j) = i' + n^2j < i + n^2j$. Note that $i' < i$ because we suppose $v_i \notin e_j$. For threshold of w_j holds

$$f(w_j) = n - i' + n^2(m - j) > n - i + n^2(m - j).$$

Since the vertex w_j has $n - i + n^2(m - j)$ activate neighbors, the vertex w_j cannot be activated in the first round. Thus, at least one of the vertices u_j, w_j is not activated in the first round.

Any vertex of the type $I_{c:cd}$-guard cannot be activated in the first round because they have threshold $m + 2$ and they have at most $m + 1$ activated neighbors. Vertices of the type $I_{c:cd}$-guard have to be activated after some other vertices in the types $I_{c:cd}$-pos or $I_{c:cd}$-neg are activated. However, there is not any new activate vertex in the neighborhood of the types $I_{c:cd}$-pos and $I_{c:cd}$-neg. The activation process does not activate all vertices in $V(G')$. Therefore, S is not a target set, which is a contradiction. □

Theorem 23 is a corollary of Theorem 36, 37 and Observation 34, 35.

3.3.2 Proof of Theorem 25

Overview of Proof of Theorem 25. In fact this can be seen as a clever twist of the ideas contained in the proof of Theorem 23. There are some nodes of the neighborhood diversity decomposition already operating in the majority mode – e.g. guard vertices – these we keep untouched. However, for vertices

with threshold set to theire degree one has to "double" the number of vertices in the neighborhood and make sure that no newly added vertex is activated before all with threshold deg. Finally, one has to deal with types having different thresholds for each of its vertices. Here we exploit the property of the previous proof – that these vertices naturally come in pairs and that it is possible to replace each of these vertices by a collection of cliques. This ensures that even if the neighborhood is the same some vertices get activated and some not.

Similarly to Proof of Theorem 23, we show a parameterized reduction from k-Multicolored Clique to Majority Target Set Selection parameterized by restricted modular width. Namely, for every instance $G = (V_1 \cup \cdots \cup V_k, E)$ of k-Multicolored Clique, we construct an instance (G', b) of Majority Target Set Selection such that

- G' has a target set of size b if and only if G has a clique of size k,
- G' can be constructed in time poly($|G|$), and
- the restricted modular width of G' is $\mathcal{O}(k^2)$.

As before, the graph G' consists of several different types of gadgets. We now focus on their description.

Throughout the description, we use the following terminology. The vertices that are adjacent to vertices outside of gadget are referred to as *interface vertices*, the other vertices of gadget are *internal vertices*. The vertices outside of gadget that are adjacent to its interface part are *neighboring vertices of the gadget*.

For the gadget to work properly, we need to know to how many neighboring vertices the gadget has.

Selection gadget $L(s, r)$. This gadget (together with an appropriate budget value) ensures that every target set encodes a number from 0 to s. The parameter r determines the number of neighboring vertices.

The gadget consists of following five types of vertices:

- two independent sets each of size s called L-pos and L-neg,
- one independent set of size $3s$ called L-guard,
- one independent set of size $r + 3s$ called L-doubling,
- one independent set of size s called L-end.

The L-guard is connected to L-pos and L-neg, L-doubling is connected to L-pos, L-neg, and L-end. The L-pos and L-neg form the interface part of the gadget. The gadget needs to be connected to the rest of the graph in such a way that there are exactly r neighboring vertices adjacent to L-pos and exactly r neighboring vertices adjacent to L-neg. See Figure 3.6 for an overview of the Selection gadget $L(s, r)$.

The next definition gives us a useful tool for showing a lower bound on number of vertices of gadget J that need to in every target set.

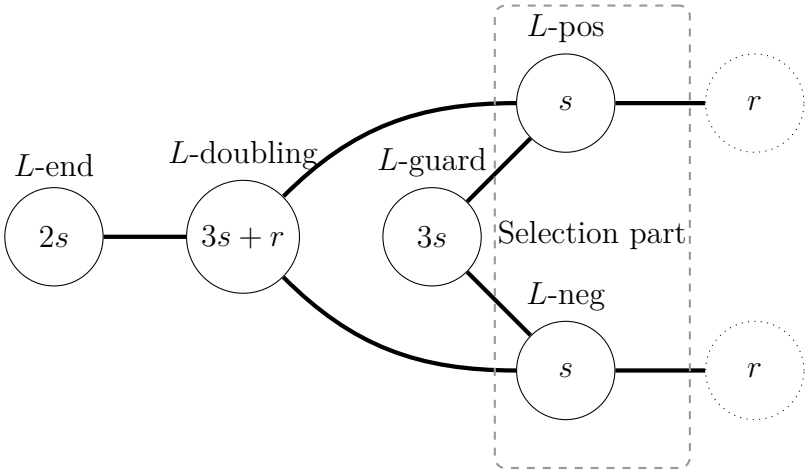

Figure 3.6: An overview of the Selection gadget for Majority Target Set Selection

Definition 20. *Let J be a gadget in G' and denote by $\overline{V_J}$ the set $V(G') \setminus V(J)$. The* deficit of a vertex $v \in V(J)$ with respect to J *is denoted by* $\mathrm{def}(v)$ *and is defined as*

$$\mathrm{def}(v) = \begin{cases} 0 & \text{if } |N(v) \cap \overline{V_J}| \geq \deg(v)/2 \\ \lceil \deg(v)/2 \rceil - |N(v) \cap \overline{V_J}| & \text{otherwise.} \end{cases}$$

We usually say just the deficit of the vertex *when the gadget is clear from the context.*

In other words, deficit tells us how many neighbors of v in J we need to activate, provided that every vertex outside of J is already activated.

For an internal vertex v, its deficit is exactly its threshold. For an interface vertex, the number is its threshold decreased by number of adjacent vertices outside J.

The following proposition gives us a lower bound on how many elements of J needs to be in a target set.

Proposition 38. *If all vertices in gadget J have deficit at least d and $S \subseteq V(G')$ is a set with $|S \cap V(J)| < d$ then S is not a target set.*

Proof. We show stronger statement that $S' = S \cup \overline{V_J}$ is not a target set. We see that every vertex $v \in V(J)$ has at most $|S \cap V(j)| + |N(v) \cap \overline{V_J}|$ neighbors in S'. By the assumption of the proposition, this is at most $d - 1 + |N(v) \cap \overline{V_J}|$. However, from the definition of deficit and the assumption that deficit is at least d we have $\lceil \deg(v)/2 \rceil \geq |N(v) \cap \overline{V_J}| + d$. This means that no vertex has enough activated neighbors and the activation process terminates immediately. □

Lemma 39. *Let $L = L(s, r)$ be a selection gadget in the graph G'. If $S \subseteq V(G')$ contains less than s vertices in $V(L)$ then S is not a target set.*

Proof. We examine the values of deficits for vertices in gadget L. The degrees of vertices in L-end, L-doubling, and L-guard are $3s + r$, $4s$, and $2s$ respectively and therefore their deficits are $(3s + r)/2$, $2s$, and s. The vertices in L-pos and L-neg

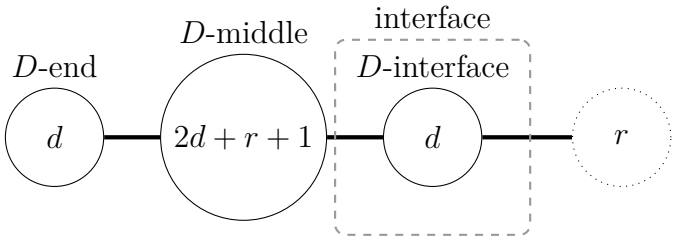

Figure 3.7: An overview of Threshold decrease gadget for Majority Target Set Selection

have degree $6s + 2r$ and so their threshold is $3s + r$. As they have r neighbors in $V(G') \setminus V(L)$, their deficit is $3s$.

We see that every vertex has deficit at least s. The lemma follows from Proposition 38. □

Lemma 40. *Let $L = L(s, r)$ be a selection gadget in a graph G' and let $S \subseteq V(G')$ be a target set that contains exactly s vertices from $V(L)$. Then the following holds.*

1. *The vertices of $V(L) \cap S$ are in the interface part of the gadget L.*
2. *Suppose that v is a vertex in interface part of L but not in S. The vertex v cannot activate before all neighboring vertices of L adjacent to v are activated.*

Proof. The only vertices with deficit s are in the L-guard part. Therefore, if S conatins less than s vertices from the neighborhood of L-guard, the activation process stops immediately. The neighborhood of L-guard is exactly the interface part of the gadget.

For the second part, we can assume $v \in V(L\text{-pos}) \setminus S$ since the gadget is symmetric. From part 1 we already know that the set S cannot contain any vertex from L-doubling. Since L-doubling contains exactly half vertices of $N(v)$, all other neighbors of $N(v)$ must be activated before v activates. □

Threshold decrease gadget $D(d, r)$. The purpose of threshold decrease gadget $D(d, r)$ is to ensure that there are d activated vertex in its interface part. This can be also viewed as effectively decreasing the threshold of neighboring vertices by $d/2$. The parameter r again determines the number of neighboring vertices.

The gadget consists of the following three parts:

- the independent set of size d called *D-interface*,
- the independent set of size $2d + r + 1$ called *D-middle*, and
- the independent set of size d called *D-end*.

The part D-middle is connected to D-interface and D-end. See Figure 3.7 for the figure of Threshold decrease gadget.

Lemma 41. *Let $D = D(d, r)$ be a threshold gadget in a graph G'. If $S \subseteq V(G)$ contains less than d vertices in $V(D)$ then S is not a target set.*

Proof. Let S_D denotes $S \cap V(D)$. Again we can assume that vertices in $V(G') \setminus V(D)$ are in S. The deficit of vertices in D-end part is $(2d + r + 1)/2 > d$, the deficit of vertices in D-middle part is d, and the deficit of vertices in D-interface part is $(2d + 2r + 1)/2 > d$. Therefore, even if the rest of the graph is activated and $|S_D| < d$, the activation process stops immediately. □

Lemma 42. *Let $D = D(d, r)$ be a threshold gadget in a graph G'. If S contains d vertices in $V(D)$ then those vertices must be in D-interface and D-end.*

Proof. The only vertices with deficit at most d are in part D-middle, therefore the d vertices of S must be in the neighborhood of D-middle, otherwise the process stops immediately. The neighborhood of D-middle is exactly D-interface adn D-end. □

Note that we can choose how we distribute d vertices in S between D-end and D-interface. The next lemma states that for determining existence of target set, it is enough to consider sets S that have d vertices in part D-interface.

Lemma 43. *Let $D = D(d, r)$ be a threshold gadget in a graph G'. If S is a target set, then the set $S' = \Big(S \setminus V(D\text{-}end)\Big) \cup V(D\text{-}interface)$ is target set as well. Moreover, $|S'| \leq |S|$.*

Proof. Denote by V_D^{int} the vertices of the internal part of the gadget D, by V_D^{if} the vertices of the interface part of the gadget D, and by $\overline{V_D}$ the set $V(G') \setminus V(D)$, i.e., vertices outside of gadget D.

Since the part V_D^{if} is already activated in S' and it separates V_D^{int} from $\overline{V_D}$, we can consider the activation process of $G'[V_D^{\text{int}} \cup V_D^{\text{if}}]$ and $G'[\overline{V_D} \cup V_D^{\text{if}}]$ separately.

The graph $G[V_D^{\text{int}} \cup V_D^{\text{if}}]$ clearly activates in two rounds. To analyze the graph $G[\overline{V_D} \cup V_D^{\text{if}}]$ first set $S'' = S \cap (\overline{V_D} \cup V_D^{\text{if}})$. Let $S'_0, S'_1, \ldots$ and $S''_0, S''_1, \ldots$ denote the activation process on $G'[\overline{V_D} \cup V_D^{\text{if}}]$ started by S' and S'' respectively.

We prove by induction on i that $S'_i \supseteq S''_i$. This easily implies that S' is a target set in $G'[\overline{V_D} \cup V_D^{\text{if}}]$. First consider the base case $i = 0$. If $v \in \overline{V_D}$, then $v \in S''_0$ if and only if $v \in S'_0$, since S'' and S' coincide on $\overline{V_D}$. If $v \in V_D^{\text{if}}$ then it is also in S'_0, since S' is a subset of V_D^{if}. Therefore $S'_0 \supseteq S''_0$.

Now suppose that $S'_i \supseteq S''_i$. If v is in V_D^{if}, then it is in S'_i because it already was in S'. Otherwise, all neighbors of v are in $\overline{V_D} \cup V_D^{\text{if}}$. But then if $v \in S''_{i+1}$, it has enough neighbors in S_i to activate. By induction hypothesis, it necessarily has enough neighbors in S'_i to activate in the process started by S' and hence it must lie in S'_{i+1}. This concludes the induction and proves that S' is a target set of $G'[\overline{V_D} \cup V_D^{\text{if}}]$. Since S' was a target set in both parts of G' separated by V_D^{if}, it is also a target set in G'.

The sets S and S' coincide on $\overline{D}$. From Lemma 41 we have that $|S \cap D| \geq d$. From definition of S' we see that $|S' \cap D| = d$. Putting these three observations together, we obtain that $|S'| \leq |S|$. □

The last lemma leads to following definition.

Definition 21. *The set S is called* canonical with respect to $D = D(d, r)$ *if all elements of $S \cap V(D)$ are in the interface part of D.*

By Lemma 43, to determine existence of target set in G', it is sufficient to consider only sets that are canonical with respect to every threshold decrease gadget in G'.

We now briefly discuss the total budget: it is set in such a way that it exactly covers all selection gadget and all threshold decrease gadgets. This means that if any element of S is outside of the interface part of Selection gadgets or outside of D-interface or D-end part of Threshold decrease gadgets, then S is not a target set. By Lemma 43, we know that we can ignore S intersecting any D-end part. This motivates the following definition.

Definition 22. *We say that a set $S \subseteq V(G')$ is* hopeful *if all its elements lie in the interface part of some selection gadget or in the interface part of some threshold decrease gadget.*

Note that every hopeful set is canonical with respect to every threshold decreae gadget.

When the budget is set as mentioned, it follows from Lemma 39, Lemma 41, and Lemma 43 that if we want to decide whether there exists a target set, it is sufficient to consider only hopeful sets.

Definition 23. *Let $S \subseteq V(G')$ be a hopeful set and $L = L(s, r)$ be a selection gadget in G'. If there are i elements of S in L-pos and $s - i$ elements of S in L-neg, we say that S* represents a number i in L.

Check gadget $C(Z, s)$. The gadget has two interface types C-pos and C-neg that can be connected to -pos and -neg parts of selection gadgets. The gadget is constructed in such a way that it activates if and only if the sum of the numbers encoded by connected selection gadgets is in Z.

We start with few auxiliary definitions. An *i-clique group* is the graph $(2(s - i) + 1) \times K_{2i+1}$ (i.e., disjoint union of $2(s - i) + 1$ cliques, each of size $2i + 1$). The *weight* of an i-clique group is the number of vertices of the group and is denoted by $w_s(i)$.

The graph C_Z is defined as the disjoint union of z-clique groups for each $z \in Z$. The graph $\overline{C}_Z$ is defined as the disjoint union of $s - z$ clique groups for each $z \in Z$. The z clique group in C_Z and the $s - z$ clique group in $\overline{C}_Z$ are referred to as *complementary clique groups*. Note that the clique group and its complementary clique group have the same weight.

The *weight of set $Z \subseteq [0, s]$* is defined as

$$w_s(Z) = \sum_{z \in Z} w_s(z).$$

The gadget $C(Z, s)$ consists of the following parts:

- interface parts C-pos and C-neg that will be formed by C_Z and $\overline{C}_Z$, respectively,
- independent set C-guard of size $2s$ that is connected to parts C-pos and C-neg,
- single vertex connected to C-guard, and

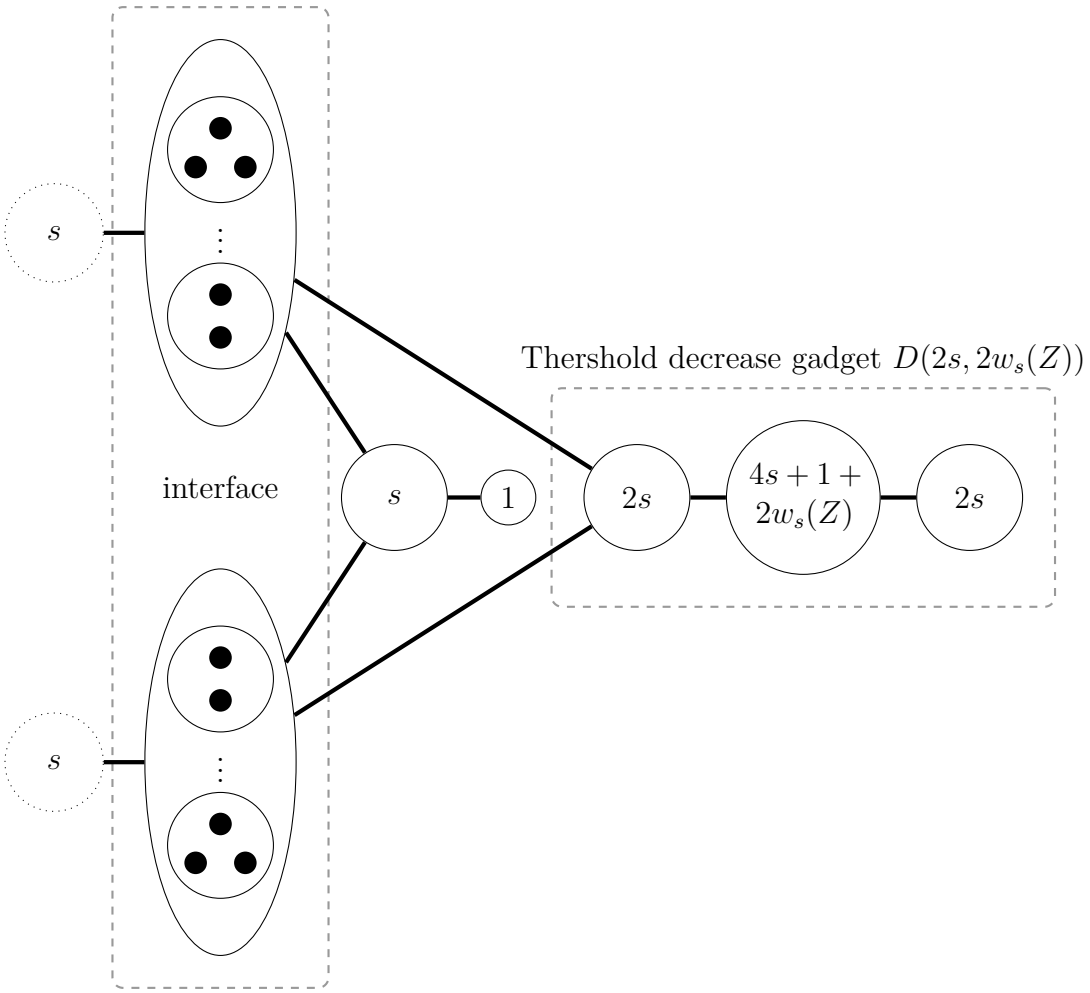

Figure 3.8: An overview of Check gadget $C(Z, s)$ for Majority Target Set Selection

- threshold decrease gadget $D_C = D(2s, 2w_s(Z))$ with its interface part connected to C-pos and C-neg.

See Figure 3.8 for overview of check gadget.

Lemma 44. *Let $C = C(Z, s)$ be a check gadget in the graph G' and let $L_1 = L(s_1, r_1), \ldots, L_h = L(s_h, r_h)$ be selection gadgets that are connected to C in such a way that each L_i-pos is connected to C-pos and each L_i-neg is connected to C-neg. We additionally require that $\sum s_i = s$ and no other gadget is connected to C. Furthermore, let S be a hopeful set that represents numbers $\ell_1, \ldots, \ell_h$ in gadgets $L_1, \ldots L_h$, respectively.*

The gadget C activates if and only if $\ell_1 + \cdots + \ell_h \in Z$.

Proof. First notice that due to Lemma 40 part 2, no vertex in L_i can activate before C-pos and C-neg are fully activated.

Let us set $\ell = \ell_1 + \cdots + \ell_h$.

Claim 1. The i-clique group K in C_Z activates in the first round if and only if $i \leq \ell$. The $(s - i)$-clique group $\overline{K}$ in $\overline{C}_Z$ activates in the first round if and only if $s - i \leq s - \ell$.

The degree of vertices in i-clique group K is $4s + 2i$; there are s neighbors in the adjacent selector gadgets, s neighbors in the C-guard part, $2s$ neighbors in the adjacent threshold decrease gadget, and $2i$ neighbors within the clique itself. Therefore the threshold of vertices in i-clique group is $2s + i$. We know that $2s$ vertices in the neighborhood of K are already activated due to the threshold decrease gadget D_C. This means that at least i another vertices need to be activated. Since S is hopeful, these vertices can be only in -pos parts of adjacent selector gadgets. From the assumption of the lemma, we know that there are ℓ such vertices, which gives us that the clique group activates if and only if $\ell \geq i$.

To prove the second part, it is enough to observe that exactly $s - \ell$ vertices are activated in -neg parts because S is hopeful. The rest can be proved by the same argument.

Claim 2. If a clique group K is not activated after first round, then its complementary clique group $\overline{K}$ is activated after first round.

Suppose that K is an i-clique group. This means that $\overline{K}$ is an $(s - i)$-clique group. If K was not activated in the first round we have $i > \ell$ by Claim 1. But them $s - i < s - \ell$ and again by Claim 1 the clique group $\overline{K}$ was activated in the first round.

Claim 3. At least half of vertices in $V(C\text{-pos}) \cup V(C\text{-neg})$ are activated in the first round.

The complementary clique groups K and $\overline{K}$ have the same number of vertices and at least of them is activated after first round by Claim 2.

Claim 4. If both i-clique group K in C_Z and its complementary clique group $\overline{K}$ are activated in the first round if and only if $\ell = i$ and therefore $\ell \in Z$.

From Claim 1 we see that both K and $\overline{K}$ activate if and only if $i \leq \ell$ and $s - i \leq s - \ell$, which together give $i = \ell$. We put an i-clique group into C_Z if and only if $i \in Z$, which finishes the proof of the claim.

Claim 5. If $\ell \notin Z$, the activation process in C stops after first round.

If $\ell \notin Z$, by Claim 4 we see that there is no pair of complementary clique groups K, $\overline{K}$ such that both of them are active. But then C-guard cannot activate, as less than half of its neighbors are active and the process stops.

Claim 6. If $\ell \in Z$, the whole gadget activates in three rounds.

We can only consider vertices outside of threshold decrease gadgets; it is straightforward to check that if S is hopeful, all threshold decrease gadgets activate in two rounds.

If $\ell \in Z$, then by Claim 4 a pair of complementary clique groups K, $\overline{K}$ that are both active after first round. Together with Claim 3, we see that strict majority of vertices in C-pos and C-neg are activated in first round. This causes the C-guard to activate in second round. In third round, the single vertex connected to C-guard clearly activates. Moreover, all remaining clique groups must activate, as they have at least $3s$ active neighbors in total ($2s$ from threshold decrease gadget, s from C-guard) and their threshold is $2s + i \leq 3s$. This finishes the activation process in C.

□

Multiple selection gadget $M(s, q, r)$ The last gadget used is the Multiple selection gadget.

It is a variant of selection gadget; similarly to selection gadget, every target set represents an integer between 0 and qs. However, multiple selection gadget additionally ensures that the number is a multiple of q. The last parameter r determines the number of neighboring vertices, as usual

The gadget consists of following parts:

- selection gadget $L(qs, r + w_{qs}(Q_{q,s}))$ and check gadget $C(Q_{q,s}, qs)$, where $Q_{q,s} = \{0, q, 2q, \ldots, sq\}$.
- the gadget encodes number from 0 to s as multiple of q
- r is the number of vertices connected to the Multiple gadget

Observe that the multiple selection gadget contains threshold decrease gadget $D(2qs, w_{qs}(Q_{q,s}))$ as a part of the check gadget $C(Q_{q,s}, qs)$.

Lemma 45. *Let $M = M(s, q, r)$ be a selection gadget in G'. If $S \subseteq V(G)$ satisfies $|S \cap V(M)| < 3qs$ then S is not a target set.*

Proof. If $|S \cap V(M)| < 3qs$ then at least one of the following necessarily happened.

- The selector gadget in M has less than qs vertices from S. Then by Lemma 39 we have that S is not a target set.
- The threshold decrease gadget in M has less than $2qs$ vertices from S. By Lemma 41 again S is not a target set.

□

Lemma 46. *Let $M = M(s, q, r)$ be a selection gadget in G' and let $S \subseteq V(G')$ be a target set with $|S \cap V(M)| = 3qs$. Then the following holds.*

1. *The qs vertices of S are in -pos and -neg parts of the selection gadget.*
2. *Suppose that v is a vertex in interface part of M but not in S. The vertex v cannot activate before all neighboring vertices of M adjacent to v are activated.*
3. *The number of vertices of $|S \cap V(M\text{-}pos)|$ is a multiple of q.*

Proof. For proof part 1 first observe that qs vertices of S must be in selection gadget in M. There cannot be more by the fact that we have $3qs$ vertices in total and Lemma 41; on the other hand, there cannot be less by Lemma 39. Now these qa vertices must be in the -pos and -neg parts by Lemma 40 part 1.

The part 2 follows immediately from Lemma 40 part 2.

Finally for the part 3, if $|S \cap V(M\text{-pos})|$ is not a multiple of q, then the check part of M does not activate by Lemma 44 and hence S is not a target set. □

We finally have all building blocks for the reduction. The graph G is constructed as follows (the numbers r_c, r_{cd}, $r_{c:cd}$ will be computed later).

- For each color class V_c we have a selection gadget $L_c = L(n, r_c)$.

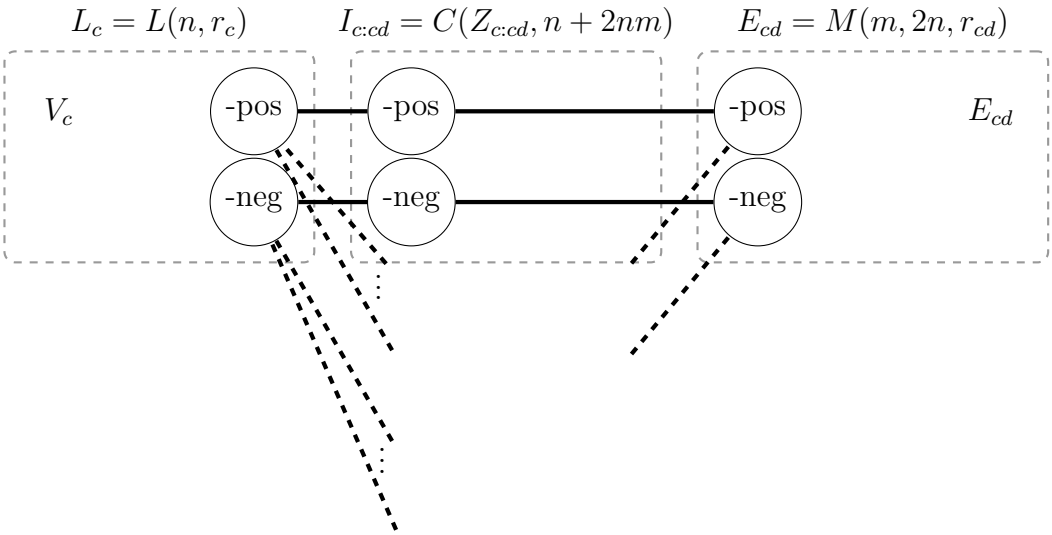

Figure 3.9: Overview of the reduction

- For each edge class E_{cd} we have a multiple selection gadget $M_{cd} = M(m, 2n, r_{cd})$.
- For each ordered pair (c, d) of colors, we have a check gadget $I_{c:cd} = C(Z_{c:cd}, r_{c:cd})$. The -pos part of $I_{c:cd}$ is connected to -pos part of L_c and M_{cd}. The -neg part is connected analogously.

We numerate each vertices in each class V_c by numbers from 0 to n and edges in each class E_{cd} by numbers from 0 to m. Then if $V_c = \{v_0, \ldots, v_n\}$ and $E_{cd} = \{e_0, \ldots, e_m\}$, we set

$$Z_{c:cd} = \{i + 2nj : v_i \in e_j\}.$$

It remains to compute the numbers r_c, r_{cd} and $r_{c:cd}$. The number $r_{c:cd}$ is $n + 2nm$, as the -pos part check gadget $I_{c:cd}$ is adjacent to -pos part of $L(n, r_c)$ gadget and -pos part of $M(m, 2n, r_{cd})$ gadget.

The gadget M_{cd} is adjacent to $I_{c:cd}$ and $I_{d:cd}$. To determine r_{cd}, we need to count the number of vertices of -pos parts in both adjacent gadgets. This can be expressed by weight function giving

$$r_{cd} = w(Z_{c:cd}) + w(Z_{d:cd}).$$

Finally, the gadget L_c is adjacent to $I_{c:cd}$ for every $d \neq c$. This analogously gives us

$$r_c = \sum_{d=1 d \neq c}^{k} w(Z_{c:cd}).$$

The budget b is set as

$$b = kn + \binom{k}{2}(6mn) + k(k-1)(4mn + 2n).$$

The first term kn is for k selection gadgets $L(n, r_c)$ corresponding to color classes. The second term accounts the $\binom{k}{2}$ multiple selection gadgets $M(m, 2n, r_c d)$ corresponding to edge classes E_{cd}; each of them requires at least $6mn$ vertices of target set by Lemma 45. The last term is for threshold decrease gadgets $D(2(2mn+n), w_{2mn+n}(Z_{c:cd}))$ inside each check gadget $C(Z_{cd}, 2mn+n)$.

We first check that G' has size poly($|G|$). First observe that each gadget is polynomial in size with respect to its parameters, and the parameters are polynomial with respect to m, n. This means that size of each gadget is polynomial with respect to $|G|$. Since we have $\mathcal{O}(k^2)$ gadgets, the whole graph G' is of size poly $|G|$. Moreover, the graph G' can be clearly constructed in polynomial time.

We turn our attention to the restricted modular width of G'. Consider a graph H obtained by replacing each -pos and each -neg part of each check gadget in G' by an independent set of same size. It is easy to see that H is a graph of neighborhood diversity $\mathcal{O}(k^2)$; every gadget in H has neighborhood diversity bounded by a constant. As each -pos or -neg part of check gadget in G' is a cograph, we can obtain G' from the type graph T_H by an appropriate substitution operation. This gives us an algebraic expression proving that restricted modular width of G' is $\mathcal{O}(k^2)$.

We need to show that $G = (V_1 \cup \cdots \cup V_k, E)$ has a multicolored clique if and only if G' has a target set of size b.

Suppose that C is a multicolored clique in G. Denote by ℓ_c the number of the vertex in V_c in the numeration we fixed before. Similarly denote by $\ell_{c,d}$ the number of edge of C that connects vertices in V_c and V_d.

The target set is constructed as follows:

1. put ℓ_c vertices of L_c-pos and $n - \ell_c$ vertices of L_c-neg into S,
2. put $2n\ell_{c,d}$ vertices of M_{cd}-pos and $2mn - 2n\ell_{c,d}$ vertices of M_{cd} into S, and
3. put all vertices into interface part of every threshold decrease gadget into S.

By the choice of b, we see that $|S| = b$. We claim that S is a target set.

We first turn our attention of check gadgets. By Lemma 44, the check gadget $C(Q_{2n,m}, 2nm)$ in M_{cd} activates in three rounds as $\ell_{c,d}$ is a multiple of $2n$ and hence is in $Q_{2n,m}$.

Similarly, as the vertex $v \in V_c$ corresponding to ℓ_c is incident to the edge $e \in E_{cd}$ corresponding to $\ell_{c,d}$, the gadget $I_{c:cd}$ activates in three rounds again by Lemma 44, since the number $\ell_c + 2n\ell_{c,d}$ is in $Z_{c:cd}$.

We now analyze gadget $L_c = L(n, r_c)$. The part L_c-guard activates in first round. In round three, all neighboring check gadgets activate. Since those gadgets together with L_c-guard constitute exactly half of neighbors of every vertex in L_c-pos or L_c-neg, these parts activate in round four. It is now easy to see that the remaining two groups activate in round five and six.

The argument for the gadget $M_{cd} = M(m, 2n, r_{cd})$ is analogous – in third round, all check gadgets (including the one inside M_{cd}) are activated and again this is enough to activate the M_{cd}-pos and M_{cd}-neg parts in round four. The rest of the gadget again activates in round six.

This proves that S is a target set.

For the opposite direction, suppose that we have a target set S in G' with $|S| \leq b$. We know that we can without loss of generality assume that S is hopeful.

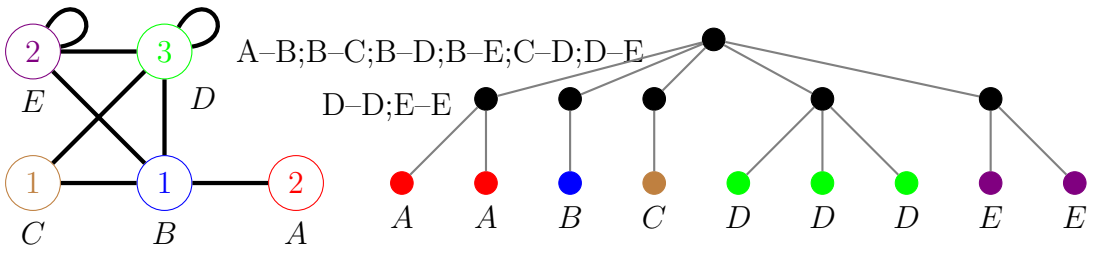

Figure 3.10: A graph with neighborhood diversity 5 (to the left) and a corresponding tree-model with 5 colors and depth 2 (to the right).

Let ℓ_c be the number represented by S in V_c and $2n\ell_{c,d}$ the number represented by S in M_{cd}. We know that the number represented by S in M_{cd} is a multiple of $2n$ by Lemma 46. Now pick ℓ_c-th vertex from each color class V_c. We claim that those vertices form a clique.

Since S was a target set, we obtain from Lemma 44 that $\ell_c + 2n\ell_{c,d} \in Z_{c:cd}$. Note that a number $z \in Z_{c:cd}$ is of the form $z = i + 2nj$ and the number i and j are uniquely determined by z alone; simply take $i = z \bmod 2n$ and $j = z \operatorname{div} 2n$. This means that number $\ell_c + 2n\ell_{c,d}$ is in Z if and only if vertex $v \in V_c$ corresponding to number ℓ_c is incident to edge $e \in E_{cd}$ corresponding to number $\ell_{c,d}$. Thus, we have proved that the set of vertex obtained from S indeed form a multicolored clique.

This finishes the proof of Theorem 25.

3.3.3 Consequences for Shrub-depth

In this section we revisit the proof of Theorem 25 and prove that the graphs showing W[1]-hardness with respect to modular-width admit a tree-model of constant depth.

Observe that the refinement of the former reduction for graphs with bounded neighborhood diversity uses only special type of cographs instead of previously used independent sets. The new nodes of the resulting graph are only disjoin union of cliques (of different sizes). It is not hard to see that one can prove that if a class of graph has bounded neighborhood diversity by k, then each graph in the class has tree-model of depth 2 (while using at most k colors). See Figure 3.10 for an example.

Proof of Theorem 26. We begin with the following observation.

Claim 1. Let $\mathcal{G}$ be a class of graphs that admit a tree-model of depth d and let $\mathcal{H}$ be a class of graph containing only disjoint union graphs from $\mathcal{G}$. Then $\mathcal{H}$ admits a tree-model of depth $d + 1$.

Proof. To see this, take a graph $H \in \mathcal{H}$ with $H = G_1 \cup \cdots \cup G_k$ where $G_i \in \mathcal{G}$ for every $i = 1, \ldots, k$. Now take the disjoin union of tree-models T_i for graphs G_i for $i = 1, \ldots, k$ and connect the roots via a new vertex v. To finish the construction there are no edges between any two colors introduced by the new root v. Note that the colors used in T_i need not to be disjoint. Thus we have proven that if $\mathcal{G} \subseteq \mathcal{TM}_m(d)$, then $\mathcal{H} \subseteq \mathcal{TM}_m(d + 1)$. □

Let $\mathcal{C}$ be the class of cliques and observe that $\mathcal{C} \subseteq \mathcal{TM}_1(1)$. It follows from Claim 1 that each node in the W[1]-hardness reduction admits a tree-model with one color of depth at most 2.

To finish the proof let t be the number of nodes from the proof of Theorem 25 and observe that $t = f(k)$ where k is the parameter from the k-Multicolored Clique problem. We enumerate all nodes by numbers in $[t]$ and assign a color i with a node enumerated by i. We take the disjoint union of tree-models T_i for every node $i = 1, \ldots, t$ and connect the roots via a new root vertex v. Finally observe that v all edges between nodes can be introduced in v (in a similar way to neighborhood diversity). Thus we conclude that the graphs from our reduction are in the class $\mathcal{TM}_{f(k)}(3)$. □

3.4 Conclusions

We have generalized ideas of previous works [4, 64] for the Target Set Selection problem. The presented results give new methods for showing W[1]-hardness result. In particular, only few problems are known to be W[1]-hard when parameterized by neighborhood diversity – which is the case for the Target Set Selection problem.

Thus, we would like to address several open problems regarding structural parameterizations of the Target Set Selection problem. Determine parameterized complexity of

- the Majority Target Set Selection problem parameterized by cluster vertex deletion number number [12];
- the Target Set Selection problem parameterized by the modular-width and the threshold upper-bound t (that is $f(v) \leq t$ for each vertex v);
- the Target Set Selection problem parameterized by shrub-depth and the threshold upper-bound t,
- the Target Set Selection problem parameterized by twin cover number;
- the Target Set Selection problem parameterized by the distance to clique [12].

Finally, we are not aware of other positive results concerning the number of different thresholds instead of the threshold upper-bound.

We would like to point out that in our proofs of W[1]-hardness the activation process terminates after constant number of rounds (independent of the parameter value and the size of the input graph). This is true also for all reductions given by Chopin et al. [12]. We conclude that both Target Set Selection and Majority Target Set Selection problems are paraNP-hard when parameterized by the number of rounds and shrub-depth.

4. Fixed parameter complexity of distance constrained labeling and uniform channel assignment problems

4.1 Introduction

The frequency assignment problem in wireless networks yields an abundance of various mathematical models and related problems. We study a group of such discrete optimization problems in terms of parameterized computational complexity, which is one of the central paradigms of contemporary theoretical computer science. We study parameterization of the problems by *clique width* and particularly by *neighborhood diversity* (nd), a graph parameter lying between clique width and the size of a minimum vertex cover.

All these problems are NP-hard even for constant clique width, including the uniform variant, as we show in this paper. On the other hand, we prove that they are in FPT with respect to nd. Such fixed parameter tractability has so far only been known only for the special case of $L(p, 1)$ labeling when parameterized by vertex cover [32].

4.1.1 Distance constrained labelings

Given a k-tuple of positive integers $p_1, \ldots, p_k$, called *distance constraints*, an $L(p_1, \ldots, p_k)$-labeling of a graph is an assignment l of integer labels to the vertices of the graph satisfying the following condition: Whenever vertices u and v are at distance i, the assigned labels differ by at least p_i. Formally, $\text{dist}(u, v) = i \Longrightarrow |l(u) - l(v)| \geq p_i$ for all $u, v : \text{dist}(u, v) \leq k$. Often only non-increasing sequences of distance constraints are considered.

Any $L(1)$-labeling is a graph coloring and vice-versa. Analogously, any coloring of the k-th distance power of a graph is an $L(1, \ldots, 1)$-labeling. The concept of $L(2, 1)$-labeling is attributed to Roberts by Griggs and Yeh [42]. It is not difficult to show that whenever l is an optimal $L(p_1, \ldots, p_k)$-labeling within a range $[0, \lambda]$, then the so called *span* λ is a linear combination of $p_1, \ldots, p_k$ [42, 56]. In particular, a graph G allows an $L(p_1, \ldots, p_k)$-labeling of span λ if and only if it has an $L(cp_1, \ldots, cp_k)$-labeling of span $c\lambda$ for any positive integer c.

For computational complexity purposes, we define the following class of decision problems:

$L(p_1, \ldots, p_k)$-LABELING	
Input:	graph G and a positive integer λ
Task:	resolve whether there is an $L(p_1, \ldots, p_k)$ labeling of G using labels from the interval $[0, \lambda]$

The $L(2, 1)$-LABELING problem was shown to be NP-complete by Griggs and Yeh [42] by a reduction from HAMILTONIAN CYCLE (with $\lambda = |V_G|$). Fiala,

Kratochvíl and Kloks [33] showed that $L(2,1)$-LABELING remains NP-complete also for all fixed $\lambda \geq 4$, while for $\lambda \leq 3$ it is solvable in linear time.

Despite a conjecture that $L(2,1)$-LABELING remains NP-complete on trees [42], Chang and Kuo [9] showed a dynamic programming algorithm for this problem, as well as for all $L(p_1,p_2)$-labelings where p_2 divides p_1. All the remaining cases of the $L(p_1,p_2)$-LABELING problem on trees have been shown to be NP-complete by Fiala, Golovach and Kratochvíl [31]. The same authors showed that $L(2,1)$-LABELING is already NP-complete on series-parallel graphs [30], which have of tree width at most 2. Note that these results imply NP-hardness of $L(3,2)$-LABELING on graphs of clique width at most 3 and of $L(2,1)$-LABELING for clique width at most 6 [15].

On the other hand, when λ is fixed, then the existence of an $L(p_1,\ldots,p_k)$-labeling of G can be expressed in MSO_1, hence it allows a linear time algorithm on any graph of bounded clique width [55].

Fiala et al. [32] showed that the problem of $L(p,1)$-LABELING is FPT when parameterized by p together with the size of the vertex cover. They also ask for the complexity characterization of the related CHANNEL ASSIGNMENT problem. We extend their work to the broader class of graphs and, consequently, in our Theorem 50 we provide a solution for their open problem.

4.1.2 Channel assignment

Channel assignment is a concept closely related to distance constrained graph labeling. Here, every edge has a prescribed weight $w(e)$ and it is required that the labels of adjacent vertices differ at least by the weight of the corresponding edge. The associated decision problem is defined as follows:

CHANNEL ASSIGNMENT	
Input:	graph G, a positive integer λ, and edge weights $w : E_G \to \mathbb{N}$
Task:	resolve whether there is a labeling l of the vertices of G by integers from $[0,\lambda]$ such that $\lvert l(u) - l(v)\rvert \geq w(u,v)$ for all $(u,v) \in E_G$

The maximal edge weight is an obvious necessary lower bound for the span of any labeling. Observe that for any bipartite graph, in particular also for all trees, it is also an upper bound — a labeling that assigns 0 to one class of the bipartition and $w_{\max} = \max\{w(e), e \in E_G\}$ to the other class satisfies all edge constraints. McDiarmid and Reed [61] showed that it is NP-complete to decide whether a graph of tree width 3 allows a channel assignment of given span λ. This NP-hardness hence applies on graphs of clique width at most 12 [15]. It is worth noting that for graphs of tree width 2, i.e. for subgraphs of series-parallel graphs, the complexity characterization of CHANNEL ASSIGNMENT is still open. Only a few partial results are known [73], among others that CHANNEL ASSIGNMENT is polynomially solvable on graphs of bounded tree width if the span λ is bounded by a constant.

Any instance G, λ of the $L(p_1,\ldots,p_k)$-LABELING problem can straightforwardly be reduced to an instance G^k, λ, w of the CHANNEL ASSIGNMENT problem. Here, G^k arises from G by connecting all pairs of vertices that are in G at distance at most k, and for the edges of G^k we let $w(u,v) = p_i$ whenever $\mathrm{dist}_G(u,v) = i$.

The resulting instances of Channel Assignment have by the construction some special properties. We explore and generalize these to obtain a uniform variant of the Channel Assignment problem.

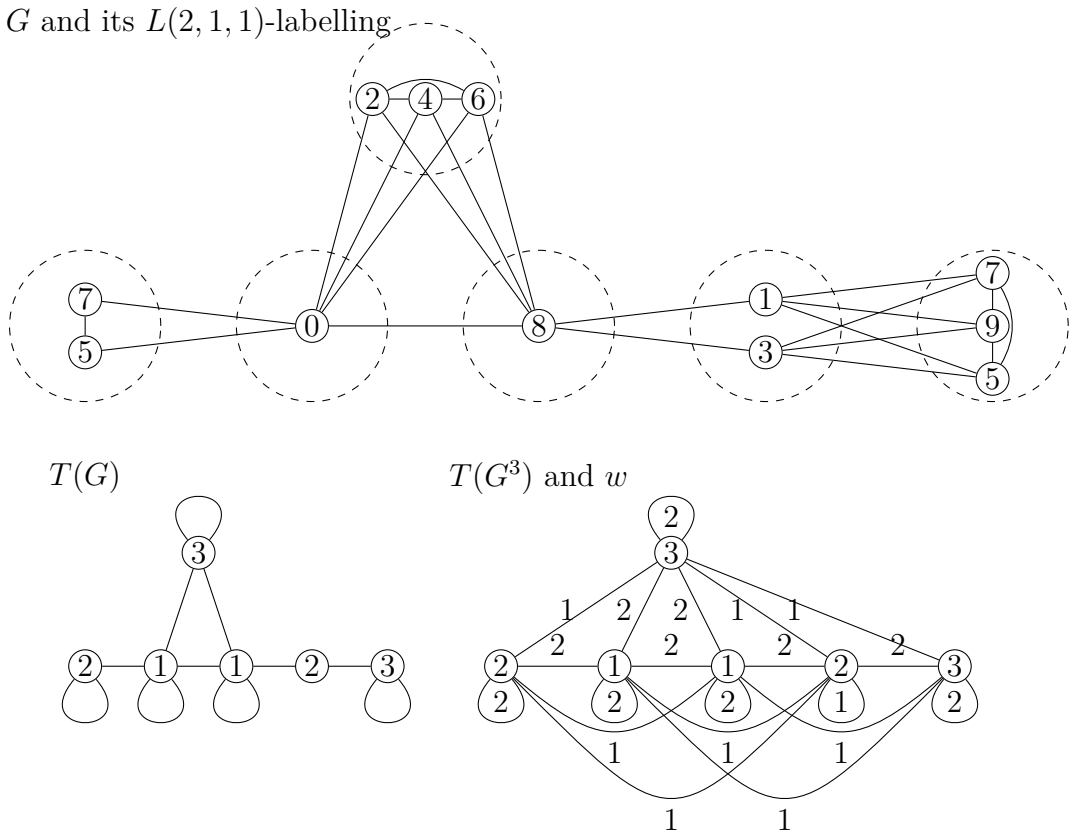

Figure 4.1: An example of a graph with its neighborhood diversity decomposition. Vertex labels indicate one of its optimal $L(2,1,1)$-labelings. The corresponding type graph. The weighted type graph corresponding to the resulting instance of the Channel Assignment problem.

For our purposes, i.e. to decide existence of a suitable labeling of a graph G, it suffices to consider only its type graph, as G can be uniquely reconstructed from $T(G)$ (upto an isomorphism) and vice-versa.

Moreover, the reduction of $L(p_1, \ldots, p_k)$-labeling to Channel Assignment preserves the property of bounded neighborhood diversity:

Observation 47. *For any graph G and any positive integer k it holds that* $\mathrm{nd}(G) \geq \mathrm{nd}(G^k)$.

Proof. The optimal neighborhood diversity decomposition of G is a neighborhood diversity decomposition of G^k. □

4.1.3 Our contribution

Our goal is an extension of the FPT algorithm for $L(2,1)$-labeling on graphs of bounded vertex cover to broader graph classes and for rich collections of distance constraints. In particular, we aim at $L(p_1, \ldots, p_k)$-labeling on graphs of bounded neighborhood diversity.

For this purpose we utilize the aforementioned reduction to Channel Assignment, taking into account that the neighborhood diversity remains bounded, even though the underlying graph changes.

It is worth to note that we must adopt additional assumptions for the Channel Assignment since otherwise it is NP-complete already on complete graphs, i.e. on graphs with $\mathrm{nd}(G) = 1$. To see this, we recall the construction of Griggs and Yeh [42]. They show that a graph H on n vertices has a Hamiltonian path if and only if the complement of H extended by a single universal vertex allows an $L(2,1)$-labeling of span $n+1$. As the existence of a universal vertex yields diameter at most two, the underlying graph for the resulting instance of Channel Assignment is K_{n+1}.

On the other hand, the additional assumptions on the instances of Channel Assignment will still allow us to reduce any instance of the $L(p_1, \ldots, p_k)$-labeling problem. By the reduction, all edges between classes of the neighborhood diversity decomposition are assigned the same weight. We formally adopt this as our additional constraint as follows:

Definition 24. *The edge weights w on a graph G are* nd-uniform *if it holds that $w(u,v) = w(u',v')$ whenever $u \sim u'$ and $v \sim v'$ with respect to the optimal neighborhood diversity decomposition. In a similar way we define uniform weights with respect to a particular decomposition.*

Our main contribution is an algorithm for the following scenario:

Theorem 48. *The* Channel Assignment *problem on* nd*-uniform instances is* FPT *when parameterized by* nd *and $w_{\max}$, where $w_{\max} = \max\{w(e) \colon e \in E_G\}$.*

Immediately, we get the following consequence:

Theorem 49. *For $p_1, \ldots, p_k$, the $L(p_1, \ldots, p_k)$-*labeling *problem is* FPT *when parameterized by* nd*, k and maximum p_i (or equivalently by* nd *and the k-tuple $(p_1, \ldots, p_k)$).*

Furthermore, our FPT result for Channel Assignment extends to vertex cover even without the uniformity requirement.

Theorem 50. *The* Channel Assignment *problem is* FPT *when parameterized by $w_{\max}$ and the size of vertex cover.*

One may ask whether the uniform version of Channel Assignment allows an FPT algorithm also for a broader class of graphs. Finally, we show that a natural generalization of this concept on graphs of bounded clique width yields an NP-complete problem on graphs of clique width at most 5.

4.2 Representing labelings as sequences and walks

We now focus on the nd-uniform instances of the Channel Assignment problem. It has been already mentioned that the optimal neighborhood diversity decomposition can be computed in cubic time. The test, whether it is nd-uniform, could

be computed in quadratic additional time. On the other hand, on nd-uniform instances it suffices to consider only the type graph, whose edges take weights from the edges of the underlying graph (see Fig. 4.1), since such a weighted type graph corresponds uniquely to the original weighted graph, up to an isomorphism.

Hence without loss of generalization assume that our algorithms are given the type graph whose edges are weighted by separation constraints w, however we express the time complexity bounds in terms of the size of the original graph.

Without loss of generality we may assume that the given graph G and its type graph $T(G)$ are connected, since connected components can be treated independently.

If the type graph $T(G)$ contains a type t not incident with a loop, we may reduce the channel assignment problem to the graph G', obtained from G by deleting all but one vertices of the type t. Any channel assignment of G' yields a valid channel assignment of G by using the same label on all vertices of type t in G as was given to the single vertex of type t in G'. Observe that adding a loop to a type, which represents only a single vertex, does not affect the resulting graph G'. Hence we assume without loss of generality that all types are incident with a loop. We call such type graph *reflexive*.

Observation 51. *If the type graph $T(G)$ is reflexive, then vertices of G of the same type have distinct labels in every channel assignment.*

Up to an isomorphism of the graph G, any channel assignment l is uniquely characterized by a sequence of type sets as follows:

Lemma 52. *Any weighted graph G corresponding to a reflexive weighted type graph $T(G), w$ allows a channel assignment of span λ, if and only if there exists a sequence of sets $\mathcal{T} = T_0, \ldots, T_\lambda$ with the following properties:*

(i) $T_i \subseteq V_{T(G)}$ *for each* $i \in [0, \lambda]$,

(ii) for each $t \in V_{T(G)} : s(t) = |\{T_i : t \in T_i\}|$,

(iii) for all $(t, r) \in E_{T(G)} : (t \in T_i \wedge r \in T_j \wedge (t \neq r \vee i \neq j)) \Rightarrow |i - j| \geq w(t, r)$

Proof. Given a channel assignment $l : V_G \to [0, \lambda]$, we define the desired sequence $\mathcal{T}$, such that the i-th element is the set of types that contain a vertex labeled by i. Formally $T_i = \{t : \exists u \in V_t : l(u) = i\}$. Now

(i) each element of the sequence is a set of types, possibly empty,

(ii) as all vertices of V_i are labeled by distinct labels by Observation 51, any type t occurs in $s(t)$ many elements of the sequence

(iii) if u of type t is labeled by i, and it is adjacent to v of type r labeled by j, then $|i - j| = |l(u) - l(v)| \geq w(u, v) = w(t, r)$, i.e. adjacent types t and r may appear in sets that are in the sequence at least $w(t, r)$ apart.

In the opposite direction assume that the sequence $\mathcal{T}$ exists. Then for each set T_i and type $t_j \in T_i$ we choose a distinct vertex $u \in V_j$ and label it by i, i.e. $l(u) = i$.

Now the condition (ii) guarantees that all vertices are labeled, while condition (iii) guarantees that all distance constraints are fulfilled. □

Observe that Lemma 52 poses no constraints on pairs of sets T_i, T_j that are at distance at least $w_{\max}$. Hence, we build an auxiliary directed graph D on all possible sequences of sets of length at most $z = w_{\max} - 1$.

The edges of D connect those sequences that overlap on a fragment of length $z - 1$, i.e. when they could be consecutive in $\mathcal{T}$. This construction is well known from the so-called shift register graph.

Definition 25. *For a general graph F and weights $w : E_F \to [1, z]$ we define a directed graph D such that*

- *the vertices of V_D are all z-tuples $(T_1, \ldots, T_z)$ of subsets of V_F such that for all $(t, r) \in E_F : (t \in T_i \wedge r \in T_j) \Rightarrow |i - j| \geq w(t, r)$*
- *$((T_1, \ldots, T_z), (T'_1, \ldots, T'_z)) \in E_D \Leftrightarrow T'_i = T_{i+1}$ for all $i \in [1, z - 1]$.*

As the first condition of the above definition mimics (iii) of Lemma 52 with $F = T(G)$, any sequence $\mathcal{T}$ that justifies a solution for $(T(G), w, \lambda)$, can be transformed into a walk of length $\lambda - z + 1$ in D.

In the opposite direction, namely in order to construct a walk in D, that corresponds to a valid channel assignment, we need to guarantee also an analogue of the condition (ii) of Lemma 52. In other words, each type should occur sufficiently many times in the resulting walk. Indeed, the construction of D is independent on the function s, which specifies how many vertices of each type are present in G.

In this concern we consider only special walks that allow us to count the occurrences of sets within z-tuples. Observe that V_D also contains the z-tuple $\emptyset^z = (\emptyset, \ldots, \emptyset)$. In addition, any walk of length $\lambda - z + 1$ can be converted into a closed walk from $\emptyset^z$ of length $\lambda + z + 1$, since the corresponding sequence $\mathcal{T}$ can be padded with z additional empty sets at the front, and another z empty sets at the end. From our reasoning, the following claim is immediate:

Lemma 53. *A closed walk $\mathcal{W} = W_1, \ldots, W_{\lambda+z+1}$ on D where $W_1 = W_{\lambda+z+1} = \emptyset^z$, yields a solution of the* Channel Assignment *problem on a* nd*-uniform instance G, w, λ with reflexive $T(G)$, if and only if $s(t) = |\{W_i : t \in (W_i)_1\}|$ holds for each $t \in V_{T(G)}$.*

We found interesting that our representation of the solution resembles the NP-hardness reduction found by Griggs and Yeh [42] (it was briefly outlined in Section 4.1.3) and later generalized by Bodlaender et al. [7]. The key difference is that in their reduction, a Hamilton path is represented by a sequence of vertices of the constructed graph. In contrast, we consider walks in the type graph, which is assumed to be of limited size.

4.3 The algorithm

In this section we prove the following statement, which directly implies our main result, Theorem 48:

Proposition 54. *Let G, w be a weighted graph, whose weights are uniform with respect to a neighborhood diversity partition with τ classes.*

Then the Channel Assignment *problem can be decided on G, w and any λ in time $2^{2^{O(\tau w_{\max})}} \log n$, where n is the number of vertices of G, provided that G, w are described by a weighted type graph $T(G)$ on τ nodes.*

A suitable labeling of G can be found in additional $2^{2^{O(\tau w_{\max})}} n$ time.

Proof. According to Lemma 53, it suffices to find a closed walk $\mathcal{W}$ (if it exists) corresponding to the desired labeling l. From the well-known Euler's theorem it follows that any directed closed walk $\mathcal{W}$ yields a multiset of edges in D that induces a connected subgraph and that satisfies Kirchhoff's law. In addition, any such suitable multiset of edges can be converted into a closed walk, though the result need not be unique.

For this purpose we introduce an integer variable $\alpha_{(W,U)}$ for every directed edge $(W, U) \in E_D$. The value of the variable $\alpha_{(W,U)}$ is the number of occurrences of (W, U) in the multiset of edges.

Kirchhoff's law is straightforwardly expressed as:

$$\forall W \in V_D : \sum_{U:(W,U)\in E_D} \alpha_{(W,U)} - \sum_{U:(U,W)\in E_D} \alpha_{(U,W)} = 0$$

In order to guarantee connectivity, observe first that an edge (W, U) and $\emptyset^z$ would be in distinct components of a subgraph of D, if the subgraph is formed by removing edges that include a cut C between (W, U) and $\emptyset^z$. Now, the chosen multiset of edges is disconnected from $\emptyset^z$, if there is such an edge (W, U) together with a cut set C such that $\alpha_{(W,U)}$ has a positive value, while all variables corresponding to elements of C are zeros. As all variable values are bounded above by λ, we express that C is not a cutset for the chosen multiset of edges by the following condition:

$$\alpha_{(W,U)} - \lambda \sum_{e\in C} \alpha_e \leq 0$$

To guarantee the overall connectivity, we apply the above condition for every edge $(W, U) \in E_D$, where $W, U \neq \emptyset^z$, and for each set of edges C that separates W or U from $\emptyset^z$.

The necessary condition expressed in Lemma 53 can be stated in terms of variables $\alpha_{(W,U)}$ as

$$\forall t \in V_{T(G)} : \sum_{W:t\in(W)_1} \sum_{U:(W,U)\in E_D} \alpha_{(W,U)} = s(t)$$

Finally, the size of the multiset is the length of the walk, i.e.

$$\sum_{(W,U)\in E_D} \alpha_{(W,U)} = \lambda + z + 1$$

Observe that these conditions for all (W, U) and all suitable C indeed imply that the $\emptyset^z$ belongs to the subgraph induced by edges with positively evaluated variables $\alpha_{(W,U)}$.

Algorithm 1 summarizes our deductions.

To complete the proof, we argue about the time complexity as follows:

- Line 1 needs $O(|E_{T(G)}|) = O(\tau^2)$ time.

Input: A reflexive type graph $T(G)$ whose edges are labeled by w and a span λ.

Output: A channel assignment $l : G \to [0, \lambda]$ respecting constraints w, if it exists.

begin

1 Compute $z := w_{\max} - 1$;

2 Construct the directed graph D;

3 Solve the following ILP in variables $\alpha_{(W,U)} : (W, U) \in E_D$:

4 for each $(W, U) \in E_D$:

$$\alpha_{(W,U)} \geq 0$$

5 for each $W \in V_D$:

$$\sum_{U:(W,U)\in E_D} \alpha_{(W,U)} - \sum_{U:(U,W)\in E_D} \alpha_{(U,W)} = 0$$

6 for each $(W, U) \in E_D$ and each cutset C between (W, U) and $\emptyset^z$ in D:

$$\alpha_{(W,U)} - \lambda \sum_{e\in C} \alpha_e \leq 0$$

7 for each $t \in V_{T(G)}$:

$$\sum_{W:t\in(W)_1} \sum_{U:(W,U)\in E_D} \alpha_{(W,U)} = s(t)$$

8 $$\sum_{(W,U)\in E_D} \alpha_{(W,U)} = \lambda + z + 1;$$

9 **if** *the ILP has a solution* **then**

10 find a walk $\mathcal{W}$ that traverses each edge (W, U) exactly $\alpha_{(W,U)}$ times;

11 convert the walk $\mathcal{W}$ into a labeling l and **return** l;

else

12 **return** *"No channel assignment l of span λ exists."*

end

end

Algorithm 1: Solving the Channel Assignment problem.

- As D has at most $2^{\tau z}$ nodes and at most $2^{\tau(z+1)}$ edges, line 2 needs $2^{O(\tau z)}$ time.
- Similarly, conditions at lines 4 and 5 require $2^{O(\tau z)}$ time and space to be composed. Analogously, conditions at lines 7 and 8 involve coefficients that are proportional to the size of the original graph G (namely λ and $s(t)$), hence $2^{O(\tau z)} \log n$ time and space is needed here.
- For line 6, we examine each subset of E_D and decide whether it is a suitable cutset C. There are at most $2^{2^{\tau(z+1)}}$ choices for C, so the overall time and space complexity for the composition of conditions at line 6 is $2^{2^{O(\tau z)}} \log n$.
- Frank and Tardos [37] (improving the former result due to Lenstra [58]) showed that the time needed to solve the system of inequalities with p integer variables is $O(p^{2.5p+o(p)}L)$, where L is the number of bits needed to encode the input. As we have $2^{O(\tau z)}$ variables and the conditions are encoded in space $2^{2^{O(\tau z)}} \log n$, the time needed to resolve the system of inequalities is $2^{2^{O(\tau z)}} \log n$.

- A solution of the ILP can be converted into the walk in time $2^{2^{O(\tau z)}}n$, and the same bound applies to the conversion of a walk to the labeling at lines 10 and 11.

Observe that if only the existence of the labeling should be decided, the lines 10 and 11 need not to be executed, only an affirmative answer needs to be returned instead. □

We are aware the the double exponential dependency on nd and $w_{\max}$ makes our algorithm interesting mostly from the theoretical perspective. Naturally, one may ask, whether the exponential tower height might be reduced or whether some nontrivial lower bounds on the computational complexity could be established (under usual assumptions on classes in the complexity hierarchy).

4.4 Bounded vertex-cover

We utilize the results of the previous sections to derive an FPT algorithm proposed as Theorem 50.

Proof of Theorem 50. Given a graph G and its optimal vertex cover U, we construct a partition of the vertices of G as follows. Next let $I = V(G) \setminus U$ be the independent set of G. We form a partition of I as follows: For every subset $X \subseteq U$ we define:

$$I_X := \{v \in I \colon \{v, x\} \in E(G) \text{ for } x \in X \text{ and } \{v, x\} \notin E(G) \text{ for } x \notin X\}.$$

Observe that for any $u, v \in I_X$ it holds that $N(u) = X = N(v)$, and hence also $u \sim v$. In particular, the optimal neighborhood diversity decomposition of G consists of all nonempty sets I_X together with a suitable partition of U. Consequently, $\mathrm{nd}(G) \leq 2^{\mathrm{vc}(G)} + \mathrm{vc}(G)$.

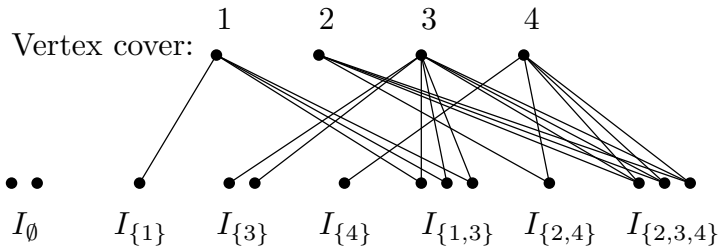

Figure 4.2: An example of a neighborhood diversity decomposition based on a vertex cover.

We further refine sets I_X, so that the edge-weights became uniform. For a set $X = \{x_1, \ldots, x_k\} \subseteq U$ and each k-tuple of positive integers $\mathbf{w} = (w_1, \ldots, w_k)$ with $0 < w_i \leq w_{\max}$ for every $1 \leq i \leq k$ we define the set $I_X^{\mathbf{w}}$ as

$$I_X^{\mathbf{w}} := \{v \in I_X \colon w(v, x_i) = w_i \text{ for } 1 \leq i \leq k\}.$$

Observe that any refinement of a neighborhood diversity decomposition is again a decomposition. We now estimate the number of types of the refined

decomposition. The number of types of the refined decomposition can be upper-bounded by $\mathrm{vc}(G) + (2^{\mathrm{vc}(G)})w_{\max}^{\mathrm{vc}(G)}$. To finish the proof we apply Proposition 54 on the refined decomposition. □

4.5 NLC-uniform channel assignment

One may ask whether the concept of nd-uniform weights could be extended to broader graph classes. We show, that already its direct extension to graphs of bounded clique width makes the Channel Assignment problem NP-complete. Instead of clique width we express our results in terms of NLC-width [74] (NLC stands for node label controlled). The parameter NLC-width is linearly dependent on clique width, but it is technically simpler.

We now briefly review the related terminology. A NLC-decomposition of a graph G is a rooted tree whose leaves are in one-to-one correspondence with the vertices of G. For the purpose of inserting edges, each vertex is given a label (the labels for channel assignment are now irrelevant), which may change during the construction of the graph G. Internal nodes of the tree are of two kinds: *relabel* nodes and *join* nodes.

Each relabel node has a single child and as a parameter takes a mapping ρ on the set of labels. The graph corresponding to a relabel node is isomorphic to the graph corresponding to its child, only ρ is applied on each vertex label.

Each join node has a two children and as a parameter takes a binary relation S on the set of labels. The graph corresponding to a join node is isomorphic to the disjoint union of the two graphs G_1 and G_2 corresponding to its children, where further edges are inserted as follows: $u \in V_{G_1}$ labeled by i is made adjacent to $v \in V_{G_2}$ labeled by j if and only if $(i, j) \in S$.

The minimum number of labels needed to construct at least one labeling of G in this way is the NLC width of G, denoted by $\mathrm{nlc}(G)$.

Observe that $\mathrm{nlc}(G) \leq \mathrm{nd}(G)$ as the vertex types could be used as labels for the corresponding vertices and the adjacency relation in the type graph could be used for S in all join nodes. In particular, in this construction the order of performing joins is irrelevant and no relabel nodes are needed.

Definition 26. *The edge weights w on a graph G are* nlc-uniform *with respect to a particular NLC-decomposition, if $w(u, v) = w(u', v')$ whenever edges (u, v) and (u', v') are inserted during the same join operation and at the moment of insertion u, u' have the same label in G_1 and v, v' have the same label in G_2.*

Observe that our comment before the last definition justifies that weights that are uniform with respect to a neighborhood diversity decomposition are uniform also with respect to the corresponding NLC-decomposition.

Gurski and Wanke showed that the NLC-width remains bounded when taking powers of trees [43]. It is well known that NLC-width of a tree is at most three. Fiala et al. proved that $L(3, 2)$-labeling is NP-complete on trees [31]. To combine these facts together we show that the weights on the graph arising from a reduction of the $L(3, 2)$-labeling on a tree to Channel Assignment are nlc-uniform.

Theorem 55. *The* Channel Assignment *problem is* NP*-complete on graphs with edge weights that are* nlc*-uniform with respect to an NLC-decomposition of width at most four.*

Proof. Let a tree T be an instance of the $L(3,2)$-labeling problem.

By induction on the size of T we show that T^2 allows an NLC-decomposition such that the weights w prescribed by the reduction of $L(3,2)$-labeling to Channel Assignment are nlc-uniform.

Assume for the induction hypothesis that such an NLC-decomposition exists for every tree T' on less than n vertices, where the labels of T' are distributed as follows: assume that T' is rooted in a vertex r', then r' is labeled by 1, its direct neighbors by 2 and all other vertices by 3.

Such a decomposition clearly exists for a tree on a single vertex.

Now consider a tree T on $n \geq 2$ nodes. Choose an edge (r', r'') arbitrarily and define two trees T' and T'' as the components of $T \setminus (r', r'')$, where T' contains r' and vice versa.

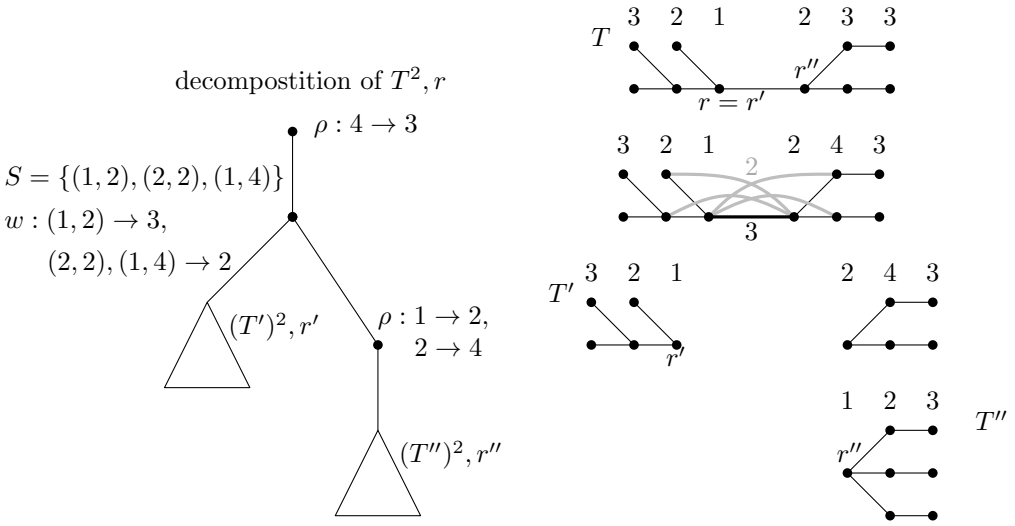

Figure 4.3: Recursive step in the construction of an NLC-decomposition. The original edges of T, T' and T'' are in black. The weighted edges added in this step are in bold.

By the induction hypothesis T' and T'' allow NLC-decompositions with the desired properties. Before we join trees T' and T'' together, we change labels in T'' as $1 \to 2, 2 \to 4$. At the join we will insert the following weighted edges: of weight 3 between vertices labeled 1 in T' and 2 in T'', and of weight 2 between vertices of labels 2 and 2, and between 1 and 4, respectively. Finally, we relabel $4 \to 3$ and promote r' to be the root r of T. All steps are depicted in Fig. 4.3. Observe that the result of this construction is T^2 with appropriate nlc-uniform weights, and that its labeling satisfies all conditions of the induction hypothesis. □

4.6 Conclusion

We have shown an algorithm for the Channel Assignment problem on nd-uniform instances and several complexity consequences for the $L(p_1, \ldots p_k)$-labeling problem. In particular, Theorem 49 extends known results for the

$L(p,1)$-LABELING problem to labelings with arbitrarily many distance constraints, answering an open question of [32]. Simultaneously, we broaden the considered graph classes by restricting neighborhood diversity instead of vertex cover.

While the main technical tools of our algorithms are bounded-dimension ILP programs, ubiquitous in the FPT area, the paper shows an interesting insight on the nature of the labelings over the type graph and the necessary patterns of such labelings of very high span. Note that the span of a graph is generally not bounded by any of the considered parameters and may be even proportional to the order of the graph.

Solving a generalized problem on graphs of bounded neighborhood diversity is a viable method for designing FPT algorithms for a given problem on graphs of bounded vertex cover, as demonstrated by this and previous papers. This promotes neighborhood diversity as a parameter that naturally generalizes the widely studied parameter vertex cover.

We would like to point out that the parameter *modular width*, proposed by Gajarský, Lampis and Ordyniak [38], offers further generalization of neighborhood diversity towards the clique width [20] (dependencies between these graph parameters are depicted in Fig. 1.1).

As an interesting open problem we ask whether it is possible to strengthen our results to graphs of bounded modular width or whether the problem might be already NP-complete for fixed modular width, as is the case with clique width. For example, the GRAPH COLORING problem ILP based algorithm for bounded neighborhood diversity translates naturally to an algorithm for bounded modular width. On the other hand, there is no apparent way how our labeling results could be adapted to modular width in a similar way.

5. Partitioning Graphs into Induced Subgraphs

5.1 Introduction

We begin with the definition of the Partition into H problem. We will then present the problem in the light of some well-known problems from computational complexity – for example Perfect Matching or Equitable Coloring – thus demonstrating it as a natural generalization of these and other problems. Finally, we give the summary of our results presented in this paper.

The Partition into H problem For graphs $G = (V, E), H = (W, F)$ with $|V| = |W| \cdot r$, we say that it is possible to *partition G into copies of H* if there exist disjoint sets $V_1, V_2, \ldots, V_r$ such that

- $\bigcup_{i=1}^{r} V_i = V$, and
- $G[V_i] \simeq H$ for every $i = 1, 2, \ldots, r$,

where by $G[V_i]$ we mean the subgraph of G induced by the set of vertices V_i. The pattern graph $H = (W, F)$ is fixed in the following problem definition.

Partition into H	
Input:	graph $G = (V, E)$ with $\|V\| = r \cdot \|W\|$ for an integer r
Task:	resolve whether there is a partition of G into copies of H

The complexity of the Partition into H problem has been studied by Hell and Kirkpatrick [52] and has been proven to be NP-complete for any fixed graph H with at least 3 vertices – they have studied the problem under a different name as the Generalized Matching problem. There are applications in the printed wiring board design [46] and code optimization [8].

Some variants of this problem are studied extensively in graph theory. For example when $H \simeq K_2$ the problem Partition into K_2 is the well known Perfect Matching problem, which can be solved in polynomial time due to Edmonds [28] – the algorithm works even for the optimization version, when one tries to maximize the number of copies of K_2 in G. The characterization theorem for $H \simeq K_2$, that is a characterization of graphs admitting a perfect matching is known due to Tutte [71].

Another frequently studied case of our problem is the Partition into K_3 problem – also known as the Triangle Partition problem. The Triangle Partition problem arises as a special case of the Set Partition problem (also known to be NP-complete [40]). Gajarský et al. [38] pointed out that the parameterized complexity of the Triangle Partition problem parameterized by the tree-width of the input graph was not resolved so far.

The last, but not least, example of a well known problem which can be viewed as a special case of the Partition into H problem is the Equitable Coloring problem. The task is to color the vertices of an n vertex graph with exactly k colors such that vertices connected by an edge receive different colors and the

resulting color classes have equal sizes. It is easy to see that the Equitable Coloring problem is the Partition into H problem with the edgeless graph on n/k vertices (it is possible to add a clique of appropriate size so that n becomes a multiple of k as it is demanded in our setting).

Very similar application can be found as the so called ℓ-bounded vertex colorings, where the task is to find a coloring of a graph G with prescribed number of colors such that each color is used at most ℓ-times. This problem allows a straightforward reduction to the Equitable Coloring by inserting a suitable number of isolated vertices. The connection between these two problems was also studied from the parameterized complexity point of view [6] – an XP algorithm is obtained for parameterization by the tree-width of graph G. Here a special case of this problem is again equivalent to the Partition into H problem with H being the edgeless graph on ℓ vertices. For this problem a polynomial time algorithm is known for trees [50]. Upper and lower bounds on the number of colors are known for general graphs [45].

Very recently van Bevern et al. [72] studied computational complexity of related problem, where for a fixed graph $H = (W, F)$ the task is to partition the set of vertices of an input graph $G = (V, E)$ into sets $V_1, V_2, \ldots, V_r$ such that $|V_i| = |W|$ and $G[V_i]$ contains subgraph isomorphic to H.

Parameterized complexity results. When dealing with an NP-hard problem it is usual to study the problem in the framework of parameterized complexity. While in the previous section we have introduced several problems of the classical complexity, here we give references to parameterized results for these problems.

A similar but more general problem (called the MSOL Partitioning problem) was studied by Rao [67]. Here the task is to partition the vertices of the graph G into several sets $A_1, A_2, \ldots, A_r$ such that $\varphi(A_i)$ holds for every $i = 1, 2, \ldots, r$, where $\varphi(\cdot)$ is an MSO_1 formula with one free set variable. If the number r and the clique-width $\mathrm{cw}(G)$ are fixed then the algorithm runs in polynomial time and hence the problem belongs to an XP class with parameterization by the clique-width.

Our contribution. Our first algorithm is based on the celebrated theorem of Courcelle [17] – an usual starting point for parameterized algorithm design. We would like to point out, that even though the result follows easily, the application is not straightforward.

Theorem 56. *For any fixed connected graph H the* Partition into H *problem is expressible by an MSO_2 formula.*

The proof of this theorem is rather technical and is contained in Section 5.5.

As the first algorithm is for graphs with bounded tree-width and thus a sparse class of graphs, we also analyze some variants of the Partition into H problem for a particular class of dense graphs. Many parameters are suitable for dense graph classes, such as neighborhood diversity and modular-width (we give formal definitions in Section 1.3).

Definition 27 (Prime graph)**.** *We say that a graph $G = (V, E)$ is* prime graph *if for every subset of vertices $U \subsetneq V$ with at least two vertices there exist $v \in V \setminus U$ such that v is connected to at least one vertex in U and v is not connected to at least one vertex in U.*

The class of prime graphs is thoroughly studied in the context of modular decompositions.

Theorem 57. *For any fixed prime graph H and a graph G the* Partition into H *problem belongs to the* FPT *class when parameterized by modular width of graph G.*

We derive the result using integer linear programming in a fixed dimension, which can be solved by a parameterized routine [58, 37]. It is worth to mention that even though the condition on prime graphs may seem very restrictive this class of graphs contains for example paths P_k on $k \geq 4$ vertices and cycles C_k on $k \geq 5$ vertices. Applications of the Partition into H problem with H being a path may be found in code optimization [8].

When it is shown that there is an FPT-algorithm for some problem, it is natural to ask, whether the problem admits a polynomial kernel – that is a preprocessing routine running in polynomial time which outputs an equivalent instance of size polynomially bounded in the assumed parameter. We prove that the Partition into H problem does not have polynomial kernel parameterized by modular-width for any reasonable graph H, that is when H has at least 3 vertices and thus the Partition into H problem is NP-hard. More precisely, we prove the following.

Theorem 58. *For any fixed graph H with at least 3 vertices. There is no polynomial kernel routine for the* Partition into H *problem when parameterized by modular width of graph G unless* NP $\subseteq$ coNP/poly.

Using techniques similar to those of proof of Theorem 57 we can prove that for every fixed H the Partition into H problem can be solved efficiently on graphs with bounded neighborhood diversity.

Theorem 59. *For any fixed graph H there is an* FPT*-algorithm for the* Partition into H *problem parameterized by neighborhood diversity of graph G.*

5.2 Preliminaries

For a graph $G = (V, E)$ we denote by $|G|$ the number of vertices of G. For a set U we denote by $\binom{U}{2}$ the set of all two element subsets of U, that is $\binom{U}{2} = \{\{u, v\} \colon u, v \in U, u \neq v\}$. Let $G = (V, E)$ be a graph and let $U \subseteq V$ the graph induced by U is denoted by $G[U]$ and it is the graph $(U, E \cap \binom{U}{2})$. Let $G = (V, E), H = (W, F)$ be graphs, we say that G is *isomorphic* to H, we denote this by $G \simeq H$, if there exists a bijective mapping $f : V \to W$ such that $\{u, v\} \in E$ if and only if $\{f(u), f(v)\} \in F$. For a set of vertices U we denote the set of incident edges as $\delta(U)$ that is the set of edges $\{\{u, v\} \colon u \in U, v \in V \setminus U\}$. Finally a complement of a graph $G = (V, E)$ is denoted by $\bar{G}$ is a graph on the same vertex set V with edge set $\binom{V}{2} \setminus E$. We say that a graph G is *connected* if there is a uv path in G for every two distinct vertices of G. For more notation on graphs, we refer reader to a monograph by Diestel [23].

We say that a relation R is an *equivalence relation* on a set X if R is reflexive, symmetric and transitive. An equivalence class determined by an element $x \in X$ is the set $\{y \in X \colon x \equiv_R y\}$.

5.3 Partition into H on graphs with bounded modular-width

In this section we give a proof of the Theorem 57. We begin with a technical Lemma 60 that demonstrates the limited possibilities of embedding the (fixed) prime graph H into the input graph G with bounded modular-width. This exploits a close connection of the parameters neighborhood diversity and modular-width.

We then formulate the problem as a mixed integer linear problem, where we can bound the number of integer variables by a function in the modular-width $\mathrm{mw}(G)$ of the input graph and thus proving the theorem.

By an *embedding* of graph $H = (W, F)$ in graph $G = (V, E)$ we mean a function $h : W \to V$ such that $G[h(W)] \simeq H$, to which we refer as an *induced copy* of H in G.

Recall that by Definition 27 for a prime graph $H = (W, F)$ it holds that $\forall U \subsetneq W$ with at least two vertices there exists vertex $w \in W \setminus U$ such that it is adjacent and non-adjacent to at least one vertex in U.

Embedding of H inside G The following lemma shows that restricting H to be prime graph leads to only two possibilities of an embedding of H on a particular level of a modular decomposition of the graph G.

Lemma 60. *Let $G = T(G_1, G_2, \ldots, G_k)$ be a graph and let $H = (W, F)$ be a prime graph. For an induced subgraph $G' \simeq H$ of G holds either*

1. *$G' \subseteq G_i$ for some $i \in \{1, 2, \ldots, k\}$, or*

2. *G' contains at most one vertex in every G_i for $i = 1, 2, \ldots, k$.*

Proof. Assume that $G' \nsubseteq G_i$ for any i as otherwise we are done. If there are at least two vertices of G' in some G_i, – we will show (using Definition 27) that this cannot be valid embedding of H inside G.

Let us rearrange vertices of a template graph T (and corresponding graphs G_i) so that G' contains vertices of graphs $G_1, G_2, \ldots, G_\ell$ (with $\ell < |H|$). Let $T' \subseteq T$ be the restriction of T to vertices $v_1, v_2, \ldots, v_\ell$. From our assumption on $\ell < |H|$ it follows that there exists a set $U \subsetneq W$ that is embedded inside one graph G_i. By the definition of prime graphs for this set U there exists vertex $w \in W \setminus U$ and two (different) vertices $u_1, u_2 \in U$ such that $\{u_1, w\} \in F$ and $\{u_2, w\} \notin F$. But this contradicts the fact that G' is an embedding of H inside G as by the definition of modular decomposition for all $v \in G_j$ with $j \neq i$ either v is adjacent to every vertex in U, or v is non-adjacent to all vertices in U. Thus, G' cannot be an embedding of H inside G. □

By Lemma 60 for the most complex operation follows that the structure inside graphs $G_1, G_2, \ldots, G_k$ is not important when we try to find graphs of the partition that contains vertices from more than one graph G_i.

Note that when deciding the Partition into H problem on graph G with $\mathrm{mw}(G) < |H|$, it follows from Lemma 60 that the answer is clearly No – thus this case is trivial. So the task is to design an algorithm when $|H| = \mathrm{nd}(H) \leq \mathrm{mw}(G)$.

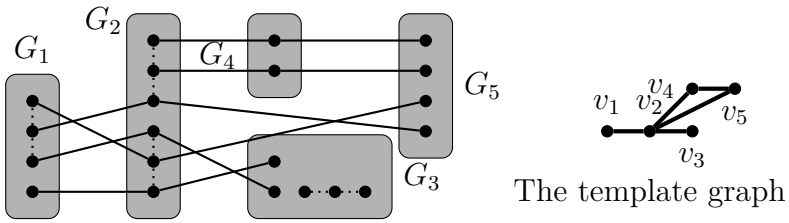

Figure 5.1: An example of a graph created by a modular decomposition. Template graph is given on the right. Paths inside blocks (precomputed by an induction) are shown as dotted, while an optimal solution for this graph is drawn with solid edges. The graph shows that it is essential to allow unlinking of previously (recursively) constructed solution.

5.3.1 Mixed integer linear program routine

We solve the Partition into H problem by a recursive algorithm. For each occurrence of operation 4 in the decomposition we design an integer linear program that computes the optimal solution, that is the solution in which there are as many induced copies of H in G as possible. The integer program uses previous solution thus we proceed (recursively) from leaves of the decomposition tree of G towards the root.

In the following mixed integer linear program for the Partition into H problem on the graph $G = T(G_1, G_2, \ldots, G_k)$ the set $\mathcal{S}$ is the set of all $|H|$-tuples of vertices of the graph T that form an induced copy of H inside T and thus, inside the graph G. An alternative point of view is that $\mathcal{S}$ is the set of all possible copies of H in T. Let U be the vertex set of the graph T, moreover, as graphs $G_1, G_2, \ldots, G_k$ correspond to vertices of T, we will denote these as G_v for $v \in U$.
The solution, constants and variables. The constants w_v represent the number of vertices of the graph G_v that are not covered by a copies of H found by the previous (recursive) solution. The constants p_v represent the number of copies of H in the recursive solution. Thus, the (recursive) solution is represented by pairs (p_v, w_v) for all $v \in U$. It may be wise to unlink some previously made copies of H's (computed by the recursive procedure) – this is why we introduce the variable y_v, which expresses how many previously constructed copies shall by unlinked. An example of a situation in which this is necessary is show on Figure 5.1. The ILP tries to cover as many vertices as possible – this is done by minimizing the number of uncovered vertices expressed by r_v for a graph G_v.

Mixed integer linear program

$$\text{minimize} \sum_{v \in U} r_i \quad \text{subject to} \quad r_v = w_v + |H| \cdot y_v - \sum_{S \in \mathcal{S}: S \ni v} x_S \qquad \forall v \in U$$

$$y_v \leq p_v \qquad \forall v \in U$$

$$\text{where} \qquad x_S \in \mathbb{N} \qquad \forall S \in \mathcal{S}$$

$$y_v \in \mathbb{N} \qquad \forall v \in U$$

$$r_v \geq 0 \qquad \forall v \in U$$

The total number of integral variables may be upper-bounded by $|T| + |T|^{|H|}$, so if we denote by k the size of the template graph (and thus the modular-width of an input graph) the upper bound can be expressed as $k + k^k$. We can apply

the following result of Lenstra [58] (with an enhancement due to Frank and Tardos [37]) to derive Theorem 57.

Proposition 61 ([58, 37])**.** *Let p be the number of integral variables in a Mixed integer linear program and let L be the number of bits needed to encode the program. Then it is possible to find an optimal solution in time $\mathcal{O}(p^{2.5p}\,\mathrm{poly}(L))$ and a space polynomial in L.*

5.3.2 Refuting polynomial kernels

In this section we prove Theorem 58. First observe that it suffices to prove the theorem only for connected graph H. This is trivial as for every graph G it holds that $\mathrm{mw}(G) = \mathrm{mw}(\bar{G})$ (the complement of graph G) – and thus we can ask the question in complementary setting.

As a polynomial equivalence relation we take the following relation. Two instances $(G_1, H_1), (G_2, H_2)$ are equivalent if $|G_1| = |G_2|$ (that is they have the same number of vertices) and $H_1 \simeq H_2$. As H is fixed this defines a polynomial equivalence relation (together with the class of malformed instances – instances that do not encode a pair of graphs).

Observe that if we take a disjoint union of two graphs G, G' then $\mathrm{mw}(G \dot\cup G') \leq \max\{|G|, |G'|\}$. This is not hard to see as we can take G to be a type graph for G and similarly G' to be a type graph for G'. Then the disjoint union does not change the modular-width as the second operation of modular decomposition is exactly the disjoint union and does not change the width of the decomposition. By an inductive argument for graphs $G_1, G_2, \ldots, G_t$ it holds that

$$\mathrm{mw}(G_1 \dot\cup G_2 \dot\cup \cdots \dot\cup G_t) \leq \max_{1\leq i\leq t}\{|G_i|\}.$$

As the designed polynomial equivalence relation assures that for all i and j the graphs $H_i \simeq H_j$ we write H to be the common graph for all instances. An AND-cross-composition of equivalent instances $(G_1, H), (G_2, H), \ldots, (G_t, H)$ we take as instance (G, H) – the disjoint union of all graphs. Because H is connected the new instance (G, H) admits a partition into copies of H if and only if all instances (G_i, H) admit such a partition. The previous paragraph shows that $\mathrm{mw}(G) \leq \max_{1\leq i\leq t}\{|G_i|\}$. This finishes the design of an AND-distillation and the proof of Theorem 58 for all graphs H for which the Partition into H problem is NP-hard.

5.3.3 Partition into H on graphs with bounded neighborhood diversity

In this section we will present a proof of Theorem 59. The proof is based on a bound of the number of possibilities of embedding a graph H into G.

Note that for the Partition into H problem parameterized by $\mathrm{nd}(G)$ it is possible that an embedding of a graph H in G does not have to obey the neighborhood diversity decomposition – it is possible that for example a clique type of H may be embedded among several clique (or even independent) types of G. For an example of such a situation see Figure 5.2. On the contrary the

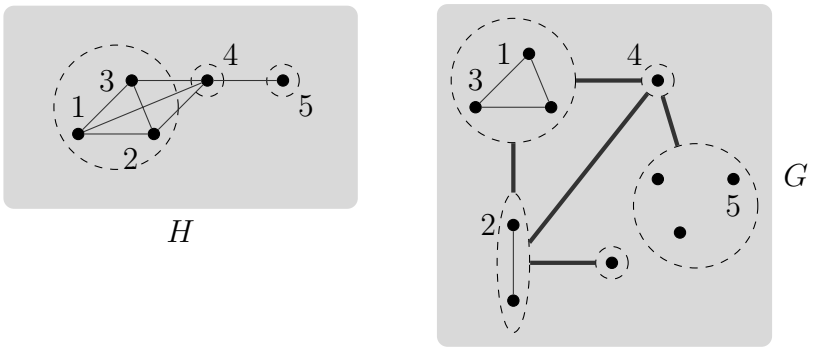

Figure 5.2: An example of embedding of a graph H in G – vertices are labeled $1, 2, \ldots, 5$ in both to show the embedding. The types in graphs are indicated by dashed circles around a group of vertices. Bold edges between types represent a complete bipartite graph.

embedding of H in G has to obey the neighborhood diversity decomposition of graph H. We formalize this in Lemma 62.

The proof of the following lemma is very similar to the proof of Lemma 60. The proof is omitted here (it is possible to find the proof in the Appendix).

Lemma 62. *Let G, H be graphs such that it is possible to partition G into copies of H. Then* $\mathrm{nd}(H) \leq \mathrm{nd}(G)$.

By Lemma 62 the algorithm for the Partition into H problem we may assume that $\mathrm{nd}(H) \leq \mathrm{nd}(G)$ and that H is a connected graph (we can take the complementary instance of the problem, see Section 5.3.2).

As the graph H is fixed in our setting the number $|H|$ is not a part of the input. Thus an embedding of H in G can be described by specifying a type to which a particular vertex of H is mapped (i.e. by a $\varphi\colon V(H) \to V(T_G)$). There are at most $\mathrm{nd}(G)^{|H|}$ possibilities of embedding H into G. It is not hard to see that it is possible to compactly describe an embedding of H in G by a vector $\mathbf{p} = (p_1, p_2, \ldots, p_{\mathrm{nd}(G)})$ of length $\mathrm{nd}(G)$, where p_i is the number of vertices used for this embedding inside the i-th type in the neighborhood diversity decomposition of graph G. In this case, we say that the embedding corresponds to vector $\mathbf{p}$. A vector $\mathbf{p} = (p_1, p_2, \ldots, p_{\mathrm{nd}(G)})$ is *admissible* if there exists an embedding of H in G which correspond to vector $\mathbf{p}$. Furthermore, note that it is possible to find the set $\mathcal{P}$ of all admissible vectors in time $\mathcal{O}(\mathrm{nd}(G)^{|H|}|H|^2)$ by searching through all embeddings and checking that edges of graph H are preserved.

The rest is to find a non-negative integral combination of vectors contained in set $\mathcal{P}$ that gives vector $(n_1, n_2, \ldots, n_{\mathrm{nd}(G)})$, where n_i is the number of vertices contained in the i-th type in the neighborhood diversity decomposition of graph G. It is straightforward to do this using integer linear programming (Lenstra's algorithm) as we are asking for non-negative integral combination of vectors in $\mathcal{P}$ of size bounded by $|H|$ and $\mathrm{nd}(G)$. This finishes the proof of Theorem 59.

5.4 Connected graph partition problem is MSO_2 definable

In this section we show that for a fixed connected graph H the Partition into H problem can be described by an MSO_2 formula. Even though this is not obvious at the first sight. The straightforward expression by a formula seems to operate with copies of H inside G – but the number of such copies is function of the number of vertices of G and thus cannot be bounded in terms of $\mathrm{tw}(G)$ and $|H|$.

MSO_2 formula idea We will proceed as follows. We will describe the solution to the Partition into H problem as a property of a set of edges which are in the solution. This approach is not expressible in MSO_1. We describe the property of being a solution to the Partition into H problem as "every connected part of the solution is an induced subgraph of G isomorphic to H". We express this by describing that a set of edges S is a solution if

- every vertex of the host graph G is in some connected component of $G[S]$,
- every connected component $G[S]$ is an induced graph (it contains exactly edges present in G between vertices of that particular component) and
- every connected component of $G[S]$ is a graph isomorphic to H (we do this by discovering a copy of H from a particular (fixed in advance) vertex in H).

It is clear that if every vertex of G is in some connected component which form an induced subgraph of G isomorphic to H, then we have the solution to the Partition into H problem. The full technical detail of the formula is noved to Section 5.5 due to space limitation.

MSO_1 inexpressibility Before we proceed to the formula defining the solution set S, we would like to give an evidence that it is not possible to express the Partition into H problem by an MSO_1 formula on an example of $H \simeq K_3$. Roughly speaking about the expressive power of MSO_1 logic it is impossible to distinguish between two large cliques [59] – namely for every MSO_1 formula φ there is a positive integer N such that it is impossible to distinguish two cliques K_N and K_{N+1}. If we build two graphs G_1, G_2 as $G_1 = K_{N-1} \cup K_{N+1}$ and $G_2 = K_N \cup K_N$ for large enough N that is divisible by 3 the framework of Ehrenfeucht–Fraïssé games [59] graphs G_1 and G_2 give that the Triangle Partition problem is not expressible in MSO_1 logic. It is possible to generalize this idea to other graphs as well.

5.5 Full details of MSO formulation

5.5.1 Identifying a vertex in H

For two distinct vertices u, v of the graph H, the distance $d_H(u, v)$ denotes the least length of an $u - v$-path in H. By a *diameter* of a graph $H = (W, F)$ we denote the number $\mathrm{diam}(H) = \max_{u,v \in W} d_H(u, v)$.

Let S be the assumed solution. For a fixed graph $H = (W, F)$, given a vertex u of a graph $G = (V, E)$ it is possible to find a particular vertex $v \in V$ (an

assumed copy of vertex $\bar{v} \in W$ chosen in advance) contained in a same connected component of S by an MSO_2 formula

$$\mathrm{Conn}(S,u,v) := (\exists e_1, e_2, \ldots, e_d \in S)(u \in e_1 \wedge \bigwedge_{i=1}^{d-1} e_i \cap e_{i+1} \neq \emptyset \wedge v \in e_d),$$

where $d = \mathrm{diam}(H)$. Note that the (sub)formula $\mathrm{Conn}(S,u,v)$ allows us do repeat edges and that we need at most $\mathrm{diam}(H)$ edges to find a desired vertex from any vertex inside the graph H.

5.5.2 Recognition of a copy of H from a particular vertex v

We now may assume that v is a vertex of H that was chosen in advance and fixed. We will proceed by identifying all vertices of a copy of H that contains v. Then we may inscribe that all edges and non-edges describing a copy of H are precisely selected in S. In the following free variable $U = \{u_1, u_2, \ldots, u_{|H|}\}, U \subseteq V$ stands for the preselected set of vertices in the assumed (tested) copy of H and v stands for the preselected vertex of H.

$$\mathrm{HCopy}(v,S,U) := (u_1 = v \wedge \bigwedge_{\{u_i,u_j\} \in F} \{u_i,u_j\} \in S \bigwedge_{\{u_i,u_j\} \notin F} \{u_i,u_j\} \notin S)$$

Here the trick is that we may assume a particular enumeration of vertices of H chosen in advance. We proceed to show that no other vertex of G is connected to those forming this particular copy of H inside the solution S.

$$\mathrm{ExactConn}(S,U) := (\forall e \in S\colon\ ((\exists x,y \in U\colon x \in e \wedge y \in e) \vee (\forall x \notin U\colon x \notin e)))$$

Finally it remains to check that every connected component of $G[S]$ forms an induced subgraph of G. Let U be the set of vertices of the connected component of S in G.

$$\mathrm{Ind}(S,U) := (\forall u,v \in U\colon (\{u,v\} \in S \Leftrightarrow \{u,v\} \in E))$$

The final formula for deciding whether graph $G = (V,E)$ admits a partition into copies of $H = (W,F)$ may be written as

$$\begin{aligned}\varphi_H(G) :=& (\exists S \subseteq E)\Big[\forall u \in V \exists v \in V\colon \Big((\mathrm{Conn}(S,u,v)) \wedge \\ & \wedge (\exists U \subseteq V\colon \mathrm{HCopy}(v,S,U)) \wedge (\mathrm{ExactConn}(S,U)) \wedge (\mathrm{Ind}(S,U)) \Big)\Big].\end{aligned}$$

This finishes the proof, as $\varphi_H(G)$ is an MSO_2 formula and so is testable in FPT time (for a fixed connected graph H) on graphs with bounded tree-width by Courcelle's theorem [17].

5.6 Conclusions

We have studied the Partition into H problem from the parameterized complexity point of view. We would like to enclose the paper with several open problems to which we would like to give a brief description.

We would like to ask a question related to the tree-depth, is there a disconnected graph H such that the Partition into H problem is FPT with respect to the tree-depth (of course, different from the edgeless graph on two vertices)?

We have presented in Section 5.3 an algorithm for a fixed graph H from a certain class of graphs. Is it possible to extend this result to a broader class of graphs H? Most important form this point of view seem graphs on 3 vertices – a path P_3 and a triangle K_3 (the rest of 3 vertex graphs would be resolved using complements).

Another important task in this area is to understand the boundary (viewed from the parameterized complexity perspective) between modular-width and neighborhood diversity, twin-cover and clique-width. We hope that our knowledge in this area can be extended in the highlight of the Partition into H problem – namely we should identify graphs H with the property that on one parameter the problem is fixed parameter tractable while it is W[1]-hard on some other parameter (higher in the parameter hierarchy).

Finally, our techniques from Lemma 62 showed that the Partition into H problem admits an FPT algorithm when the graph H is a prime graph even when the graph H is a part of the input. More generally, there is an FPT algorithm, if we extend the graph class by allowing constant number of vertices inside every type of the graph H. Is it possible to show an FPT algorithm parametrized by neighborhood diversity of graph G that takes the graph H as input?

Our results give possibilities for parametrized algorithms with respect to clique-width – namely for the class of prime graphs. Here we would like to propose a concrete question for the Partition into P_4 (the smallest prime graph) – is there an FPT algorithm for this problem with respect to clique-width?

Conclusion

There are many structural graph parameters used nowadays in theory. It is worth noting that we have discovered some problems that divide selected known structural graph parameters. We have successfully found examples of problems dividing tree-width and tree-depth, neighborhood diversity and vertex cover, and modular-width and neighborhood diversity. All these problem allow better understanding to parameters they divide.

Secondary purpose of this thesis is to promote some structural graph parameters that are not that much known as tree-width is. We have done this by showing how one can solve graph problems with respect to these parameterizations as well as showing hardness results.

One of the most crucial problems we can post here is to find more examples of these problems dividing parameters. We have to understand the sources of hardness of a particular problem as well as to the limitations of algorithmic design. In particular, it seems to be interesting to find a problem dividing twin-cover and vertex cover number. It is possible to find further and more specific open problems in end of each problem specific chapter.

Bibliography

[1] E. Ackerman, O. Ben-Zwi, and G. Wolfovitz. Combinatorial model and bounds for target set selection. *Theoretical Computer Science*, 411(44):4017 – 4022, 2010.

[2] J. Adámek and V. Koubek. Remarks on flows in network with short paths. *Commentationes mathematicae Universitatis Carolinae*, 12(4):661 – 667, 1971.

[3] G. Baier, T. Erlebach, A. Hall, E. Köhler, P. Kolman, O. Pangrác, H. Schilling, and M. Skutella. Length-bounded cuts and flows. *ACM Trans. Algorithms*, 7(1):4:1–4:27, Dec. 2010.

[4] O. Ben-Zwi, D. Hermelin, D. Lokshtanov, and I. Newman. Treewidth governs the complexity of target set selection. *Discrete Optimization*, 8(1):87 – 96, 2011. Parameterized Complexity of Discrete Optimization.

[5] 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):423 – 434, 2009.

[6] H. L. Bodlaender and F. V. Fomin. Equitable colorings of bounded treewidth graphs. *Theoretical Computer Science*, 349(1):22–30, 2005. Graph Colorings 2003.

[7] H. L. Bodlaender, T. Kloks, R. B. Tan, and J. van Leeuwen. λ-coloring of graphs. In H. Reichel and S. Tison, editors, *STACS*, volume 1770 of *Lecture Notes in Computer Science*, pages 395–406. Springer, 2000.

[8] F. T. Boesch and J. F. Gimpel. Covering points of a digraph with point-disjoint paths and its application to code optimization. *J. ACM*, 24(2):192–198, Apr. 1977.

[9] G. J. Chang and D. Kuo. The $L(2,1)$-labeling problem on graphs. *SIAM Journal on Discrete Mathematics*, 9(2):309–316, 1996.

[10] N. Chen. On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415, 2009.

[11] C.-Y. Chiang, L.-H. Huang, B.-J. Li, J. Wu, and H.-G. Yeh. Some results on the target set selection problem. *Journal of Combinatorial Optimization*, 25(4):702–715, 2013.

[12] M. Chopin, A. Nichterlein, R. Niedermeier, and M. Weller. Constant thresholds can make target set selection tractable. *Theory Comput. Syst.*, 55(1):61–83, 2014.

[13] F. Cicalese, G. Cordasco, L. Gargano, M. Milanič, J. Peters, and U. Vaccaro. Spread of influence in weighted networks under time and budget constraints. *Theoretical Computer Science*, 586:40–58, 2015.

[14] F. Cicalese, G. Cordasco, L. Gargano, M. Milanič, and U. Vaccaro. Latency-bounded target set selection in social networks. *Theoretical Computer Science*, 535:1–15, 2014.

[15] D. C. Corneil and U. Rotics. On the relationship between clique-width and treewidth. *SIAM J. Comput.*, 34(4):825–847, 2005.

[16] B. Courcelle. Graph rewriting: An algebraic and logic approach. *Handbook of Theoretical Computer Science*, pages 194–242, 1990.

[17] B. Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and Computation*, 85(1):12 – 75, 1990.

[18] B. Courcelle, J. Engelfriet, and G. Rozenberg. Handle-rewriting hypergraph grammars. *Journal of Computer and System Sciences*, 46(2):218–270, 1993.

[19] B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique width. In *WG 98*, pages 1–16, London, UK. Springer-Verlag.

[20] B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Appl. Math.*, 101(1–3):77–114, 2000.

[21] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

[22] G. Dahl and L. Gouveia. On the directed hop-constrained shortest path problem. *Operations Research Letters*, 32(1):15 – 22, 2004.

[23] R. Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[24] M. Dom, D. Lokshtanov, S. Saurabh, and Y. Villanger. Capacitated domination and covering: A parameterized perspective. In M. Grohe and R. Niedermeier, editors, *Parameterized and Exact Computation*, volume 5018 of *Lecture Notes in Computer Science*, pages 78–90. Springer Berlin Heidelberg, 2008.

[25] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66. ACM, 2001.

[26] P. Dvorak and D. Knop. Parametrized complexity of length-bounded cuts and multi-cuts. In R. Jain, S. Jain, and F. Stephan, editors, *Theory and Applications of Models of Computation - 12th Annual Conference, TAMC 2015, Singapore, May 18-20, 2015, Proceedings*, volume 9076 of *Lecture Notes in Computer Science*, pages 441–452. Springer, 2015.

[27] P. Dvorák, D. Knop, and T. Toufar. Target set selection in dense graph classes. *CoRR*, abs/1610.07530, 2016.

[28] J. Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[29] J. Fiala, T. Gavenciak, D. Knop, M. Koutecký, and J. Kratochvíl. Fixed parameter complexity of distance constrained labeling and uniform channel assignment problems - (extended abstract). In T. N. Dinh and M. T. Thai, editors, *Computing and Combinatorics - 22nd International Conference, COCOON 2016, Ho Chi Minh City, Vietnam, August 2-4, 2016, Proceedings*, volume 9797 of *Lecture Notes in Computer Science*, pages 67–78. Springer, 2016.

[30] J. Fiala, P. A. Golovach, and J. Kratochvíl. Distance constrained labelings of graphs of bounded treewidth. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 360–372. Springer, 2005.

[31] J. Fiala, P. A. Golovach, and J. Kratochvíl. Computational complexity of the distance constrained labeling problem for trees (extended abstract). In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *ICALP (1)*, volume 5125 of *Lecture Notes in Computer Science*, pages 294–305. Springer, 2008.

[32] J. Fiala, P. A. Golovach, and J. Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theor. Comput. Sci.*, 412(23):2513–2523, 2011.

[33] J. Fiala, J. Kratochvíl, and T. Kloks. Fixed-parameter complexity of λ-labelings. *Discrete Applied Mathematics*, 113(1):59–72, 2001.

[34] R. W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, June 1962.

[35] T. Fluschnik, D. Hermelin, A. Nichterlein, and R. Niedermeier. Fractals for kernelization lower bounds, with an application to length-bounded cut problems. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 25:1–25:14, 2016.

[36] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian journal of mathematics*, 8(3):399–404, 1956.

[37] A. Frank and E. Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.

[38] J. Gajarský, M. Lampis, and S. Ordyniak. Parameterized algorithms for modular-width. In *IPEC 2013, Revised Selected Papers*, pages 163–176, 2013.

[39] R. Ganian. Twin-cover: Beyond vertex cover in parameterized algorithmics. In *IPEC 2011, Revised Selected Papers*, pages 259–271, 2011.

[40] 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.

[41] P. A. Golovach and D. M. Thilikos. Paths of bounded length and their cuts: Parameterized complexity and algorithms. *Discrete Optimization*, 8(1):72 – 86, 2011.

[42] J. R. Griggs and R. K. Yeh. Labelling graphs with a condition at distance 2. *SIAM Journal on Discrete Mathematics*, 5(4):586–595, 1992.

[43] F. Gurski and E. Wanke. The NLC-width and clique-width for powers of graphs of bounded tree-width. *Discrete Applied Mathematics*, 157(4):583–595, 2009.

[44] G. Gutin, M. Jones, and M. Wahlström. Structural parameterizations of the mixed chinese postman problem. In N. Bansal and I. Finocchi, editors, *Algorithms – ESA 2015*, volume 9294 of *Lecture Notes in Computer Science*, pages 668–679. Springer Berlin Heidelberg, 2015.

[45] P. Hansen, A. Hertz, and J. Kuplinsky. Bounded vertex colorings of graphs. *Discrete Mathematics*, 111(1–3):305–312, 1993.

[46] A. Hope. Component placement through graph partitioning in computer-aided printed-wiring-board design. *Electronics Letters*, 8(4):87–88, February 1972.

[47] F. Hüffner, C. Komusiewicz, H. Moser, and R. Niedermeier. Fixed-parameter algorithms for cluster vertex deletion. *Theory Comput. Syst.*, 47(1):196–217, 2010.

[48] D. Huygens, M. Labbé, A. R. Mahjoub, and P. Pesneau. The two-edge connected hop-constrained network design problem: Valid inequalities and branch-and-cut. *Networks*, 49(1):116–133, 2007.

[49] A. Itai, Y. Perl, and Y. Shiloach. The complexity of finding maximum disjoint paths with length constraints. *Networks*, 12(3):277–286, 1982.

[50] M. Jarvis and B. Zhou. Bounded vertex coloring of trees. *Discrete Mathematics*, 232(1–3):145–151, 2001.

[51] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[52] D. G. Kirkpatrick and P. Hell. On the completeness of a generalized matching problem. In *STOC '78*, pages 240–245, New York, NY, USA, 1978. ACM.

[53] T. Kloks. *Treewidth, Computations and Approximations (Lecture notes in computer science, 842)*. Springer-Verlag New York, Inc., 1994.

[54] D. Knop. Partitioning graphs into induced subgraphs. In F. Drewes, C. Martín-Vide, and B. Truthe, editors, *Language and Automata Theory and Applications - 11th International Conference, LATA 2017, Umeå, Sweden, March 6-9, 2017, Proceedings*, volume 10168 of *Lecture Notes in Computer Science*, pages 338–350, 2017.

[55] D. Kobler and U. Rotics. Polynomial algorithms for partitioning problems on graphs with fixed clique-width (extended abstract). In *Symposium on Discrete algorithms, 12th SODA'01, Washington*, pages 468–476. ACM-SIAM, 2001.

[56] D. Král. The channel assignment problem with variable weights. *SIAM J. Discrete Math.*, 20(3):690–704, 2006.

[57] M. Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012.

[58] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

[59] L. Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[60] D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh. Faster parameterized algorithms using linear programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014.

[61] C. McDiarmid and B. Reed. Channel assignment on graphs of bounded treewidth. *Discrete Mathematics*, 273(1–3):183–192, 2003.

[62] J. Nesetril and P. O. de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012.

[63] J. Nešetřil and P. O. de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *European Journal of Combinatorics*, 27(6):1022 – 1041, 2006.

[64] A. Nichterlein, R. Niedermeier, J. Uhlmann, and M. Weller. On tractable cases of target set selection. *Social Netw. Analys. Mining*, 3(2):233–256, 2013.

[65] J. B. Orlin. Max flows in o(nm) time, or better. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 765–774, 2013.

[66] D. Peleg. Local majorities, coalitions and monopolies in graphs: A review. *Theor. Comput. Sci.*, 282(2):231–257, 2002.

[67] M. Rao. MSOL partitioning problems on graphs of bounded treewidth and clique-width. *Theor. Comput. Sci.*, 377(1-3):260–267, 2007.

[68] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 61–70, New York, NY, USA, 2002. ACM.

[69] N. Robertson and P. D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309 – 322, 1986.

[70] M. Tedder, D. G. Corneil, M. Habib, and C. Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In *ICALP 2008*, pages 634–645, 2008.

[71] W. T. Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, s1-22(2):107–111, 1947.

[72] R. van Bevern, R. Bredereck, L. Bulteau, J. Chen, V. Froese, R. Niedermeier, and G. J. Woeginger. Partitioning Perfect Graphs into Stars. *Journal of Graph Theory*, page n/a–n/a, 2016.

[73] M. Škvarek. The channel assignment problem for series-parallel graphs. Bachelor's thesis, Charles University, Prague, 2010. in Czech.

[74] E. Wanke. k-NLC graphs and polynomial algorithms. *Discrete Applied Mathematics*, 54(2–3):251–266, 1994.

[75] S. Warshall. A theorem on boolean matrices. *J. ACM*, 9(1):11–12, Jan. 1962.

List of Figures