FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	The Parameterized Complexity of Network Microaggregation
Student:	Bc. Jan Pokorný
Supervisor:	doc. RNDr. Dušan Knop, Ph.D.
Study program:	Informatics
Branch / specialization:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	until the end of summer semester 2023/2024

Instructions

Microaggregation is a classical statistical disclosure control technique that requires the input data to be partitioned into clusters while adhering to specified size constraints. The task is to design algorithms and/or hardness results for the clustering problems related to user-data anonymization.
The focus of the thesis should be on structural parameters such as treewidth, vertex cover, or related ones.
Additional parameters, e.g., cluster size or cluster diameter, might be used as additional parameters if necessary.

Electronically approved by doc. Ing. Jan Janoušek, Ph.D. on 16 January 2023 in Prague.

Master's thesis

THE PARAMETERIZED COMPLEXITY OF NETWORK MICROAGGREGATION

Bc. Jan Pokorný

Faculty of Information Technology
Department of Theoretical Computer Science
Supervisor: doc. RNDr. Dušan Knop, Ph.D.
June 29, 2023

Contents

List of Figures

List of Tables

I would like to thank my supervisor for arranging a research internship at TU Wien and helping me transition into the world of research. Then I would like to thank Robert Ganian for hosting me, and together with Dušan Knop for their guidance. Moreover, I would like to thank Dušan Knop, Robert Ganian, Václav Blažej, Šimon Schierreich, Kirill Simonov for their expertise, helpful insights and advice while working with them. Last but not least, I want to thank my family and my cat for their support not only during writing this thesis but throughout my whole studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis. I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Section 2373(2) of Act No. 89/2012 Coll., the Civil Code, as amended, I hereby grant a non-exclusive authorization (licence) to utilize this thesis, including all computer programs that are part of it or attached to it and all documentation thereof (hereinafter collectively referred to as the "Work"), to any and all persons who wish to use the Work. Such persons are entitled to use the Work in any manner that does not diminish the value of the Work and for any purpose (including use for profit). This authorisation is unlimited in time, territory and quantity

In Prague on June 29, 2023 .....

Abstract

Microaggregation is a classical statistical disclosure control technique that requires the input data to be partitioned into clusters while adhering to specified size and distance constraints. We explore the problem when the data points are embedded into a network that specifies the distances. Additionally, we consider the case when the clusters are connected. We use a combination of input parameters (maximum cluster size, maximum distance) and structural parameters (vertex cover, treewidth, treedepth) to achieve or rule out tractability in various settings. Namely, we propose a dynamic programming algorithm for the connected version. Further, we show FPT algorithms parameterized by vertex cover and maximum distance based on integer linear programming for both versions of the problem. On the other hand, we show that the connected version is W[1]-hard with respect to treewidth and maximum distance. For disconnected clusters, the problem is also W[1]-hard when parameterized by the vertex cover number. Lastly, we establish W[1]-hardness of the problem Equitable Connected Partition when parameterized by treedepth.

Keywords parameterized complexity, microaggregation, clustering, structural parametrization

Abstrakt

Mikroagregace je technika sloužící k rozdělení dat do shluků dle zadaných velikostních a vzdálenostní požadavků. V této práci se zabýváme případem, kdy data tvoří sít a dvěma variantami, zda shluky jsou souvislé nebo nejsou. Problém zkoumáme z hlediska kombinací parametrů danými problémem (maximální velikost shluků, maximální vzdálenost) a strukturálních parametrů (velikost vrcholového pokrytí, stromové šířky, stromové hloubky). Představujeme několik algoritmů, které ukazují, že problém lze efektivně řešit pomocí parametrizované složitosti. Jmenovitě představujeme pro souvislou verzi problému dynamické programování na stromovém rozkladu. Dále při parametrizaci pomocí velikosti vrcholového pokrytí a maximální vzdálenosti do centra ukazujeme FPT algoritmus na bázi celočíselného programování pro obě verze problému. Na druhé straně ukazujeme že problém je W[1] těžký při parametrizaci stromovou hloubkou a maximální vzdáleností v rámci souvislých shluků. Pokud jsou povoleny nesouvislé shluky, je problém W[1] těžký při parametrizaci velikostí vrcholového pokrytí. Nakonec ukazujeme, že problém Equitable Connected Partition je W[1] těžký vzhledem k parametru stromové hloubky.

Klíčová slova parametrizovaná složitost, mikroagregace, shlukování, strukturální parametrizace

Chapter 1
Introduction

In the current era, the amount of data and the need for them has increased tremendously. As much as the data might be useful for statistics and further research, it is essential not to invalidate the privacy of any individual when shared. Preservation of the individual's anonymity and relevancy of datasets is the main task of statistical disclosure control (SDC). One of the used metrics in SDC is the so called k-Anonymity. To achieve k-Anonymity no single record can be distinguished from at least $k-1$ other records when looking at identifying columns of the database. So any malicious actor should not be able to link the released records to an individual even when they have access to another database.

One particular approach used to achieve k-Anonymity is through microaggregation [1]. Microaggregation is a clustering method in which the records are partitioned into small clusters of size at least k based on their similarity. Then for each cluster a candidate that best represents the cluster is chosen. It was shown that with this method the loss of information is negligible for performance of many machine learning algorithms [2].

However, it is NP-hard to achieve k-Anonymity with microaggregation when the distances adhere into Euclidean space [3]. One of the directions to solve this problem was using heuristics [4] and approximations [5, 6]. Another approach is to change the way the instances are described and use graphs to encode closeness of the records. Both these approaches were utilized in [7] to produce clusters with lower-bound for their sizes. The graph structure of the graph can be used to produce faster algorithms or tighter approximations in reasonable time using parameterized complexity [8].

In the parameterized complexity paradigm, we are given a graph and a parameter k which captures some special structure of the graph. When considering both the size of the instance n and the parameter k, it enables us to construct faster algorithms than we could otherwise. The best outcome from parameterized complexity view is that our problem is *fixed parameter tractable* (FPT) which means it is solvable in time $f(k)n^c$ for some computable function f. Another desired outcome is to have XP algorithm running in time $n^{f(k)}$. To show that it is unlikely for a problem to admit an XP algorithm, it is enough to show that the problem is NP-hard for a constant value of the parameter k. Similarly, to rule out FPT algorithm, one can show the problem is W[1]-hard.

The precise problem we study is Network Microaggregation (NMA) where our goal is to partition a given edge-weighted graph into clusters and provide representatives for them. For each cluster we require its size to be between a lower-bound (ℓ) and upper-bound (u) to preserve privacy and together with the maximum distance d from cluster vertices to the cluster center to retain as much information. We also study a variation of the problem Connected Network Microaggregation (CNMA) where the clusters are connected. The problem (Connected) Network Microaggregation was first defined in [9]. This thesis expands the article, it provides further details and continues the study of the problem.

In particular, we study the problem exactly from the viewpoint of parameterized complexity. We propose two types of algorithms in Chapter 2 while focusing on two structural parameters – vertex cover number (vc) and treewidth (tw) because without structural parameters the problem remains NP-hard. With a dynamic programming on tree decomposition, we show an FPT algorithm for CNMA when parameterized by tw, u, and d. The same algorithm also shows that the problem belongs to XP when parameterized only by tw. The second type of algorithms is based on exhaustive branching and reduction to ILP and achieves FPT algorithm for both NMA and CNMA when parameterized by vc and d.

In Chapter 3, we supplement shown algorithms with appropriate hardness results. We show that Connected Network Microaggregation is W[1]-hard with respect to tw $+d$ which pairs tightly with the previously shown dynamic programming. Then we show W[1]-hardness for Network Microaggregation with respect to vc by a reduction from Multicolored Independent Set. In the last chapter, we focus on a different problem called Equitable Connected Partition (ECP). In ECP the task is to partition a given graph into connected parts that differ in size by at most 1. We establish W[1]-hardness of the problem with respect to treedepth and then use it to establish W[1]-hardness of Connected Network Microaggregation when parameterized by treedepth.

1.1 Notation

The set of natural numbers including zero is denoted $\mathbb{N}$. By $[n]$, where $n \in \mathbb{N}$, we denote the set of positive integers up to n. Let A be a set and $\mathcal{A} := \{A_1, A_2, \ldots, A_n\}$ be a set of subsets of A. The set $\mathcal{A}$ is a *partition* of A if $A := \bigcup_{i \in [n]} A_i$ and no two elements of $\mathcal{A}$ have a nonempty intersection. We also say that $\mathcal{A}$ *partitions* A.

1.2 Graph theory

A *graph* G is an ordered pair of sets (V, E) such that each element of E is a two element subset of V. We call V *vertices* and E *edges* of graph G and refer to them also as $V(G)$ and $E(G)$, respectively. Two vertices u, v are *adjacent* if $\{u, v\} \in E$, we also say that u is a *neighbor* of v. The set of all vertices adjacent to a vertex v is called the *neighborhood* of v and is denoted as $N(v)$. A vertex with only one neighbor is called a *leaf*. Let $G = (V, E), G' = (V', E')$ be graphs, if $V' \subseteq V$ and $E' \subseteq E$, then G' is a *subgraph* of G. Furthermore, if every edge $\{u, v\} \in E$ such that $u, v \in V'$ is in E', we call G' *induced subgraph* of G and denote it as $G' := G[V']$.

A *path* $P_k = (V, E)$ is a graph such that V has k distinct vertices $\{v_1, v_2, \ldots, v_k\}$ and $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{k-1}, v_k\}\}$. We also call it a *path from* $v1$ *to* v_k. The *length* of P_k is $k - 1$. The *distance* $\mathrm{dist}(u, v)$ of two vertices u, v in graph G is the length of a shortest path from u to v. If there is no such path, we set $\mathrm{dist}(u, v) := \infty$. A graph G is *connected* if for each pair of vertices $u, v \in V(G)$ there is a path from u to v, otherwise, we call it *disconnected*. Let $P_k = (V, E), k > 2$ be a path from u to v, graph $C_k := (V, E \cup \{\{u, v\}\})$ is a *cycle*. Graph that does not contain a cycle as a subgraph is a *forest*. If it is also connected, it is a *tree*. A tree with one vertex designated as a *root* is a *rooted tree*. The *height* of a rooted tree is the length of a longest path from the root to a leaf plus one. A *rooted forest* is a union of rooted trees and its height is the maximum height of one of its trees. A graph that does not contain any edge is an *independent set*.

Let $G = (V, E)$ be a graph and let $X \subseteq V \cup E$. If removing elements of X from G makes G disconnected, then X is a *separator* of G.

Computational Complexity

Our goal is to find a fast algorithm or find arguments that there does not exist. For that, we need a way of measuring how fast an algorithm is. The function $T\colon \mathbb{N} \to \mathbb{N}$ that computes the maximum number of steps that the algorithm spends on input of size n might not capture this notion well. There might exist two algorithms such that one spends fewer steps than the other depending on how exactly we set what the elemental operations are. Furthermore, it is not that important for us to study the speed of an algorithm with small, constant-sized, inputs as we can precompute the results for the finitely many cases and then answer immediately. Even a constant factor in the speed of the algorithm is not that interesting because we can simply buy a better computational machine to match the faster algorithm. The notion of the speed of an algorithm we are looking for is the asymptotic complexity.

▶ **Definition 1** (Bih-oh notation, Definition 0.2 in [10])**.** If f, g are two functions from $\mathbb{N}$ to $\mathbb{N}$, then we say that:

- $f = \mathcal{O}(g)$ if there exists a constant c such that $f(n) \leq c \cdot g(n)$ for every sufficiently large n,
- $f = \Omega(g)$ if $g = \mathcal{O}(f)$,
- $f = \Theta(g)$ if $f = \mathcal{O}(g)$ and $g = \mathcal{O}(f)$,
- $f = o(g)$ if for every $\varepsilon > 0\colon f(n) \leq \varepsilon \cdot g(n)$ for every sufficiently large n, and
- $f = \omega(g)$ if $g = o(f)$.

To emphasize the input parameter, we often write $f(n) = \mathcal{O}(g(n))$ instead of $f = \mathcal{O}(g)$, and use similar notation for $o, \omega, \Omega, \Theta$.

To understand what we mean when we say an algorithm, we define the computational model on which the algorithms are realized. We have chosen to use Turing machine (first defined by Turing [11]) as our computational model, however, this choice is arbitrary, and we could have built the same notion of algorithms and complexity upon other models of computation such as Lambda calculus as its expressive power is the same [12]. Later, we don't describe an algorithm this precisely but still with the following formalization in mind.

Intuitively, a Turing machine is a simplification of a computer that has k tapes each with an infinite amount of cells containing a symbol and a head that looks onto a single cell on each tape. The first tape serves as an input to the machine, the last as an output and the remaining tapes are for computing. The algorithms in the Turing machine is implemented as a set of states which dictate what to do in each step of the computation if the head sees certain cells on the tapes. More formally, the Turing machine is defined as follows.

▶ **Definition 2** (The Turing machine, Section 1.2 in [10])**.** A *Turing machine* (TM) M is a tuple (Σ, Q, δ) containing:

- A finite set Σ of the symbols that M's tapes can contain. The set Σ contains a designated "blank" symbol, denoted $\square$; a designated "start" symbol, denoted $\triangleright$; and the numbers 0 and 1. We call Σ the *alphabet* of M.
- A finite set Q of possible states M's register can be in. We assume that Q contains a designated start state, denoted q_{start}, and a designated halting state, denoted q_{halt}.
- A function $\delta\colon Q \times \Sigma^k \to Q - q_{\text{halt}} \times \Sigma^{k-1} \times \{\leftarrow, \bullet, \rightarrow\}^k$, where $k \geq 2$, describing the rules M use in performing each step. This function is called the *transition function* of M.

The computation starts with M in state q_{start}, each tape has $\triangleright$ in the first cell and the rest is filled with $\square$ except for the input tape, which has the input x written after $\triangleright$ as a finite string without blank symbols. This is the starting configuration of M with input x. The machine goes from one configuration to another with a step.

Let $(a_1, a_2, \ldots, a_k) \in \Sigma^k$ be the symbols under the head on each tape and $q \in Q$ be a state of the machine M. The *step* of a machine is applying the result of $\delta(q, (a_1, a_2, \ldots, a_k)) := (q', (a'_2, a'_3, \ldots, a'_k), (z_1, z_2, \ldots, z_k))$ as changing the state of the machine to q', writing a'_i to the cell under i-th head and moving the head according to z_i.

Once the machine reaches the state q_{halt} it *halts* and the non-blank string y written on the output tape is the answer for input x. We write that as $M(x) = y$.

▶ **Definition 3** (Computing a function and running time, Definition 1.3 in [10])**.** Let $f\colon \{0,1\}^* \to \{0,1\}^*$ and let $T\colon \mathbb{N} \to \mathbb{N}$ be some functions, and let M be a Turing machine. We say that M *computes* f if for every $x \in \{0,1\}^*$, whenever M is initialized to the start configuration on input x, then it halts with $f(x)$ written on its output tape. We say M *computes* f *in* $T(n)$*-time* if its computation on every input x requires at most $T(|x|)$ steps.

Furthermore, we say that a TM *decides* a language $L \subseteq 0,1^*$ if it computes the function $f_L\colon 0,1^* \to \{0,1\}$, where $f_L(x) = 1 \iff x \in L$.

In this thesis, we are not computing the solution to a problem, but instead, we formulate the problems as decision problems. By simplifying the possible answers to only Yes and No , or 1 and 0, we are able to use many of the tools of complexity theory without sacrificing much expressiveness. Every algorithm presented here can be easily modified to output the solution and the lower bounds hold as outputting the solution is not easier than saying whether a solution exists.

What we strive to do, is to find efficiently solvable algorithms. It might seem clear at first whether an algorithm running in $\mathcal{O}(n^4)$ or $\mathcal{O}(n^3)$ should be called efficient. But from a broad point of view, we want to preserve the property of combining two efficient algorithms is still an efficient algorithm even if we run one inside the other. This is well captured by the algorithms running in polynomial time, but not with a fixed constant with an exponent.

▶ **Definition 4** (The class P, Definition 1.12 and 1.13 in [10])**.** Let $T\colon \mathbb{N} \to \mathbb{N}$ be a function. A language L is in $\mathsf{DTIME}(T(n))$ iff there is a Turing machine that runs in time $c \cdot T(n)$ for some constant $c > 0$ and decides L. Then, the class P is defined as $\bigcup_{c \geq 1} \mathsf{DTIME}(n^c)$.

For some problems, we do not know whether we are able to solve them efficiently, but we might know whether we can verify that the solution exists efficiently.

▶ **Definition 5** (The class NP, Definition 2.1 in [10])**.** A language $L \subseteq \{0,1\}^*$ is in NP if there exists a polynomial $p\colon \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM M (called the *verifier* for L) such that for every $x \in \{0,1\}^*$:

$$x \in L \iff \exists u \in \{0,1\}^{p(|x|)} \text{ s.t. } M(x,u) = 1.$$

If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $M(x,u) = 1$, then we call u a *certificate* for x (with respect to the language L and Turing machine M).

We can easily observe, that every problem in P is also in NPas the certificate can be the answer to the problem and the verifier the TM that shows the problem is in P. However, the other direction of set inclusion is still not known and is one of the fundamental open problems in computer science. It is widely believed, that $\mathsf{P} \neq \mathsf{NP}$. The usual approach to show that a polynomial-time algorithm for a certain problem is highly unlikely is to show that if this problem allows a polynomial-time algorithm, so does every problem in NPand thus $\mathsf{P} = \mathsf{NP}$. This process is called a reduction with which we are able to show that one problem is at least as hard as the other as defined now.

▶ **Definition 6** (Reductions, NP-hardness and NP-completeness, Definition 2.7 in [10])**.** A language $L \subseteq \{0,1\}^*$ is *polynomial-time reducible* to a language $L' \subseteq \{0,1\}^*$, denoted by $L \leq_p L'$, if there is a polynomial-time computable function $f\colon \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*\colon x \in L$ if and only if $f(x) \in L'$. We say that L' is NP-hard if $L \leq_p L'$ for every $L \in \mathsf{NP}$. We say that L' is NP-complete if L' is NP-hard and $L \in \mathsf{NP}$.

The set of NP-complete problems is non-empty as Cook showed by reducing every problem in NPto SAT [13]. But we do not have to show a problem is NP-hard by definition how Cook did. Now we are able to reduce just from a single NP-hard problem because the reduction is transitive. Many of the essential problems such as Vertex Cover or Integer Linear Programming were first shown to be NP-complete by Karp [14] and Definition 6 is also called Karp reduction.

As said earlier, NP-hard problems most likely don't have a polynomial-time algorithm, to solve them we turn to another paradigm, the parameterized complexity.

1.3 Parameterized Complexity

The main goal of parameterized complexity is to provide a theoretical framework for understanding the complexity of problems that are difficult in the worst case but may have efficient algorithms when we have extra information about the problem. The extra property of the problem is captured by a parameter k that is assumed to be small in relevant instances of our problem. This parameter is given to us alongside the description of the instance.

▶ **Definition 7** (Parameterized problem, Definition 1.1 in [15])**.** A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed, finite alphabet. For an instance $(x,k) \in \Sigma^* \times \mathbb{N}, k$ is called the *parameter*.

To generalize the parameterized problem to multiple parameters $k_1, k_2, \ldots, k_r$, one can simply create a new parameter as the sum $\sum_{i=1}^{r} k_i$.

The parameterized complexity further classifies problems into the following classes. The most favorable outcome we can hope for with a NP-complete problem is that it admits an FPT algorithm.

▶ **Definition 8** (FPT, Definition 1.2 in [15])**.** A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called *fixed-parameter tractable* (FPT) if there exists an algorithm $\mathcal{A}$ (called a *fixed-parameter algorithm*), a computable function $f\colon \mathbb{N} \to \mathbb{N}$, and a constant c such that, given $(x,k) \in \Sigma^* \times \mathbb{N}$, the algorithm $\mathcal{A}$ correctly decides whether $(x,k) \in L$ in time bounded by $f(k) \cdot |(x,k)|c$. The complexity class containing all fixed-parameter tractable problems is called FPT.

Another positive but less favorable outcome is to classify that our problem belongs to the class XP.

▶ **Definition 9** (XP, Definition 1.3 in [15])**.** A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called *slice-wise polynomial* (XP) if there exists an algorithm $\mathcal{A}$ and two computable functions $f, g\colon N \to N$ such that, given $(x,k) \in \Sigma^* \times \mathbb{N}$, the algorithm A correctly decides whether $(x,k) \in L$ in time bounded by $f(k) \cdot |(x,k)|g(k)$. The complexity class containing all slice-wise polynomial problems is called XP.

Similarly, as with NP-hard problems, we are able to say that a fast algorithm is highly unlikely by using reductions. The difference between Karp reduction and parameterized reduction is that it can run in an FPT-time and that we have to bound our parameter by a function of a parameter from the problem we are reducing from.

▶ **Definition 10** (Parameterized reduction, Definition 13.1 (Parameterized reduction in [15]))**.** Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ 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 (x',k') of B such that

1. (x, k) is a yes-instance of A if and only if (x', k') is a yes-instance of B,
2. $k' \leq g(k)$ for some computable function g, and
3. the running time is $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable function f.

As with polynomial reduction, the parameterized reduction has a transitive property, so we can reduce from any problem that is hard for a given class to establish that our problem is also hard. The problems that most likely do not have an FPT algorithm are by the hard problems in the W[1] hierarchy. Usually, we do not need to know the exact class of the W-hierarchy that our problem belongs to, rather we are interested in whether the problem is at least W[1]-hard. For that, we can reduce from the problem Clique as it is W-complete [16]. Problems that are W[k]-hard for any $k > 0$ still might belong to XP. To also exclude XP algorithm we have to show that our problem is paraNP -hard i.e. for each constant value of the parameter k the problem is NP-hard. The class paraNP has the same relation to FPT as NPto P– FPT = paraNP only if P = NP [17, Corollary 2.13].

Therefore, there are 3 possibilities of characterization we want to show. Either the problem (1) has FPT algorithm, (2) has an XP algorithm and is W[1]-hard or (3) is paraNP -hard.

1.3.1 Treewidth

One of the most used parameters to obtain an efficient algorithm is the treewidth. From a high-level perspective, it captures how much the graph is similar to a tree. Many algorithms that work on trees can be generalized to work on graphs.

▶ **Definition 11** (Tree decompositions in [15])**.** A *tree decomposition* of a graph G is a pair $\mathcal{T} = (T, \{X^t\}_{t \in V(T)})$, where T is a tree whose every node t is assigned a vertex subset $X^t \subseteq V(G)$, called a *bag*, such that the following three conditions hold:

1. $\bigcup_{t \in V(T)} X^t = V(G)$. In other words, every vertex of G is in at least one bag.
2. For every $uv \in E(G)$, there exists a node t of T such that bag X^t contains both u and v.
3. For every $u \in V(G)$, the set $T^u = \{t \in V(T) \mid u \in X^t\}$, i.e., the set of nodes whose corresponding bags contain u, induces a connected subtree of T.

The *width* of tree decomposition $\mathcal{T} = (T, \{X^t\}_{t \in V(T)})$ equals $\max_{t \in V(T)} |X^t| - 1$, that is, the maximum size of its bag minus 1. The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the minimum possible width of a tree decomposition of G.

To avoid confusion, we call the vertices of T *nodes*.

One of the properties of the treewidth decomposition we use to create faster algorithms is that the intersection of bags of two neighboring nodes is a separator.

▶ **Lemma 1** (Lemma 7.3. in [15])**.** *Let $(T, \{X^t\}_{t \in V(T)})$ be a tree decomposition of a graph G and let ab be an edge of T. The forest $T - ab$ obtained from T by deleting edge ab consists of two connected components T_a (containing a) and T_b (containing b). Let $A = \bigcup_{t \in V(T_a)} X^t$ and $B = \bigcup_{t \in V(T_b)} X^t$. Then $X^a \cap X^b$ is a* separator*.*

To be able to design simpler algorithms, we will be using a nice tree decomposition.

▶ **Definition 12** (Nice tree decomposition in [15])**.** For a tree decomposition $\mathcal{T} = (T, \{X^t\}_{t \in V(T)})$ we distinguish one vertex r of T which will be the root of T. This introduces natural parent-child and ancestor-descendant relations in the tree T. We will say that such a rooted tree decomposition $\mathcal{T}$ is *nice* if the following conditions are satisfied:

- $X^r = \emptyset$ and $X^\ell = \emptyset$ for every leaf ℓ of T. In other words, all the leaves as well as the root contain empty bags.
- Every non-leaf node of T is of one of the following three types:
  - **Introduce node**: a node t with exactly one child t' such that $X^t = X^{t'} \cup \{v\}$ for some vertex $v \notin X^{t'}$; we say that v is *introduced* at t.
  - **Forget node**: a node t with exactly one child t' such that $X^t = X^{t'} \setminus \{w\}$ for some vertex $w \in X^{t'}$; we say that w is *forgotten* at t.
  - **Join node**: a node t with two children t_1, t_2 such that $X^t = X^{t_1} = X^{t_2}$.

The computation of a tree decomposition is in FPT with respect to the treewidth [18] even if we use the nice tree decomposition as shown in Lemma 2. Therefore, when we propose an FPT or slower algorithm, the algorithm's complexity class remains the same even when we do not get the tree decomposition on the input.

▶ **Lemma 2** (Lemma 7.4. in [15]). *If a graph G admits a tree decomposition of width at most k, then it also admits a nice tree decomposition of width at most k. Moreover, given a tree decomposition $\mathcal{T} = (T, \{X^t\}_{t \in V(T)})$ of G of width at most k, one can in time $\mathcal{O}(k^2 \cdot \max(|V(T)|, |V(G)|))$ compute a nice tree decomposition of G of width at most k that has at most $\mathcal{O}(k|V(G)|)$ nodes.*

1.3.2 Other structural parameters

Many parameters on graphs are derived from a fundamental graph problem such as Vertex Cover (VC). The parameter is the smallest integer k such that there exist a solution for VC of size k. We call the parameter *vertex cover number* or we write it as *vertex cover #*. The rest of the problems which can be easily transformed into a parameter, and we use them are described in Section 1.5.

Other parameters we use are pathwidth and treedepth. Pathwidth has the same definition as treewidth except the nodes of the decomposition form a path. Treedepth is defined as follows.

▶ **Definition 13** (Definition 6.1 in [19]). Let clos(F) of a rooted forest F be the graph with vertex set $V(F)$ and edge set $\{\{x, y\} : x \text{ is an ancestor of } y \text{ in F}, x \neq y\}$. The *treedepth* td (G) of a graph G is the minimum height of a rooted forest F such that $G \subseteq \text{clos}(F)$.

For some parameters A, B, if we show there is an FPT (or XP) algorithm parameterized by A we immediately also obtain FPT (or XP) algorithm parameterized by B if B bounds A. We say parameter B *bounds* parameter A if there exists a function f such that for each graph G it holds that $f(B(G)) > A(G)$. A small hierarchy of used parameters is in Figure 1.1. Vertex cover bounds 4-pvcn because if we remove the vertex cover, no edge is present and so there is no path on 4 vertices. The graph without paths of length more than k has treedepth at most k [19, Proposition 6.1] and so 4-pvcn bounds treedepth. The remaining relations are that treedepth bounds pathwidth, both vertex cover number and feedback edge set number bound feedback vertex set number and both feedback vertex set number and pathwidth bound treewidth [20].

1.4 Our problem

To model a network, we use a weighted graph where the weight of an edge is the distance between its incident vertices. We assume that the input network is connected, otherwise, it could be split into its connected components and solve each independently. Our problem is formally defined as follows:

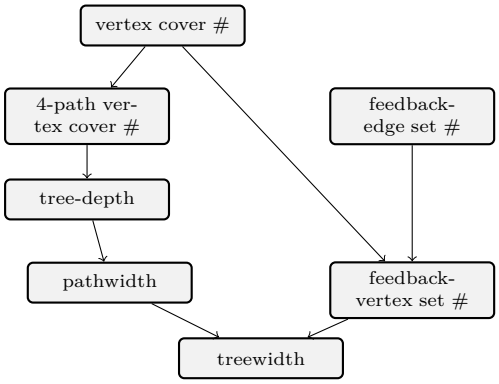

Figure 1.1 An overview of parameters used in this thesis. An arrow from parameter A to parameter B means that A bounds B. The relation is transitive, and so arrows that can be obtained from the transitivity are omitted.

	NMA		CNMA	
	tw	vc	tw	vc
—	NP-h	W[1]-h (Thm. 10), XP	W[1]-h, XP (Cor. 5)	W[1]-h, XP
d	W[1]-h, XP	FPT (Thm. 8)	W[1]-h (Thm. 9), XP	FPT (Thm. 7)
u	NP-h	FPT	W[1]-h, XP	FPT
$d+u$	W[1]-h, XP	FPT	FPT (Cor. 5)	FPT

Table 1.1 The full complexity-theoretic landscape of (Connected) Network Microaggregation under all combinations of considered input-specified parameters (rows) and structural parameters (columns); NP-h means that the problem remains NP-hard even for a fixed value of the parameters. Results that are marked blue are results obtained in this thesis. A blue result without a linked theorem or corollary is trivially derived from other results. The rest of the results are part of the article [9].

Network Microaggregation (NMA)

Input: An undirected n-vertex graph $G = (V, E)$, a lower-bound $\ell \in \mathbb{N}$, an upper-bound $u \in \mathbb{N}$, a maximum allowed distance to a cluster center $d \in \mathbb{N}$ and a length function $\omega\colon E \to [d]$.

Question: Is there an integer m and a partition $\Pi = (C_1, \ldots, C_m)$ of V together with a list of vertices $\mathcal{C} = (c_1, \ldots, c_m)$ such that $\forall i \in [m]\colon \ell \leq |C_i| \leq u$, $\forall v \in C_i\colon \operatorname{dist}_G(v, c_i) \leq d$?

The Connected Network Microaggregation problem (CNMA) is defined analogously, but with the additional requirements that the subgraph induced by each cluster $C \in \Pi$ is connected. We mention that while (C)NMA are formulated as decision problems for complexity-theoretic reasons, all our algorithms are constructive and can also output a microaggregation Π as a witness.

In Table 1.1 we can see the overview of all the results of both CNMA and NMA, when parameterized by the vertex cover number and treewidth of the graph in combination with the maximum distance d to the cluster center and upper-bound u on the cluster size. We have to use the structural parameters because without them the problem is NP-hard even when u and d are constant as we show now.

P_3-Partitioning	
Input:	A graph $G = (V, E)$.
Question:	Can G be partitioned into vertex disjoint subgraphs $G_1, G_2, \ldots, G_{\frac{\lvert V \rvert}{3}}$ such that each G_i is isomorphic to P_3.

The problem P_3-Partitioning is a special case of the problem Generalized Matching Problem and was proven to be NP-hard in [21]. We use it as a starting point of a simple reduction to prove the hardness of Connected Network Microaggregation.

▶ **Claim 1.** *Connected Network Microaggregation is* paraNP *hard even if* $d = 1$ *and* $u = 3$.

Proof. The reduction is just using the same graph on the input and setting $d = 1$ and $\ell = u = 3$, and setting the weight $\omega(e) := 1$ for each edge $e \in E$. If the Connected Network Microaggregation instance has a solution then every cluster C_i has a cluster center c_i which is neighboring both of the other 2 vertices of the cluster, and they induce either P_3 or C_3 which contains P_3 as a subgraph. Therefore, P_3-Partitioning has a solution as well. Conversely, each P_3 in the solution of P_3-Partitioning is also a connected cluster of size 3 and the cluster center is the inner vertex of the path. ■

1.5 Compedium of problems

We include a small compendium of computational problems used in this thesis either as a problem we are reducing from, reducing to or a parameter is derived from it.

Equitable Connected Partition (ECP)	
Input:	A simple undirected and connected n-vertex graph $G = (V, E)$ and a positive integer $p \in \mathbb{N}$.
Question:	Is there a partition $\pi = \{V_1, \ldots, V_p\}$ of V such that $\forall i \in [p]\colon G[V_i]$ is connected and $\lvert\lvert V_i \rvert - \lvert V_j \rvert\rvert \leq 1$ for every pair of $i, j \in [p]$?

Vertex Cover (VC)	
Input:	A graph $G = (V, E)$ and an integer k.
Question:	Is there a set $X \subset V$ such that $\lvert X \rvert \leq k$ and $G - X$ is an independent set?

d-Path Vertex Cover (d-PVC)	
Input:	A graph $G = (V, E)$ and an integer k.
Question:	Is there a set $X \subset V$ such that $\lvert X \rvert \leq k$ and $G - X$ does not contain P_d as a subgraph?

Feedback Vertex Set (FVS)	
Input:	A graph $G = (V, E)$ and an integer k.
Question:	Is there a set $X \subset V$ such that $\lvert X \rvert \leq k$ and $G - X$ is a forest?

Feedback Edge Set (FES)	
Input:	A graph $G = (V, E)$ and an integer k.
Question:	Is there a set $X \subset E$ such that $\lvert X \rvert \leq k$ and $G - X$ is a forest?

Independent set	
Input:	A graph $G = (V, E)$ and an integer k.
Question:	Has G independent set as a subgraph of size at least k?

Multicolored Independent set (MIS)	
Input:	A graph $G = (V, E)$, an integer k and a coloring $c\colon V \to [k]$.
Question:	Has G independent set I as a subgraph such that $\{c(v) \mid v \in I\} = k$?

Integer Linear Programming (ILP)	
Input:	Integers n, m, a matrix $A \in \mathbb{N}^{n,m}$ a vector $b \in \mathbb{N}^m$.
Question:	Is there a vector $x \in \mathbb{N}^m$ such that $Ax \leq b$?

Integer Linear Programming is usually defined with extra vector $c \in \mathbb{N}^p$, and we want to maximize $c \cdot x$. We call one row $(Ax)_i \leq b_i$ a *condition* and one column of A a *variable* when referring to an ILP.

Chapter 2
Algorithms

Our goal is to establish whether the problems can be solved efficiently – has an FPT or XP algorithm. For that we consider two structural parameters, the vertex cover number and treewidth of a graph as we already showed without them the problem is NP-hard in Claim 1 even if the cluster size u and the maximum distance to the center d are constants. Primarily, we focus on obtaining an algorithm parametrized by treewidth as the result immediately carries over to vertex cover.

2.1 Dynamic programming on tree decomposition

In this section, we provide a dynamic programming algorithm for CNMA that establishes the tractability treewidth is used as a parameter. We remark that while the use of dynamic programming that relies on bags acting as separators in the graph is the "golden standard" for treewidth-based algorithms, the technical details here are far from standard. Among others, to obtain dynamic programming tables which are succinct enough for our purposes, we needed to identify a suitable notion of vertex types that capture their properties when used as a center and incorporate these into the tables.

As the types used in this section will crucially depend on vertex distances, it will be useful to introduce some additional terminology to capture this. As we will not need to distinguish distances longer than d, let $\mathrm{dist}_d(v,w) = \mathrm{dist}(v,w)$ if $\mathrm{dist}(v,w) \leq d$ and ∞ otherwise. Our operations over distances are additive, assuming the result is set to ∞ whenever the value exceeds d.

For each node x in a tree decomposition, we can now partition the vertices of G into *center-types* that are based on their membership to V^x and their distances to X^x. Formally, let the *center-type* of a vertex v for $v \in V(G)$ be

$$t^{x,v} = (\mathrm{dist}_d(v,w_1), \mathrm{dist}_d(v,w_2), \ldots, \mathrm{dist}_d(v,w_{|X^x|})),$$

where w_i are the vertices of X^x. A core ingredient for the proof is that if two vertices have the same type and are both in the past or both in the future, then they are "indistinguishable" from the viewpoint of the bag.

Let $T^x \subseteq ([d] \cup \{0,\infty\})^{|X^x|}$ be the set of all *center-types* capturing all possible distances a center could have to the bag vertices. We further distinguish the set of center-types that are in the past as T^x_{T} and in the future as T^x_{F}.

The set T^x can be computed by running Dijkstra's algorithm from each vertex in the bag X^x. The complexity of finding T^x is only $\mathcal{O}(\mathrm{tw}(\mathrm{tw}\, n + n \log n))$ since there are at most $\mathrm{tw}\, n$ edges in G [18] and Dijkstra's algorithm runs in $\mathcal{O}(|E| + |V| \log |V|)$ time [22]. There are $\mathcal{O}(n\, \mathrm{tw})$ nodes

in the tree decomposition of G [18]. Therefore, the calculation of the center-types for every node x can be done in $\mathcal{O}(n^2\,\mathrm{tw}^2(\mathrm{tw}+\log n))$.

▶ **Definition 14.** Let us have two bags x and y and a vertex v such that $X^y \setminus X^z = v$. Let $w \in V$ be a vertex on the other side of the bag as v. We call $\mathrm{dist}_d(v,w)$ for any $w \in B$ *uniquely determined*. We denote the center-type with the uniquely determined distance from v as $t^{y,\mathrm{unq}(w)} = (t^{z,w}_{v_1}, t^{z,w}_{v_2}, \ldots, \mathrm{dist}_d(v,w))$, where $\forall i \in [|X^z|] : v_i \in X^z$. Moreover, we call $t^{y,\mathrm{unq}(w)}$ as *uniquely determined*.

▶ **Observation 3.** *The uniquely determined distance* $\mathrm{dist}_d(v,w) = \min_{v' \in X^z} \mathrm{dist}_d(v,v') + \mathrm{dist}_d(v',w)$ *because* X^z *separates* v *from* w.

For a bag x let us call a cluster C *open* when $C \subseteq V^x$ and *closed* when $C \subseteq V^x \setminus X^x$. The intuition is that we are incrementally creating the clusters that together make the solution. Once they are closed, we do not change them, and they have to satisfy all conditions we require from the solution i.e.: their size has to be between ℓ and u, and there has to exist a center vertex which is at most d far from all cluster vertices. For an open cluster, we do not require the lower-bound on the size to be satisfied.

The presented algorithm in this section proceeds by leaf-to-root dynamic programming along a nice tree decomposition computed using well-known algorithms [23, 24] in FPT-time.

▶ **Theorem 4.** *The problem* Connected Network Microaggregation *can be solved in time* $\mathcal{O}(n^2\,\mathrm{tw}^2(\mathrm{tw}+\log n)) + d^{\mathcal{O}(\mathrm{tw}^2)}u^{\mathcal{O}(\mathrm{tw})}n$.

Proof. Since the clusters are connected and the bag is a separator, the number of open clusters is upper bounded by $\mathrm{tw}+1$. We remember the partition of the bag into open clusters and the current size of the clusters including already forgotten vertices. However, the cluster center doesn't have to be inside the cluster or neighboring a vertex from the cluster. It is sufficient to use just the center-type of the center. Therefore, for each open cluster, we guess whether its center is in the past or future and its center-type. The center-type has $(d+2)^{\mathrm{tw}+1}$ possibilities, and there are at most $\mathrm{tw}+1$ many open clusters, so we have at most $((d+2)^{\mathrm{tw}+1})^{\mathrm{tw}+1} = (d+2)^{(\mathrm{tw}+1)^2}$ different possibilities in total. For each bag, we compute a set of all existing center-types T^x both in the past and in the future. This way we know whether a center exists when we are closing the cluster or when we are creating a new cluster.

When a new vertex v is introduced, we guess its cluster and check if the distance to its center is at most d. Vertex v is in the bag now, so we know its distance from the center. We also consider that v can create new cluster C. In that case, we have to guess the center-type of the center of C. For future centers we don't know the distance from v to the center, we have to guess it as well. However, we can check the correctness of our guess in forget node, since then it is uniquely determined and the distance can be calculated using Observation 3. In forget nodes, we also verify that the cluster size is correct if we are closing the cluster. In join nodes, we are joining records that partition the bag into the same clusters with the same center-types. For the same clusters, we add the sizes together and check that center is in the future in at least one of the records.

To ensure that the clusters are connected, we further partition the clusters into connected groups. The idea of the groups is similar as in the dynamic programming for finding Steiner Tree on graphs with a small treewidth [15]. Two groups are merged in an introduce node if a vertex is added to the cluster that neighbors both of the groups. The groups are also merged in the join node if they have a nonempty intersection But the group does not split when a vertex is forgotten, the group remains connected through the past. We forbid forgetting a vertex if it is the last vertex of its group when the cluster still has more than one group. The number of groups in a bag is at most $\mathrm{tw}+1$ and since the number of groups is not lower than the number of open clusters we have at most $\mathrm{tw}^{\mathcal{O}(\mathrm{tw})}$ different ways to partition the bag into groups and clusters. We stress that it is not sufficient to partition the bag only into clusters and solve the

connectivity by merging the clusters because past vertices might be verified to different cluster centers which now may be incompatible.

The table for the dynamic programming is as follows:

$$D[x, \mathcal{P}, c, s, b] \in \{\mathsf{T}, \mathsf{F}\},$$

where

- x is a node of the nice tree decomposition $\mathcal{T}$,
- $\mathcal{P} = \{P_1, P_2, \ldots, P_{N^x}\}$ is a set of open clusters, where $P_i = \{\overline{P}_i^1, \overline{P}_i^2, \ldots \overline{P}_i^{n_i}\}$ is an open cluster,
- $c\colon \mathcal{P} \to T^x$ are the center-types of centers of clusters in $\mathcal{P}$,
- $s\colon \mathcal{P} \to [u]$ determines the size of the open clusters,
- $b\colon \mathcal{P} \to \{\mathsf{T}, \mathsf{F}\}$ says whether the center of each cluster is in the past (T) or the future (F).

The visualization of an entry can be seen in Figure 2.1.

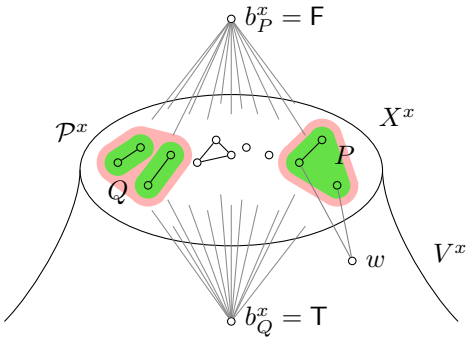

Figure 2.1 Example of an entry; red shows clusters, green groups; a future and past center is depicted; note how w ($w \in P$) causes a group to span the whole cluster P

To obtain a value tied to a specific cluster we use bottom indices, e.g., c_P^x is the center-type of an open cluster P. We later prove that the value of an entry in the table is T if and only if there exists a partition of V^x into closed and open clusters such that the open clusters are partitioned according to $\mathcal{P}$, c, s, and b.

The values in the table are computed in leaf to root fashion. We fix the entry arguments to $x, \mathcal{P}^x, c^x, s^x, b^x$ and compute the value for the entry in the following way.

- **Leaf node:**

  There are no vertices, therefore no open clusters and the solution is valid. We slightly abuse the notation here to say that c, s, b have an empty domain.

$$D[x, \emptyset, \emptyset, \emptyset, \emptyset] = \mathsf{T}$$

- **Introduce node:** Introduction of v, i.e., $X^x = X^y \cup \{v\}$.

  We divide the computation of the current entry into a few equations. First, we rule out the invalid table entries.

  The vertex v has to be within the distance d to the center of its cluster P_v. If the current arguments do not satisfy this, we mark the entry as invalid (F), that is,

$$D[x, \mathcal{P}^x, c^x, s^x, b^x] = \mathsf{F} \quad \text{if } c^x_{P_v, v} = \infty. \tag{2.1}$$

Since v is in the bag, c^x contains the distance from each open cluster center to it.

For each cluster $P \in \mathcal{P}^x$, a vertex of center-type c^x_P has to exist in the correct part of the graph determined by b^x_P. If it does not, we invalidate the entry, i.e.,

$$D[x, \mathcal{P}^x, c^x, s^x, b^x] = \mathsf{F} \quad \text{if } \exists P \in \mathcal{P}^x : c^x_P \notin T^x_{b^x_P}. \tag{2.2}$$

Now we subdivide the search for a previous T entry which is extended by the current entry. In each of the equations we bind only a subset of the arguments, but together it constructs an exact procedure for finding the previous entry.

The vertex v may either create a new cluster or join an existing one. In the first case, we consider that v creates its own open cluster P with a single group containing only v. Therefore, $\mathcal{P}^x$ differs from $\mathcal{P}^y$ by containing P. The other arguments c^x, s^x, b^x now contain a new entry of the cluster P. For the cluster sizes, it holds that $s^x_P := 1$, but the sizes of other clusters remain the same in s^y. We set the constraints for c^x and b^x later.

In the second case, v was added to an open cluster $P^x \in \mathcal{P}^x$, and it might have connected a few groups in P^x. Let $\mathcal{P}^*$ be the set of all the possible open clusters P^y in $\mathcal{P}^y$ such that adding the vertex v into them and merging the groups neighboring v creates the cluster P^x. Then the result of entry is computed by iterating over the set $\mathcal{P}^*$ and finding one suitable previous entry with open cluster $P^y \in \mathcal{P}^*$ and the size s^y satisfying $s^y_{P^y} = s^x_{P^x} - 1$ and for every other cluster $P \in \mathcal{P}^x \setminus P^x$ the size $s^y_P = s^x_P$ remains the same.

$$D[x, \mathcal{P}^x, c^x, s^x, b^x] = \bigvee_{P^y \in \mathcal{P}^*} D[y, (\mathcal{P}^x \setminus \{P^x\}) \cup \{P^y\}, c^y, s^y, b^y] \tag{2.3}$$

Adding the vertex v to the bag introduces new distance between v and the cluster centers in the center-types c^x. From c^x we obtain c^y by dropping the v coordinate of the center-type c^x_P for each cluster $P \in \mathcal{P}^x$ (except for the cluster of v if v is the only vertex in it). Then the shortest path from the center in the past through v has to go through the bag X^y. We can verify the distance to v from each past cluster from Observation 3 as it is uniquely determined. For each cluster $P \in \mathcal{P}^x$ with cluster center in the future, we already verified the distance to all its vertices and thus nothing has to be done.

$$D[x, \mathcal{P}^x, c^x, s^x, b^x] = D[y, \mathcal{P}^y, c^y, s^y, b^y] \wedge \\ \forall P \in \mathcal{P}^x : \neg b^x_P \vee c^x_{P,v} = \min_{w \in X^y} \{c^y_{P,w} + \mathrm{dist}_d(v, w)\} \tag{2.4}$$

If for some cluster P holds that $t^{v,=} c^x_P$ and $b^x_P = \mathsf{T}$, both $b^y_P = \mathsf{T}$ and $b^y_P = \mathsf{F}$ may be valid options as v may but doesn't have to be the cluster center for P. However, if $b^x_P = \mathsf{T}$ and $b^y_P = \mathsf{F}$, then v is the center of the cluster P. For a given partition $\mathcal{P}^x$, center-types c^x, and a boolean vector b^x of which centers are in the past, we create a set $B(\mathcal{P}^x, c^x, b^x)$ of all possible boolean vectors b^y determining if the cluster centers are in the past or in the future. More formally, let $B(\mathcal{P}^x, c^x, b^x) = \{b^y \in \{\mathsf{T}, \mathsf{F}\}^{|\mathcal{P}^y|} \mid \forall P \in \mathcal{P}^y \setminus \{\{v\}\} \colon b^y_P \implies b^x_P \wedge (b^x_P \wedge \neg b^y_P) \implies c^x_{P,w} = t^{x,v}\}$. We also have to reflect the changes v does to the cluster P and if v created its own cluster we have to skip it because it does not exist in $\mathcal{P}^y$ Then the dynamic programming table entry is computed as follows.

$$D[x, \mathcal{P}^x, c^x, s^x, b^x] = \bigvee_{b^y \in B(\mathcal{P}^x, c^x, b^x)} D[y, \mathcal{P}^y, c^y, s^y, b^y] \tag{2.5}$$

- **Forget node:** This is the last occurrence of a vertex v in the tree decomposition, i.e., $X^x = X^y \setminus \{v\}$. This node is symmetric to introduce node but simpler.

  We first describe how to find $\mathcal{P}^y, s^y$ and b^y, later we will focus on c^y. The partition $\mathcal{P}^y$ may differ from $\mathcal{P}^x$ only by having v inside one of the groups of an open cluster or v is the only vertex in an open cluster. The size of the clusters s^x and the flag b^x remain the same, except for the case v is the last vertex of an open cluster, where we additionally have to check its size. More formally, let $S(\mathcal{P}^x, s^x, P_v) = \{s^y \mid \ell \leq s^y_{P_v} \leq u \wedge \forall P \in \mathcal{P}^x \colon s^y_P = s^x_P\}$ be the set of possible sizes and $B(\mathcal{P}^x, b^x, P_v) = \{b^y \mid b^y_{P_v} \in \{\mathsf{T}, \mathsf{F}\} \wedge \forall P \in \mathcal{P}^x \colon b^x_P = b^y_P\}$ the set of possible boolean vectors of whether the center is in the past.

$$D[x, \mathcal{P}^x, c^x, s^x, b^x] = \bigvee_{\substack{b^y \in B(\mathcal{P}^x, b^x, P_v) \\ s^y \in S(\mathcal{P}^x, s^x, P_v)}} D[y, \mathcal{P}^x \cup \{P_v\}, c^y, s^y, b^y] \tag{2.6}$$

  Now we focus on the center-types c^x. As opposed to the introduction of v, here, the distances from every cluster center to v are being forgotten. We validate them by checking b^x whether v is on the other side of X^x than the centers, then the distance is uniquely determined. We define $C(\mathcal{P}^y, c^x, b^y) := \{c^y \mid \forall P \in \mathcal{P}^y : c^y_P \in T^y_{b^y_P} \wedge \forall w \in X^x : c^x_{P,w} = c^y_{P,w}\}$ as the set of all the possible extension of c^x by adding the v coordinate.

$$D[x, \mathcal{P}^x, c^x, s^x, b^x] = \bigvee_{c^y \in C(\mathcal{P}^y, c^x)} D[y, \mathcal{P}^y, c^y, s^y, b^y] \wedge \forall P \in \mathcal{P}^y : b^y_P \vee c^y_{P,v} = \min_{w \in X^y} \{c^y_{P,w} + \mathrm{dist}_d(v, w)\} \tag{2.7}$$

- **Join node:** In the join node, we are joining two nodes y, z that have the same partition of open clusters with the same center-types, but the group partition can be different. The group in V^x may be connected through a path in V^y and V^z as seen in Figure 2.2. Let $\mathbb{P}$ be the set of all possible pairs of partitions $(\mathcal{P}^y, \mathcal{P}^z)$ such that for each pair of groups $\overline{P}^y \in P^y, \overline{P}^z \in P^z$ from respective partitions holds two conditions. They contain the same vertices $\texttt{flat}(P^x) = \texttt{flat}(P^y) = \texttt{flat}(P^z)$ and if $\overline{P}^y \cap \overline{P}^z \neq \emptyset$ then $(\overline{P}^y \cup \overline{P}^z) \subseteq \overline{P}^x$, where $\overline{P}^x$ is a group in an open cluster $P^x \in \mathcal{P}^x$. Whether the cluster center is in the past or not may differ, but it is not possible for the center to be in the past in both children's entries. Let $B(\mathcal{P}^x, b^x) := \{(b^y, b^z) \mid \forall P \in \mathcal{P} : ((b^y_P \vee b^z_P) = b^x_P) \wedge (\neg b^y_P \vee \neg b^z_P)\}$ be the set of possible pairs of whether cluster centers are in the past or future that add up to b^x. If the center is in the future in both of the children's entries, the center-type has to exist in the set of future centers T^x_{F}, this is ensured with the extra condition in Equation (2.8). The sizes of the clusters in nodes y, z have to add up to the size of x. Let $S(\mathcal{P}^x, s^x) := \{(s^y, s^z) \mid \forall P \in \mathcal{P}^x : s^y_P + s^z_P - |P \cap X^x| = s^x_P\}$ be the set of all possible pairs of cluster sizes in nodes y and z.

  To compute the value of an entry in the table, we iterate through the sets $\mathbb{P}, S$ and B until we find two compatible entries that are both true.

$$D[x, \mathcal{P}^x, c^x, s^x, b^x] = \bigvee_{\substack{(\mathcal{P}^y, \mathcal{P}^z) \in \mathbb{P}(\mathcal{P}^x) \\ (s^y, s^z) \in S(s^x, \mathcal{P}^x) \\ (b^y, b^z) \in B(b^x, \mathcal{P}^x)}} D[y, \mathcal{P}^y, c^x, s^y, b^y] \wedge D[z, \mathcal{P}^z, c^x, s^z, b^z] \wedge \forall P \in \mathcal{P}^x : \neg b^y_P \wedge \neg b^z_P \implies c^x_P \in T^x_{\mathsf{F}} \tag{2.8}$$

This concludes the description of the computation of dynamic programming table D.

As with any dynamic programming algorithm, even though we only get a single value from any cell, the reconstruction of the solution from the table is quite simple. We start from $D[r, \emptyset, \emptyset, \emptyset, \emptyset]$,

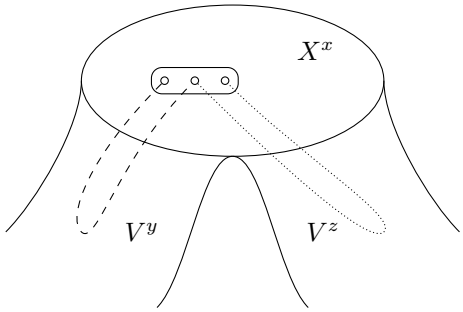

Figure 2.2 Joining groups to form bigger connected components

where r is the node of the tree decomposition $\mathcal{T}$ of G. If it is equal to F, there does not exist any solution. Otherwise, we recursively find the solution. We find a cell that we are extending as described in the computation and also its value is T. Then, we recursively ask for its partial solution - a *partial partition* which is made of open clusters that are made of connected groups $\mathcal{Q}^y$ and closed clusters $\mathcal{R}^y$ with assigned centers $\mathcal{C}^y$, together partitioning the seen vertices V^x. We remind that both open and closed clusters must have its size at most u, the distance to the cluster center has to be at most d for every cluster vertex. Closed clusters have to also satisfy the lower-bound on their size. From the arguments identifying the cell we infer the taken action and apply it to $(\mathcal{Q}^y, \mathcal{R}^y)$, giving us solution $(\mathcal{Q}^x, \mathcal{R}^x)$ for node x. To simplify the notation, we define $\flat(X)$ to be X if X is not a set, otherwise, it is the union of all its elements after calling $\flat$ on them.

Now, we show the correctness of the computation.

▶ **Claim 2.** *If $D[x, \mathcal{P}^x, c^x, s^x, b^x] = \mathsf{T}$ then there exists a partial partition $(\mathcal{Q}^x, \mathcal{R}^x)$ of V^x with assigned centers $\mathcal{C}^x$ and for each open cluster $P \in \mathcal{P}^x$ there exists an open cluster $Q \in \mathcal{Q}^x$ such that*

1. *Every group in Q is a superset of a group in P.*
2. *Let c_Q be the assigned center for Q, its center-type is $c_P^x \in T^x_{b_P^x}$.*
3. *Let P be the representation of the open cluster Q in $\mathcal{P}^x$, then $|\mathtt{flat}(Q)| = s_P^x$.*

Note that in Item 2 checking $c_P^x \in T^x_{b_P^x}$ is enough since the existence of vertices of a given type in the correct part of graph T^x_{F} and T^x_{T} is precomputed correctly. Also note that for a given cluster center c_Q its distance to a vertex w in the bag X^x can be derived from the center type as $\mathrm{dist}_d(w, c_Q) = t_w^{x,c_Q}$.

Proof. We prove the claim by induction on the nice tree decomposition. Starting in the leaf node x, there are no seen vertices, $V^x = \emptyset$, so the partition $\mathcal{P}^x$ is empty, thus we set $(\mathcal{Q}^y = \emptyset, \mathcal{R}^y = \emptyset)$ and $\mathcal{C}^x = \emptyset$ and the claim holds trivially.

For the inductive step, we know there exists $\mathcal{P}^y, c^y, s^y, b^y$ such that $D[y, \mathcal{P}^y, c^y, s^y, b^y] = \mathsf{T}$ or even exists $\mathcal{P}^z, c^z, s^z, b^z$ such also $D[z, \mathcal{P}^z, c^z, s^z, b^z] = \mathsf{T}$, because each non-leaf entry extends one of the entries of the children node(s). We assume the induction hypothesis holds and so the partial partition $(\mathcal{Q}^y, \mathcal{R}^y)$ of V^y exists (or even the partial partition $(\mathcal{Q}^z, \mathcal{R}^z)$ of V^z exists). With small modifications, we create the partial partition $(\mathcal{Q}^x, \mathcal{R}^x)$.

- **Introduce node:** It is the first occurrence of the vertex v in the node x, the child node of x is y.

  Let P^x be the cluster that contains v in $\mathcal{P}^x$. If $P^x = \{\{v\}\}$, then we set $\mathcal{Q}^x := \mathcal{Q}^y \cup \{P^x\}$. We also set an arbitrary vertex that satisfies the center types c_P^x and is in the correct part of the graph b_P^x to be the center type c_Q.

Else, let $P^y \in \mathcal{P}^y$ be the cluster to which v was added. For P^y we obtain the corresponding open cluster $Q^y \in \mathcal{Q}^y$. We connect the same groups in Q^y to obtain Q^x as were connected in P^y to obtain P^x and add v to it. Then we set $\mathcal{Q}^x = (\mathcal{Q}^y \setminus Q^y) \cup Q^x$. We let $\mathcal{R}^x := \mathcal{R}^y$ to remain the same in both cases. For the assigned centers we check whether each center in $\mathcal{C}^y$ satisfies the center types c^x_P, if not we set c_Q in $\mathcal{C}^x$ to another vertex satisfying the type c^x_P.

The pair $(\mathcal{Q}^x, \mathcal{R}^x)$ is a valid partitioning as it partitions V^x (v was added to both V^x and $\mathcal{Q}^x$). Size of clusters is correct because it is equal to the s^x_P which is at most u (this is explained in Item 3). We also have to check that for its cluster center c_Q holds that $\forall w \in \bigcup Q : \mathrm{dist}(w, c_Q) \leq d$.

First, we focus on all vertices except for v and then on v separately. The cluster center c_Q is of the type c^x_P, but also of type c^y_P, because c^y_P is the same as c^x_P except for the missing v coordinate. Now we continue with a case analysis of whether the center c_Q is in the past or future.

- $c_Q \in V^x$: Distance of every vertex from V^y to a center in the V^y is at most d from the induction hypothesis. The only change in cluster centers is for clusters with center $c_Q = v$. But every vertex $w' \in V^y$ was previously verified to a distance $\min_{w \in X^x}\{\mathrm{dist}_d(w, w') + c^y_P, w\} = \mathrm{dist}_d(w', c_Q) \leq d$. Which is c^x_P, v as asserted in Equation (2.4), so the distance is at most d.
- $c_Q \notin V^x$: Every vertex in V^x other than v had already verified its distance to c_Q through a vertex $w \in X^y$, and it still holds $c^y_{P,w} = c^x_{P,w}$.

Since v is in the bag X^x, we know the distance to its center c_{Q^x} is the coordinate v in the center-type t^{x,c_Q}. That the distance is at most d is checked in Equation (2.1).

Now, we prove the items in the claim.

1. We have to check only the open cluster containing v because other clusters remain the same and this condition is satisfied for them through the induction hypothesis. If P^x contains only v, we added it to $\mathcal{Q}^x$ and this condition is satisfied. Otherwise, v was added to Q^x and all group merges in Q^y were done accordingly from P^y to P^x. Since all the merged groups in P^y had a superset from Q^y, the merge of the supersets is a superset of the merged group. And the vertex v is both in the group in P^x and P^y.
   Formally written as $\overline{P}^x = \overline{P}^y_1 \cup \overline{P}^y_2 \cup \cdots \cup \overline{P}^y_k \cup \{v\} \subseteq \overline{Q}^y_1 \cup \overline{Q}^y_2 \cup \cdots \cup \overline{Q}^y_k \cup \{v\} = \overline{Q}^x$.
2. We have checked that a vertex of center-type c^x_P exists in Equation (2.2).
3. For all clusters except P^x, the size remains the same both $|\texttt{flat}(Q^x)| = |\texttt{flat}(Q^y)|$ as well in the table $s^x_{P^x} = s^y_{P_y}$. Only for $P^x : s^x_{P^x} = s^y_{P^y} + 1$ which is asserted in Equation (2.3). From the induction hypothesis we get that it is equal to $|\texttt{flat}(Q^y)| + 1$, and we obtain $|\texttt{flat}(Q^x)|$ by adding v which is the only size change of Q^x.

- **Forget node:** Let $P^y \in \mathcal{P}^y$ be the open cluster containing v at node y. If v isn't the only vertex in P^y, we let $\mathcal{Q}^x := \mathcal{Q}^y$ and $\mathcal{R}^x := \mathcal{R}^y$ which is trivially a partial partition.

  Otherwise, $P^y = \{\{v\}\}$, let $Q^y \in \mathcal{Q}^y$ be the cluster containing v and $\overline{Q}_v$ is the only group in Q^y. We set $\mathcal{Q}^x := \mathcal{Q}^y \setminus Q^y$ and $\mathcal{R}^x := \mathcal{R}^y \cup \overline{Q}_v$. To show that $(\mathcal{Q}^x, \mathcal{R}^x)$ is a partial partition, it is enough to see that Q^y satisfies constraints of this claim from the induction hypothesis and thus $\overline{Q}_v$ is a closed cluster from the perspective of x.

  In both cases, the assigned centers $\mathcal{C}^x$ remain the same. Because also $V^x = V^y$, the maximal distances to cluster centers are also satisfied.

1. One cluster might have been removed from both $\mathcal{Q}^y$ and $\mathcal{P}^y$ without any change to other clusters, this condition holds from the induction hypothesis.

2. The center c_Q exists as $V^x = V^y$ remains the same and has type c_P^y for which the center exists from the induction hypothesis. From the definition of C in Equation (2.7) we know, that c_P^x is the same as c_P^y except it does not have the v coordinate. Therefore, c_Q also is of type c_P^x.
   From the induction hypothesis, we know $c_P^y \in T^y_{b_P^x}$. It remains in the same part of the graph as $V^x = V^y$ and $b_P^x = b_P^y$ as asserted in the definition of B in Equation (2.6). Therefore, it holds that $c_P^x \in T^x_{b_P^x}$.
3. Each open cluster in $\mathcal{Q}^x$ is unchanged from $\mathcal{Q}^y$ and also in Equation (2.6) we contain the cluster size to remain the same, so it still holds that $|\texttt{flat}(Q)| = s_P^x$.

- **Join node:** We set $\mathcal{R}^x := \mathcal{R}^y \cup \mathcal{R}^z$, and we construct $\mathcal{Q}^x$ by merging pairs of open clusters $Q^y \in \mathcal{Q}^y, Q^z \in \mathcal{Q}^z$ into Q^x in such a way that two groups are joined together if they have nonempty intersection. We pair the clusters Q^y, Q^z if $\texttt{flat}(Q^y) \cap X^x = \texttt{flat}(Q^z) \cap X^x$. Non-matching intersection is not possible, as the clusters are also paired this way in $\mathbb{P}$. The set of centers of type $c_P^x = c_P^y = c_P^z$ and part of the graph according to b_P^y or b_P^z might have split with the introduction of b_P^x therefore we adjust the centers $\mathcal{C}^x$ according to that. The existence of such center is checked in Item 2.

  Now, we show $(\mathcal{Q}^x, \mathcal{R}^x)$ is a partial partition of V^x. Every vertex outside the bag is exactly in one cluster because it was either in $V^y \setminus X^x$ or in $V^z \setminus X^x$ and $V^y \cap V^z = X^x$. If a vertex is inside the bag, then it is inside an open cluster and the two copies from different nodes are merged. Each group of a cluster in $\mathcal{Q}^x$ remains connected because the groups from different branches intersect only in the bag which connects the intersecting groups. The size bound is correct because the size of Q is equal to s_P^x as checked in Item 3.

  The last thing remaining to check is the distance bound. The vertices that are in an open cluster with its center in the past in their branch are already verified. If in both of the branches, the cluster center is in the future, then all vertices in that cluster had their distance to center already verified to the same center-type, since the bag remained the same and $c_P^x = c_P^y = c_P^z$. In the remaining case, the remaining vertices have their cluster center in the future in their branch. In the other branch, the center exists with the same center type. The distance to the center of this type was already verified in an earlier node.

  1. For an arbitrary group $\overline{P}^x$ it holds that $\overline{P}^x = \bigcup_{i=1}^k \overline{P}_i^y \subseteq \bigcup_{i=1}^k \overline{Q}_i^y \cup \overline{Q}_i^z = \overline{Q}^x$ as each $\overline{P}^y$ has superset $\overline{Q}^y$ from the induction hypothesis and $\overline{Q}^y \cup \overline{Q}^z$ is a superset of that.
  2. Let c_Q be the center of Q in $\mathcal{Q}^y$. The center-type of c_Q is the same in all the three nodes x, y, z as c^x are the center types even for nodes y and z in Equation (2.8). There are four cases how the center c_Q could be seen from the nodes y, z. The first case when in both children nodes the center c_Q is in the past cannot happen, because of the construction of set B in Equation (2.8). The second case is when both of the centers are in the future, then the existence of the center of the center-type $t^{x,c_Q} \in T^x_{\mathsf{F}}$ is checked for each pair of vectors b^y, b^z in Equation (2.8).
     The remaining two cases are symmetrical. The center c_Q is in the past in respect to the bag X^x and T^x_{T} is the superset of both T^y_{T} and T^z_{T}. The center-type c_Q is in either of them (from the induction hypothesis and that $b_P^y \vee b_P^z$), therefore it is also in T^x_{T}.
  3. For an open cluster $Q^x \in \mathcal{Q}^x$, which was constructed by merging Q^y and Q^z, the set $\texttt{flat}(Q^x)$ equals to $\texttt{flat}(Q^y) \cup \texttt{flat}(Q^z)$. Therefore, the size $|\texttt{flat}(Q^x)| = |\texttt{flat}(Q^y)| + |\texttt{flat}(Q^z)| - |\texttt{flat}(Q^y) \cap \texttt{flat}(Q^z)|$. From the induction hypothesis, we know $s_P^y = |\texttt{flat}(Q^y)|$ and $s_P^z = |\texttt{flat}(Q^z)|$. And we also know the intersection is only in the bag, and it is P, therefore, $|\texttt{flat}(Q^x)| = s_P^x + s_P^z - |X^x \cap P| = s_P^x$. ■

For the root r of the tree decomposition, all clusters are closed because $X_r = \emptyset$ and $V_r = V(G)$. Therefore, if $D[r, \emptyset, \emptyset, \emptyset, \emptyset] = \mathsf{T}$, then $\mathcal{R}^r$ is the partition of V into connected clusters and $\mathcal{I}$ is a Yes – *instance*.

▶ **Claim 3.** *If $(\mathcal{Q}^x, \mathcal{R}^x)$ is a partial partition of V^x with assigned centers $\mathcal{C}^x$ then there exists $\mathcal{P}^x, c^x, s^x, b^x$ such that $D[x, \mathcal{P}^x, c^x, s^x, b^x] = \mathsf{T}$ and for each open cluster $Q \in \mathcal{Q}^x$ there exists an open cluster $P \in \mathcal{P}^x$ such that*

1. *Every group in Q is a superset of a group in P.*
2. *Let c_Q be the assigned center for Q, its center-type is $c^x_P \in T^x_{b^x_P}$.*
3. *Let P be the representation of the open cluster Q in $\mathcal{P}^x$, then $|\texttt{flat}(Q)| = s^x_P$.*

Proof. We prove this claim by induction as well. We start with the leaf node, where $V^x = \emptyset$ and also $\mathcal{Q}^x = \emptyset$ and $\mathcal{R}^x = \emptyset$. By setting $\mathcal{P}^x, c^x, s^x, b^x$ to $\emptyset$ for which $D[x, \emptyset, \emptyset, \emptyset, \emptyset] = \mathsf{T}$. The claim holds trivially, as $\mathcal{Q}^x$ is empty.

In the other nodes, we construct a partial partition $(\mathcal{Q}^y, \mathcal{R}^y)$ (or even $(\mathcal{Q}^z, \mathcal{R}^z)$) from $(\mathcal{Q}^x, \mathcal{R}^x)$. From the induction hypothesis we obtain $\mathcal{P}^y, c^y, s^y, b^y$ such that $D[y, \mathcal{P}^y, c^y, s^y, b^y] = \mathsf{T}$ (or even $\mathcal{P}^z, c^z, s^z, b^z$ such that $D[z, \mathcal{P}^z, c^z, s^z, b^z] = \mathsf{T}$). From which we finally construct $\mathcal{P}^x, c^x, s^x, b^x$ that satisfy $D[x, \mathcal{P}^x, c^x, s^x, b^x] = \mathsf{T}$.

- **Introduce node:** We start by setting $\mathcal{R}^y := \mathcal{R}^x$. Let $Q^x \in \mathcal{Q}^x$ be the cluster containing v and let $\overline{Q}^x$ be the group of v. First, we handle the case v is the only vertex in Q^x. We set $\mathcal{Q}^y := \mathcal{Q}^x \setminus \{Q^x\}$. And we drop the center of Q^x from $\mathcal{C}^x$ to obtain $\mathcal{C}^y$. The pair $(\mathcal{Q}^y, \mathcal{R}^y)$ is a partial partition of V^y since it partitions V^y and $\mathcal{Q}^y \subseteq \mathcal{Q}^x$ and $\mathcal{R}^y := \mathcal{R}^x$.

  In the other case, let $\overline{Q}^y_1, \overline{Q}^y_2, \ldots, \overline{Q}^y_k$ be the connected components $\overline{Q}^x$ splits into if we remove v. Then we set $Q^y := (Q^x \setminus \overline{Q}^x) \bigcup_{i=1}^k \overline{Q}^y_i$ and $\mathcal{Q}^y := \mathcal{Q}^x \setminus \{Q^x\} \cup \{Q^y\}$. And we set $\mathcal{C}^y := \mathcal{C}^x$.

  We show that $(\mathcal{Q}^y, \mathcal{R}^y)$ is a partial partition of V^y. It partitions V^y as we removed v. The size upper-bound is satisfied because we only removed one vertex. The distance is also satisfied because every vertex is the same distance from the same Every group is connected because they are connected in $\mathcal{Q}^x$, or we removed v, and we split the original group into connected components. And the $\mathcal{R}^x$ remained the same.

  Since $(\mathcal{Q}^y, \mathcal{R}^y)$ is a partial partition of V^y, from the induction hypothesis we know that exists $\mathcal{P}^y, c^y, s^y, b^y$ such that $D[y, \mathcal{P}^y, c^y, s^y, b^y] = \mathsf{T}$ and Items 1 to 3 hold. We now show that also suitable $\mathcal{P}^x, c^x, s^x, b^x$ exists. If $Q^x = \{\{v\}\}$ we set $P^x := Q^x$ and $\mathcal{P}^x := \mathcal{P}^y \cup P^x$. Let c_Q be the center of Q and t^{x,c_Q} its center-type, we set $c^x_{P^x} := t^{x,c_Q}$. Furthermore, we set $s^x_{P^x} := 1$ and $b^x_{P^x} := \mathsf{T}$ if $c_Q \in V^x$, otherwise F. The values for remaining clusters will be set later.

  Otherwise, v is not the only vertex in Q^x, and it joins the groups $\overline{Q}^y_1, \overline{Q}^y_2, \ldots, \overline{Q}^y_k$. For $i \in [k]$ we know the groups $\overline{P}^y_i = \overline{Q}^y_i \cap X^y$ are in the open cluster P^y. We let $P^x := (P^y \setminus \{\overline{P}^y_1, \overline{P}^y_2, \ldots \overline{P}^y_k\}) \cup v \cup \bigcup_{i=1}^k \overline{P}^y_i$ and $\mathcal{P}^x := (\mathcal{P}^y \setminus P^y) \cup P^x$. We then set $s^x_{P^x} := s^x_{P^y} + 1$.

  For both of the cases, we further specify the yet unspecified values of c^x, s^x, b^x. For each of the open cluster $Q \in \mathcal{Q}^x$, we let the distances from the center c_Q to every bag vertex $w \in X^y$ be the same as before – $c^x_{P,w} := c^y_{P,w}$. The remaining v coordinate in the center-type of c_Q is set as $c^x_{P,v} := dist_d(v, c_Q)$. The size of the cluster remains the same $s^x_P := s^y_P$ except for the cluster containing v (which was already set). And if the center is in the past changes only if $c_Q = v$ to T, otherwise it remains the same.

  1. If P^x contains only the vertex v, then Q^x is the same. In the other case where v is not the only vertex in P^x this claim holds for all the groups of P^y that were connected by v because $\overline{P}^y_i = \overline{Q}^y_i \cap X^y$. Then the groups were connected the same way in Q^x and P^x, therefore every vertex in the union of groups in P^x is also in the union of groups in Q^x. The rest of the groups remain the same, and thus this claim holds for them through the induction hypothesis.

2. If P^x contains only the vertex v, we have set the center-type $c^x_{P^x} = t^{x,c_Q}$ according to the definition. For the cluster $P = f^x(Q) \in \mathcal{P}^x$, the center-type c^y_P is correct from the induction hypothesis and the v coordinate was set as $dist_d(v, c_q)$ Therefore, all the coordinates are set to distance to c_Q which is the type t^{x,c_Q}.
3. For the cluster containing v, the size was set to 1 if it was the only vertex which is also the size of Q^x. Or it was set to $s^x_P = s^y_P + 1$ which is equal to $|\texttt{flat}(Q^y) \cup \{v\}| = |\texttt{flat}(Q^x)|$ thanks to the induction hypothesis. For the remaining clusters, the size of the cluster is the same because it is the same as in the previous node and there this claim holds through the induction hypothesis.

We now show $D[x, \mathcal{P}^x, c^x, s^x, b^x] = \mathsf{T}$, we show that according to the text setting up the table D. We start with the two condition that if satisfied $D[x, \mathcal{P}^x, c^x, s^x, b^x] = \mathsf{F}$ The condition Equation (2.1) is not satisfied, since $\text{dist}_d(v, c_Q) \leq d$, otherwise v wouldn't be able to be in Q. Also, the condition Equation (2.2) is not satisfied since Item 2 holds.

If P^x only contains the vertex v, we have set the size accordingly to $s^x_{P^x} = 1$. Similarly, in the other case, $\mathcal{P}^y = (\mathcal{P}^x \setminus P^x) \cup P^y$ and $s^x_{P^x} = s^x_{P^y} + 1$ and other sizes remained unchanged. The set $\mathcal{P}^*$ in Equation (2.3) contains P^y since P^x was created by merging multiple groups in P^y neighboring v. Therefore, $D[y, \mathcal{P}^y, c^y, s^y, b^y]$ is one of the iterated entries on the right side of Equation (2.3) and since it is T, this equation is satisfied.

In Equation (2.4), c^y does not contain the v coordinate and for any open cluster $P \in \mathcal{P}^x$. The center c_Q is in the future if and only if $b^x_P = \mathsf{F}$. Otherwise, the bag X^y is a separator between c_Q and $c^x_{P,v} = \min_{w \in X^y}\{c^y_{P,w} + \text{dist}_d(v, w)\}$ holds, because the shortest path from v to the center has to go through a vertex $w \in X^y$ and $c^y_{P,w}$ is correct size from the induction hypothesis.

Finally, Equation (2.5) contains $D[y, \mathcal{P}^y, c^y, s^y, b^y]$ on the right side of the equation since if c_Q is in V^y it is also in V^y, and we only changed b^x_P to T if $c_Q = v$, and we have checked that c_Q satisfies the distances of c^x_P earlier. Therefore, Equation (2.5) is satisfied because $D[y, \mathcal{P}^y, c^y, s^y, b^y] = \mathsf{T}$.

- **Forget node:** First, we handle the case when v is in a cluster in $\mathcal{R}^x$. Let $R \in \mathcal{R}^x$ be the cluster containing v, we set $Q^y := \{R\}$. Then we set $\mathcal{Q}^y := \mathcal{Q}^x \cup Q^y$ and $\mathcal{R}^y := \mathcal{R}^x \setminus \{R\}$.

  The pair $(\mathcal{Q}^y, \mathcal{R}^y)$ is a partial partition of V^y because the closed cluster has to satisfy everything open cluster and the lower-bound on the size as well. There is an exception that a group has to be connected, not the open cluster, but we made a group from R.

  Otherwise, v is in an open cluster $Q^x \in \mathcal{Q}^x$. We set $\mathcal{Q}^y := \mathcal{Q}^x$ and $\mathcal{R}^y := \mathcal{R}^x$ and because it remained the same it trivially is a partial partition.

  Since $(\mathcal{Q}^y, \mathcal{R}^y)$ is a partial partition of V^y, from the induction hypothesis we know that exists $\mathcal{P}^y, c^y, s^y, b^y$ such that $D[y, \mathcal{P}^y, c^y, s^y, b^y] = \mathsf{T}$ and Items 1 to 3 hold.

  If $v \in \texttt{flat}(\mathcal{R}^x)$, we set $P^y := \{\{v\}\}$ and $\mathcal{P}^x := \mathcal{P}^y \setminus \{P^y\}$. In the other case, let $P^y \in \mathcal{P}^y$ is the open cluster of v and let $\overline{P}^y \in P^y$ be the group containing v. We set the group $\overline{P}^y := \overline{P}^x \setminus \{v\}$, the cluster $P^x := (P^y \setminus \{\overline{P}^x\}) \cup \{\overline{P}^y\}$ and the set of open clusters $\mathcal{P}^x := (\mathcal{P}^y \setminus \{P^y\}) \cup P^x$. Additionally, we set $s^x_{P^x} := s^y_{P^y}$.

  In both cases, we set the remaining entries of c^x, s^x, b^x. For all open clusters $P \in \mathcal{P}^x \setminus P^x$, we set c^x to remain the same on all coordinates as c^y except for the removed v coordinate. The sizes we set as $s^x_P := s^y_P$ and $b^x_P := b^y_P$ to remain the same.

  1. Every group in Q^x is also in Q^y for which this condition holds. And the corresponding group P^x was also the same in P^y. Therefore, this condition trivially holds.
  2. Let c_Q be the center of an open cluster $Q \in \mathcal{Q}^x$, then its center-type t^{x,c_Q} is given by the distances to vertices in the bag X^x. From X^x the vertex v was dropped to get X^y, but the

distances remained the same. Additionally, $V^x = V^y$ and $b^x_P = b^y_P$ for each P. Therefore, if $t^{x,c_Q} \in T^x_{b^x_P}$, so is $t^{y,c_Q} \in T^y_{b^y_P}$.

3. The sizes remained the same for every open cluster as well as $s^x = s^y$. And $s^y_P = |Q^x|$ holds from the induction hypothesis.

We now show $D[x, \mathcal{P}^x, c^x, s^x, b^x] = \mathsf{T}$, we show that according to the text setting up the table D. Starting with Equation (2.6) where the case $P^y = \{\{v\}\}$ is forgotten. The sets $C(\mathcal{P}^x, c^x), S(\mathcal{P}^x, s^x), B(\mathcal{P}^x, b^x)$ contain c^y, s^y, b^y respectively. Because the sets $C(\mathcal{P}^x, c^x), B(\mathcal{P}^x, b^x)$ does not impose any limits to c^y and b^y. The set $S(\mathcal{P}^x, s^x)$ contains s^y, because $\ell \leq s^y_{P^y} \leq u$ holds since $s^y_{P^y} = |R|$. Therefore, $D[x, \mathcal{P}^x, c^x, s^x, b^x]$ is on the right side of Equation (2.6)which is hence satisfied.

We continue with the case where P^y contains more vertices than just v. The only difference between $\mathcal{P}^x$ and $\mathcal{P}^y$ indeed is that P^x does not contain v and P^v contains v. The set $C'(\mathcal{P}^y, c^x)$ contains c^y because we created c^y in such a way we only forgot the v coordinate of the center type, and we kept the others the same. Therefore, $D[x, \mathcal{P}^x, c^x, s^x, b^x]$ is on the right side of Equation (2.7).

If the center c_Q is in the past, the extra condition holds, since $b^y_P = \mathsf{T}$. Otherwise, c_Q is in the future and the bag x^x is a separator between c_Q and v. Therefore, the distance $c^y_{P,v}$ represents a shortest path from v to c_Q which goes through the bag.

- **Join node:**

We set $\mathcal{R}^y := \{R \in \mathcal{R}^x \mid R \subset V^y\}$. Then for each $Q^x \in \mathcal{Q}^x$, we transform it to $Q^y := \{\overline{Q}^y \subseteq V^y \mid \exists \overline{Q}^x \in Q^x : \overline{Q}^y \subseteq \overline{Q}^x \wedge \overline{Q}^y$ is maximal set such that $G[\overline{Q}^y]$ is connected$\}$ and add it to $\mathcal{Q}^y$. Similarly, we create $\mathcal{R}^z$ and $\mathcal{Q}^z$.

We show that $(\mathcal{Q}^y, \mathcal{R}^y)$ is a partial partition of V^y. All vertices in V^y must be in $\mathcal{Q}^x$ or $\mathcal{R}^x$. Every open cluster was transformed into a smaller open cluster with only the vertices from V^y. The closed clusters are entirely in V^y or V^z as they do not go through the bag and are connected. Therefore, $(\mathcal{Q}^y, \mathcal{R}^y)$ is a partition. The closed clusters were not changed, so their size is correct, and they remain connected. The open clusters were split into connected groups and their size was made smaller, but there is no lower-bound only upper-bound that was satisfied before. The distance requirement is satisfied for both open and closed clusters, as they are only a subset of the vertices that they were in V^x.

Similarly, $(\mathcal{Q}^z, \mathcal{R}^z)$ is a partial partition of V^z. Since $(\mathcal{Q}^y, \mathcal{R}^y)$ and $(\mathcal{Q}^z, \mathcal{R}^z)$ are partial partition of V^y and V^z respectively, from the induction hypothesis we know that exists $\mathcal{P}^y, c^y, s^y, b^y$ and $\mathcal{P}^z, c^z, s^z, b^z$ such that $D[y, \mathcal{P}^y, c^y, s^y, b^y] = D[y, \mathcal{P}^y, c^y, s^y, b^y] = \mathsf{T}$ and Items 1 to 3 holds in respect to them.

We create the partition of the bag $\mathcal{P}^x$ by joining groups that share vertices in $\mathcal{P}^y$ and $\mathcal{P}^z$, and set the center-types $c^x := c^y$. Additionally, for each open cluster $P \in \mathcal{P}^x$, we set $s^x_P := s^y_P + s^z_P - |P|$ and $b^x_P := b^y_P \vee b^z_P$.

1. For an arbitrary group $\overline{P}^x$ it holds that $\overline{P}^x = \bigcup_{i=1}^k \overline{P}^y_i \subseteq \bigcup_{i=1}^k \overline{Q}^y_i \cup \overline{Q}^z_i = \overline{Q}^x$ as each $\overline{P}^y$ has superset $\overline{Q}^y$ from the induction hypothesis and $\overline{Q}^y \cup \overline{Q}^z$ is a superset of that.
2. The bags X^x and X^y are the same, therefore the distances from the center c_Q to it, remain the same. Hence, the center-types $t^{x,c_Q} = t^{y,c_Q}$. And it holds that $c^x_P = c^y_P = t^{y,c_Q} = t^{x,c_Q}$. The b^x_P was set as logical or of b^y_P and b^z_P. The cluster center c_Q satisfied both of them because of the induction hypothesis. If it is in the past in one branch, then it is in the past with respect to x as $V^x = V^y \cup V^z$. Therefore, b^x_P is set correctly and $t^{x,c_Q} \in T^x_b$ holds.
3. For cluster Q^x its size is $|\texttt{flat}(Q^x)| = |\texttt{flat}(Q^y)| + |\texttt{flat}(Q^z)| - |\texttt{flat}(Q^y) \cap \texttt{flat}(Q^z)|$ from the induction hypothesis and the fact that $|\texttt{flat}(Q^y) \cap \texttt{flat}(Q^z)| = |P|$ we get then it equals to $s^y_P + s^z_P - |P|$ which is exactly how we set s^x_P to be.

We now show $D[x, \mathcal{P}^x, c^x, s^x, b^x] = \mathsf{T}$, we show that according to the text setting up the table D. There is only Equation (2.8) we have to verify. The pair of open cluster partitions $(\mathcal{P}^y, \mathcal{P}^z)$ is in $\mathbb{P}$ because if two groups $\overline{P}^y, \overline{P}^z$ share a vertex then $(\overline{P}^y \cup \overline{P}^z) \subseteq \overline{P}^x$. The center-types c^x, c^y, c^z are the same. The pair of size vectors (s^y, s^z) are in $S(s^x, \mathcal{P})$ because for each open cluster $P \in \mathcal{P}$, we have set $s^x_P := s^y_P + s^z_P - |P|$. And the pair of boolean vectors (b^y, b^z) are in $B(b^x, \mathcal{P})$ since for each open cluster $P \in \mathcal{P}$, we have set $b^x_P := b^y_P \vee b^z_P$. Therefore, both entries $D[y, \mathcal{P}^y, c^y, s^y, b^y]$ and $D[z, \mathcal{P}^z, c^z, s^z, b^z]$ can be found on the right side of Equation (2.8) and remains to prove the extra condition is true.

For an open cluster $P \in \mathcal{P}^x$ if either $b^y_P = \mathsf{T}$ or $b^y_P = \mathsf{T}$, the condition holds trivially. Let c_Q be the cluster of $Q \in \mathcal{Q}^x$ such that $f^x(Q) = P$. If both $b^y_P = b^y_P = \mathsf{F}$ the center c_Q is not in V^y neither in V^z. Since $V^x = V^y \cup V^z$ it is neither in V^x, therefore $t^{x,c_Q} \in T^x_{\mathsf{F}}$. And we know that $c^x_P = t^{x,c_Q}$ from Item 2. ■

Let r be the root node in the tree decomposition of G. If $(\mathcal{Q}^r, \mathcal{R}^r)$ is a partial solution of $V^r = V(G)$, then from this claim $D[r, \mathcal{P}^r, c^r, s^r, b^r] = \mathsf{T}$, where $D[r, \mathcal{P}^r, c^r, s^r, b^r] = D[r, \emptyset, \emptyset, \emptyset, \emptyset]$ since $\mathcal{Q}^r = \emptyset$ because $X^r = \emptyset$.

It remains to show the time complexity of the algorithm. There are at most tw groups in all the open clusters for a given node, since each has to contain at least one vertex. Each group is in one of the at most tw open clusters, therefore there are at most $\mathrm{tw}^{\mathrm{tw}}$ ways to arrange just the groups to clusters. Additionally, there are at most tw vertices that are being distributed and each is assigned to one group. That makes at most $\mathrm{tw}^{\mathrm{tw}}$ ways to assign vertices to a fixed structure of groups within open clusters. Therefore, there is $\mathrm{tw}^{2\cdot\mathrm{tw}} \in \mathcal{O}(\mathrm{tw}^{\mathrm{tw}})$ possible ways to arrange $\mathcal{P}$.

Then the set of all possible center-types for a given bag T^x contains at most $(d+2)^{\mathrm{tw}}$ elements. The function c^x is determined by the center-type of each open cluster, therefore there are $|T^x|^{\mathrm{tw}} = ((d+2)^{\mathrm{tw}})^{\mathrm{tw}} \in d^{\mathcal{O}(\mathrm{tw}^2)}$ possible ways of choosing c^x. The size of an open cluster is between in $[u]$, therefore there are $u^{\mathcal{O}(\mathrm{tw})}$ ways to arrange s^x Finally, the center of a cluster can only be in the past or future, hence, there are $2^{\mathcal{O}(\mathrm{tw})}$ ways of b^x.

In total, for each node of the tree decomposition, there are at most $\mathrm{tw}^{\mathcal{O}(\mathrm{tw})}\, d^{\mathcal{O}(\mathrm{tw}^2)} u^{\mathcal{O}(\mathrm{tw})} 2^{\mathcal{O}(\mathrm{tw})} \in d^{\mathcal{O}(\mathrm{tw}^2)} u^{\mathcal{O}(\mathrm{tw})}$ entries in D. But to compute an entry, one has to search through entries on one or two previous nodes, which can be upper-bounded by all the entries. Therefore, all the entries tied to a specific tree decomposition node can be computed in time $(d^{\mathcal{O}(\mathrm{tw}^2)} u^{\mathcal{O}(\mathrm{tw})})^3$, which is still $d^{\mathcal{O}(\mathrm{tw}^2)} u^{\mathcal{O}(\mathrm{tw})}$. There are $\mathcal{O}(n\,\mathrm{tw})$ nodes in the tree decomposition of G. And at the beginning of this section, it was argued the calculation of distances can be computed in $\mathcal{O}(n^2\,\mathrm{tw}(\mathrm{tw} + \log n))$. Hence, the algorithm runs in time $\mathcal{O}(n^2\,\mathrm{tw}^2(\mathrm{tw} + \log n)) + d^{\mathcal{O}(\mathrm{tw}^2)} u^{\mathcal{O}(\mathrm{tw})}\,\mathrm{tw}\, n$ which can be simplified to $\mathcal{O}(n^3) + d^{\mathcal{O}(\mathrm{tw}^2)} u^{\mathcal{O}(\mathrm{tw})} n$. ◀

▶ **Corollary 5.** *Connected Network Microaggregation is in* XP *parameterized by* tw *only, and is fixed-parameter tractable when parameterized by* $\mathrm{tw} + u + d$.

2.2 Algorithms Based on Branching and ILP

Our next set of algorithms combines exhaustive branching with an encoding into an Integer Linear Program (ILP) where the number of variables is bounded by the parameters. Such ILPs are known to be fixed-parameter tractable:

▶ **Proposition 6** (Lenstra Jr. 1983; Kannan 1987; Frank; Tardos 1987)**.** *There is an algorithm that solves an input* ILP *instance $\mathcal{I}$ with p variables in time $p^{\mathcal{O}(p)} \cdot |\mathcal{I}|$.*

Both of our ILP algorithms are parameterized by the vertex cover number, and we expect the instance to come equipped with the vertex cover. This does not falsify our results as the vertex cover of size k can be calculated in FPT time [28]. We start by defining the types of vertices outside the vertex cover and the types of clusters, both of which are based on the distances to the vertices in M.

▶ **Definition 15.** Let $M = \{v_1, v_2, \dots v_{\text{vc}}\}$ be a vertex cover of the graph $G = (V, E)$ and let $v \in V \setminus M$. The *vertex-type* $t^v \in \{[d] \cup \{\infty\}\}^M$ of v is

$$t_i^v = \begin{cases} \omega(v, v_i) & \text{If } \{v, v_i\} \in E, \\ \infty & \text{Otherwise.} \end{cases}$$

By T we denote the set of all vertex-types of vertices.

▶ **Definition 16.** Let $C \subseteq V$ be a cluster. The *cluster-type* $t(C) \subseteq T$ of C is defined as $t(C) = \{t \in T \mid \exists v \in C : t^v = t\}$. Moreover, let $c \in T \cup M$. We call $(t(C), c)$ the *extended-cluster-type* of cluster C with center c.

In this section, we will often be referring to vertices through their vertex-types, and hence we use $\text{dist}(v, t)$ as shorthand for the distance from v to any vertex of vertex-type t excluding v. Furthermore, we use $\text{dist}(t) := \text{dist}(v, w)$, where $v \neq w$ and $t^v = t^w = t$ as the distance between two different vertices of vertex-type t.

We observe that there are $|T| \leq (d+1)^{\text{vc}}$ different vertex-types, plus additional vc vertices in the vertex cover that are dealt with separately. Moreover, the number of different cluster-types is at most $2^{\text{vc}+|T|} = 2^{\text{vc}+(d+1)^{\text{vc}}} \in 2^{d^{\mathcal{O}(\text{vc})}}$. When also considering the centers, we get at most $2^{d^{\mathcal{O}(\text{vc})}} \cdot (\text{vc} + (d+1)^{\text{vc}}) \in 2^{d^{\mathcal{O}(\text{vc})}}$ possible extended-cluster-types.

▶ **Theorem 7.** *Connected Network Microaggregation is fixed-parameter tractable parameterized by the vertex-cover number* vc *and the maximum distance d.*

Proof. Assume $u > 1$, otherwise the instance is trivial. Then, every cluster has at least one vertex in the vertex cover because the cluster is connected. This upper bounds the number of clusters by vc.

We start by guessing the number of clusters $m \in [\text{vc}]$, all theirs extended-cluster-types $((t(C_1), c_1), (t(C_2), c_2), \dots, (t(C_m), c_m))$ and their vertex cover vertices $(M_1, M_2, \dots, M_m)$. The total number of guesses can be bounded by $(2^{d^{\mathcal{O}(\text{vc})}} \cdot 2^{\text{vc}})^{\text{vc}+1} \in 2^{d^{\mathcal{O}(\text{vc})}}$. Then we check whether all guessed extended-cluster-types are valid, if not, we continue with the next guess. For an extended-cluster-type $(t(C_i), c_i)$ to be valid the following has to hold.

- The graph on vertices M_i and all vertices of vertex-types in $t(C_i)$ is connected.
- $\forall x \in t(C_i) \cup M_i$ we have $\text{dist}(c_i, x) \leq d$

Finally, the guessed collection is valid if M_i partition M, that is, if $\bigcup_{i=1}^{m} M_i = M$ and $\sum_{i=1}^{m} |M_i| = |M|$.

Then we use an ILP to find whether there exists a solution respecting the guessed extended-cluster-types. In order to do so, we use non-negative variables x_i^t representing the number of vertices of vertex-type t in the cluster C_i.

For each $i \in [m]$ we add the following conditions:

$$t \in t(C_i) : x_i^t \geq 1 \tag{2.9}$$

$$t \notin t(C_i) : x_i^t = 0 \tag{2.10}$$

$$x^i_{t^{c_i}} \leq 1 \text{ if } \operatorname{dist}(t^{c_i}) > d \tag{2.11}$$

$$\ell \leq \sum_{t \in t(C_i)} x_i^t + |M_i| \leq u \tag{2.12}$$

And then we add these conditions:

$$\forall t \in T : \sum_{i=1}^{m} x_i^t = |V^t|, \tag{2.13}$$

where V^t is the set of all vertices of vertex-type t.

If the vertex-type t is not in the cluster-type of i-th cluster, we still create the variable x_i^t but set it to zero. Otherwise, there has to be at least one vertex as the certificate of the cluster-type. A vertex of the same vertex-type as the cluster center might be farther than d. To prevent adding more vertices than the cluster center, we add Equation (2.11). Then in Equation (2.12) we check that all clusters have the correct sizes. Lastly, Equation (2.13) ensures that every vertex is used.

▶ **Claim 4.** *If the ILP with Equations* (2.9) *to* (2.13) *has a solution, then $\mathcal{I}$ is a* Yes *-instance.*

Proof. Let $\boldsymbol{x}$ be a solution of the ILP. First, we partition the vertices V^t into sets V_i^t such that $|V_i^t| = x_i^t$ and $V_i^t \cap V_j^t \neq \emptyset$ implies $j = i$. This is possible because of Equation (2.13). Now we set clusters $C_i = \cup_{t \in T} V_i^t \cup M_i$. Each vertex is in a cluster since vertices in M were checked beforehand and Equation (2.13) forces that every vertex in $V \setminus M$ is used exactly once.

Every cluster has a correct size:

$$\ell \leq \sum_{t \in t(C_i)} x_i^t + |M_i| = \sum_{t \in t(C_i)} |V_i^t| + |M_i| = |C_i| \leq u$$

The distance to the center c_i is the same for each vertex of a given vertex-type except for vertices of vertex-type t^{c_i}. For contradiction, suppose that there is another vertex $v \in C_i$ of vertex-type t^{c_i} and $\operatorname{dist}(c_i, v) > d$. But then for t^{c_i} Equation (2.11) is present, and it is not satisfied, since $x^i_{t^{c_i}} \geq 2$. Therefore, it is sufficient to check the distances only by examining the vertex-types in $t(C_i)$ and the vertex cover vertices M.

Each path to the center has to go through the vertex cover, hence it is also sufficient to validate connectivity beforehand by examining only $t(C_i)$ and M_i. Finally, we can say that the cluster partitioning $\Pi = (C_1, C_2, \ldots, C_m)$ with centers $\mathcal{C} = (c_1, c_2, \ldots, c_m)$ is a valid solution. ■

▶ **Claim 5.** *If $\mathcal{I}$ is a* Yes *-instance, then the ILP Equations* (2.9) *to* (2.13) *has a solution.*

Proof. Let $\Pi = (C_1, C_2, \ldots, C_m)$ with centers $\mathcal{C} = (c_1, c_2, \ldots, c_m)$ be a solution. For cluster C_i and vertex-type $t \in t(C_i)$ set the variable $x_i^t = |V_i^t|$, where $V_i^t = \{v \in C_i \mid t^v = t\}$.

Now we show that every condition in the ILP formulation is satisfied one by one in the corresponding order. Equations (2.9) to (2.13) are shown for fixed $i \in [m]$.

1. Recall that in Definition 15 $t \in t(C) \iff \exists v \in C : t^v = t$ therefore $\forall t \in t(C_i) : 1 \leq |V_i^t| = x_i^t$.
2. Similarly, if $t \notin t(C_i)$, then there is no vertex v with $t^v = t$ in C_i, that is, $0 = |V_i^t| = x_i^t$.

3. For contradiction, suppose Equation (2.11) is not satisfied. Then, there is a vertex $v \in C_i$ of vertex-type t^{c_i} such that $v \neq c_i$. This means that $\text{dist}(c_i, v) > d$, which is a contradiction.
4. We know that $\ell \leq |C_i| \leq u$ holds, and we will rewrite it as $|C_i| = \sum_{t \in t(C_i)} |V_i^t| + |M_i| = \sum_{t \in t(C_i)} x_i^t + |M_i|$.
5. Finally, for any vertex-type $t \in T$ the number of vertices $|V^t| = \sum_{i=1}^{m} |V_i^t| = \sum_{i=1}^{m} x_i^t$. ■

The number of vertex-types is bounded as $|T| \leq (d+1)^{\text{vc}}$ and the number of cluster-types is bounded by $2^{\text{vc}+(d+1)^{\text{vc}}+1}$. Therefore, the number of variables is at most $(d+1)^{\text{vc}} \cdot 2^{d^{\mathcal{O}(\text{vc})}} \in 2^{d^{\mathcal{O}(\text{vc})}}$ and the ILP can be solved in FPT time as noted in Proposition 6. Together with that the total number of guesses (the number of ILP formulations) is bounded by a function of vc and d, the theorem follows. ◀

Next, we turn to Network Microaggregation. The following algorithm is based on similar ideas as the previous Theorem, but there is an additional complication: the number of clusters is no longer upper bounded by vc. Hence, we use variables in the ILP formulation to capture how often each of the extended-cluster-types occurs.

▶ **Theorem 8.** *Network Microaggregation is fixed-parameter tractable parameterized by* $\text{vc} + d$.

Proof. Let U be the set of all valid cluster-types. Furthermore, let $P = U \times (T \cup M)$ be the set of all extended-cluster-types.

The main observation in this setting is that we can decide how many vertices of various vertex-types to assign to cluster-types without deciding distribution within clusters. Then, the viability of such an assignment can be checked as it is sufficient if a vertex assignment that makes all the clusters the same size (up to a difference of 1) is between ℓ and u. More precisely, if we know the number of clusters z of extended-cluster-type (τ, c) then we need that

$$\ell \leq \frac{\text{total number of vertices assigned to } (\tau, c)}{z} \leq u.$$

To deal with the vertex cover vertices we use similar technique as in the connected case. We start by guessing the number of clusters $m \leq \text{vc}$ with vertex cover vertices, their extended-cluster-types $\mathcal{T} = ((t(C_1), c_1), (t(C_2), c_2), \ldots, (t(C_{m'}), c_{m'}))$ and their vertex cover vertices $\mathcal{M} = (M_1, M_2, \ldots, M_{m'})$.

We find the number of assigned vertices of vertex-type t to each extended-cluster-type p as well as the number of clusters z of each extended-cluster-type using the following ILP. First, we define the variables.

1. x_i^t representing the number of vertices of vertex-type t in the cluster C_i.
2. y_p^t for every $t \in T, p \in P$ indicating how many vertices of vertex-type t are used in all clusters with extended-cluster-type p.
3. $z_p \geq 0$ for every $p \in P$ indicating the number of clusters with extended-cluster-type p.

And then the conditions. We use Equations (2.9) to (2.12) from ILP in Theorem 7. For each $i \in [m']$ we add the following conditions:

$$t \in t(C_i) : x_i^t \geq 1 \tag{2.14}$$

$$t \notin t(C_i) : x_i^t = 0 \tag{2.15}$$

$$x_{t^{c_i}}^i \leq 1 \text{ if } \text{dist}(t^{c_i}) > d \tag{2.16}$$

$$\ell \leq \sum_{t \in t(C_i)} x_i^t + |M_i| \leq u \tag{2.17}$$

Then, for every extended-cluster-type $p = (\tau, c) \in P$ we further add these conditions.

$$\forall t \in \tau : y_p^t \geq z_p \tag{2.18}$$

$$\forall t \notin \tau : y_p^t = 0 \tag{2.19}$$

$$y_{t^c}^p \leq z_p \text{ if } \operatorname{dist}(t^c) > d \tag{2.20}$$

$$\ell \cdot z_p \leq \sum_{t \in T} y_p^t \leq u \cdot z_p \tag{2.21}$$

Equations (2.18) to (2.21) are equivalents of Equations (2.14) to (2.17) for variables of multiple clusters. In contrast, either we have to have at least one vertex representing the vertex-type for each cluster of extended-cluster-type, or none at all. Same with Equation (2.20), we cap the total cluster center vertex-type vertices to one vertex per cluster if they are too far apart. Then in Equation (2.21) we bound the total number of vertices assigned to one extended-cluster-type.

Finally, we add the following conditions.

$$\forall t \in T : \sum_{i=1}^{m'} x_i^t + \sum_{p \in P} y_p^t = |V^t| \tag{2.22}$$

Which makes sure that in total we counted exactly all vertices across the variables in $\boldsymbol{x}, \boldsymbol{y}$ for each vertex-type.

▶ **Claim 6.** *If ILP above has a solution, then $\mathcal{I}$ is a* Yes $-$ *instance.*

Proof. Let $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ be a solution. First, we partition the vertices V^t into sets $\forall i \in m' : V_i^t, \forall p \in P : V_p^t$. This is possible thanks to Equation (2.22).

We start with creating clusters containing vertex cover vertices $C_i = \cup_{t \in t(C_i)} V_i^t \cup M_i$ for $i \in [m']$. Those are valid clusters as proved in Claim 4, because the variables $\boldsymbol{x}$ and Equations (2.18) to (2.21) are identical to Equations (2.14) to (2.17).

Next, for each extended-cluster-type $p = (\tau, c)$ we create z_p many clusters $C_p^1, C_p^2, \ldots C_p^{z_p}$. To each of them we add one vertex of each vertex-type that is in the cluster-type τ. Thanks to Equation (2.18) we have at least z_p vertices of every used extended-cluster-type. Then we divide the remaining vertices V_p^t such that every cluster has size within the lower and upper bound. This can be done, since Equation (2.21) holds by simply distributing the number of vertices as evenly as possible.

The distance to the center c is the same for each vertex of a given vertex-type except for vertices of vertex-type t^c. But if another vertex v of the vertex-type t^c is in C, then $\operatorname{dist}(c, v) \leq d$, because we know that $y_p^{t^c} > z_p$ holds. So, the extra condition upper bounding the number of vertices of this vertex-type was not added. Therefore, all vertices in cluster are close enough to the cluster center.

Each vertex is in a cluster, since vertices in M were checked beforehand, and Equation (2.22) forces that every non vertex cover vertex is used exactly once.

Finally, we can say that the created cluster partitioning is valid and the instance is Yes -instance. ■

▶ **Claim 7.** *If $\mathcal{I}$ is a* Yes *-instance, then ILP above has a solution.*

Proof. Let $\Pi = (C_1, C_2, \ldots, C_{m'}, \ldots, C_m)$ with centers $\mathcal{C} = (c_1, c_2, \ldots, c_{m'}, \ldots, c_m)$ be a solution and clusters $C_1, C_2, \ldots, C_{m'}$ are exactly those clusters that contain vertex cover vertices. For cluster $C_i, i \leq m'$ and vertex-type $t \in t(C_i)$ set variable $x_i^t = |V_i^t|$, where $V_i^t = \{v \in C_i | t^v = t\}$. Then for each of the extended-cluster-type $p \in P$, we set $y_p^t = |\{v \in C_i \mid t^v = t, p = (t(C_i), c_i), i > m'\}|$ and $z_p = |\{C_i \mid p = (t(C_i), c_i), i > m'\}|$.

Equations (2.14) to (2.17) are exactly the same as in Theorem 7 and the vector $\boldsymbol{x}$ has the same meaning therefore correctness of these conditions hold from Claim 5. Now we show the

remaining conditions of the ILP formulation above is satisfied. We show that Equations (2.18) to (2.21) hold for arbitrary $p = (\tau, c) \in P$.

11. Since $t \in \tau$, then each cluster contain at least one vertex of vertex-type t, with z_p clusters of the extended-cluster-type p, it holds that $y_p^t \geq z_p$.

12. No vertices of vertex-type t are in the clusters of cluster-type τ because $t \notin \tau$, so $y_p^t = 0$.

13. For contradiction, suppose that this condition is not satisfied. Either $y_p^{t^c} < z_p$, but that means there are some clusters of cluster-type τ that does not contain any vertex of vertex-type t^c therefore this is not possible. Or $y_p^{t^c} > z_p$, but then one of the clusters contain at least two vertices of vertex-type t^c. The distance between each pair is more than d so if one of them is the cluster center then the clustering is not valid and we have a contradiction.

14. Because $\forall i \in [m] : \ell \leq |C_i| \leq u$ we bound the total amount of vertices with the same extended-cluster-type $p = (\tau, c)$ by

$$z_p \cdot \ell \leq \sum_{\substack{i:i>m \wedge \\ (t(C_i),c_i)=p}} |C_i| \leq z_p \cdot u.$$

Then we rewrite it to wanted form

$$\sum_{\substack{i:i>m' \wedge \\ (t(C_i),c_i)=p}} |C_i| = \sum_{t \in T} \sum_{\substack{i:i>m' \wedge \\ (t(C_i),c_i)=p}} |V_i^t| = \sum_{t \in T} y_p^t$$

15.

$$\begin{aligned} |V^t| &= \sum_{i=1}^{m} |V_i^t| = \sum_{i=1}^{m'} x_i^t + \sum_{p \in P} \sum_{\substack{i:i>m' \wedge \\ (t(C_i),c_i)=p}} |V_i^t| \\ &= \sum_{i=1}^{m'} x_i^t + \sum_{p \in \mathcal{P}} y_{\tau,c}^t \end{aligned}$$

■

We have only guessed the clusters for vertex cover vertices therefore the number of guesses can be limited by $2^{d^{\mathcal{O}(\mathrm{vc})}}$ as in Theorem 7. For each guess we run ILP with three types of variables, the amount of each type of variables is at most $|T| \cdot |P| \cdot (|T| + \mathrm{vc})$. In total, the number of variables is bounded by the parameters, so the ILP is solvable in FPT time as noted in Proposition 6. ◀

Chapter 3
Lowerbounds

A fast algorithm for our algorithm does not have to exist. The proposed algorithm can be the fastest one that can be achieved when looking from the complexity classes perspective. This is usually too ambitious to prove. Instead, we use a widely believed hypothesis FPT $\neq$ W[1] to show that there is no FPT algorithm with respect to a certain parameterization unless the hypothesis is false. Ideally, we also want to justify the usage of every parameter for every algorithm that was shown in Chapter 2. The justification is made in the sense that if that parameter is dropped, an algorithm belonging to the same class as before also falsifies the hypothesis.

In Theorem 4 we have shown an FPT algorithm for CNMA when parameterized by tw $+d+u$. The problem is NP-hard even when d and u are constants, as we have already shown in Claim 1, therefore FPT algorithm is the best what we can hope for. Also, the structural parameters are always justified. In this chapter, we show that the CNMA is W[1]-hard parameterized by tw $+d$. This justifies the parameters tw and d, but not u in the FPT algorithm. It also shows that the XP algorithm is the best we can hope for when parameterized by tw $+d$ or just by tw. Later in Chapter 4 we provide an even stronger result that the problem W[1]-hard parameterized by td $+d$. For the FPT algorithm for NMA when parameterize by vc $+d$ we show the problem is W[1]-hard when parameterized only by vc.

▶ **Theorem 9.** *Connected Network Microaggregation is* W[1]*-hard parameterized by* tw*, even if* $d = 1$.

Proof. We reduce from the Equitable Connected Partition (ECP for short) problem. Here, we are given an undirected graph H and an integer $r \in \mathbb{N}$, and our goal is to decide whether there is a partition of $V(H)$ into r components $B_1, \ldots, B_r$ such that for any $i \in [r]$ it holds that $H[B_i]$ is connected and $|B_i| \in \{\lfloor \frac{n_H}{r} \rfloor, \lceil \frac{n_H}{r} \rceil\}$, where $n_H = |V(H)|$. It is known that ECP is W[1]-hard with respect to the treewidth plus the number of components r [29]. Moreover, the hardness reduction for ECP holds in the case $n_H \mod r = 0$, which we assume for the rest of the proof.

Let (H, r) be an instance of ECP with $n_H \mod r = 0$, we construct an equivalent instance of Connected Network Microaggregation. We set the bounds $\ell = u = \frac{n_H}{r}$ and $d = 1$, and obtain a new graph G from H as follows. First, we add an extra vertex v to the graph and connect it to every other vertex. The intuition here is that v makes any connected cluster of valid size admissible, bypassing the distance requirement. Second, we make sure that v cannot join any cluster in the original vertex set $V(G)$ by introducing $\ell - 1$ additional vertices $v_1, \ldots, v_{\ell-1}$ and connecting them only to v. Finally, the unweighted graph is converted to a weighted graph by setting each edge weight to one.

To show the correctness of the reduction, let $B_1, \ldots, B_r$ be the solution for the instance (H, r) of ECP. We claim that clusters $\Pi = (B_1, \ldots, B_r, \{v, v_1, v_2, \ldots v_{\ell-1}\})$ with centers $\mathcal{C} = (v, \ldots, v)$

form a solution for the instance (G, ℓ, u, d) of Connected Network Microaggregation. The tuple Π is a partition of $V(G)$ since $B_1, \ldots, B_r$ is a partition of $V(H)$, and all vertices of $V(G) \setminus V(H)$ belong to the last cluster. The clusters are connected since B_i is connected for all $i \in [r]$, and the last cluster is connected via the vertex v. Every vertex except v is adjacent to v via an edge of length 1, and every cluster center is v, thus the distance from any vertex to its center is at most $d = 1$. The cluster sizes are within the required bounds since by definition of ECP for each $i \in [r], \ell = \lfloor \frac{n_H}{r} \rfloor \leq |B_i| \leq \lceil \frac{n_H}{r} \rceil = u$, and the size of the extra cluster is $|\{v, v_1, v_2, \ldots v_{\ell-1}\}| = \ell$.

Now to the other direction, let $\Pi = (C_1, C_2, \ldots C_m)$ and $\mathcal{C} = (c_1, c_2, \ldots c_m)$ be the solution to Connected Network Microaggregation. By reordering, we can assume that $v \in C_m$. Vertices $v_1, v_2, \ldots, v_{\ell-1}$ have v as their only neighbor, so they have to be in C_m as well. No other vertices can be in C_m since $|\{v, v_1, v_2, \ldots, v_{u-1}\}| = u$. Thus, C_m is exactly the vertices outside $V(H)$, and C_1, …, C_{m-1} is a partition of $V(H)$. By definition of Connected Network Microaggregation, for each $i \in [m-1]$, C_i is connected, and $\frac{n_H}{r} = \ell \leq |C_i| \leq u = \frac{n_H}{r}$. The latter also implies that $m - 1 = \frac{n_H}{n_H/r} = r$. Therefore, C_1, …, C_{m-1} is a solution to the instance (H, r) of ECP.

Finally, observe that the reduction takes polynomial time, and by removing the vertex v it is easy to see that $\mathrm{tw}(G) \leq \mathrm{tw}(H) + 1$. ◀

Next, we continue with showing W[1]-hardness reduction to NMA parameterized by the vertex cover number. The reduction is from Multicolored Independent Set (MIS). It consists of an undirected graph $G = (V, E)$, an integer $k \in \mathbb{N}$, and a function $c\colon V \to \{1, \ldots, k\}$, which is a proper k-coloring of the graph G. A set $X \subseteq V$ is called *multicolored* if for each pair of distinct vertices $v, w \in X$ it holds that $c(v) \neq c(w)$. In MIS, we have to decide whether there is a multicolored set $I \subseteq V$ of size k such that $G[I]$ is an edgeless graph. It is well-known that both problems are W[1]-hard parameterized by the number of colors k [30, 15].

Before we dive deep into the reduction, we define some basic notation for MIS. Let $\mathcal{I} = (G, c, k)$ be an instance of a multicolored problem. For a given color $i \in [k]$, we let V_i denote the set of vertices with color i, i. e., $V_i = \{v \in V(G) \mid c(v) = i\}$. Analogously, for a pair of distinct colors $i, i' \in [k]$ we denote by $E_{i,i'}$ the subset of edges between these two color classes, that is, $E_{i,i'} = \{\{v_i, v_{i'}\} \in E(G) \mid v_i \in V_i \wedge v_{i'} \in V_{i'}\}$.

▶ **Theorem 10.** *Network Microaggregation is* W[1]*-hard parameterized by the vertex-cover number* vc.

Proof. We reduce from the Multicolored Independent Set problem where the color classes are equally sized. The problem with this extra constraint is still W[1]-hard as any instance can be padded with vertices of missing colors that are connected to every other vertex. Assuming $k > 1$, these vertices are not in any independent set of size k and each independent set of the original graph is preserved. The blowup is still polynomial, as at most $\mathcal{O}(n^2)$ vertices were added. Let $\mathcal{I} = (G, c, k)$ be an instance of MIS and let $N = \frac{n}{k}$ be the size of every V_i.

The basic building block of our construction is *selection gadget*. We introduce exactly one copy of the selection gadget, denoted Q_i, for every color $i \in [k]$. The purpose of this gadget is to select one of the vertices colored i to the solution.

Let $i \in [k]$ be a fixed color and let $V_i = \{v_i^1, \ldots, v_i^N\}$. First, we introduce in Q_i an isolated vertex for every element of V_i. Slightly abusing the notation, we will also call these new vertices V_i. Then, we add two special vertices v_i^r and v_i^l which serve as connection ports with outsides of the gadget, and we call them *r-port* and *l-port*, respectively. For every $j \in [N]$, we add an edge $\{v_i^r, v_i^j\}$ of length $\frac{d}{3} + 1 + j$ and an edge $\{v_i^l, v_i^j\}$ of length $\frac{d}{3} + 1 - j$. Next, we add a *guard* vertex g_i that is connected to all vertices of V_i with edges of length $d - 1$. Finally, we introduce a set $A_i = \{a_i^1, \ldots, a_i^{n^2}\}$, where $n = |V(G)|$, of auxiliary vertices connected to guard vertex g_i with edges of length 1. The illustration of the selection gadget is depicted in Figure 3.1.

The new graph G' of the equivalent NMA instance $\mathcal{J}$ is then created as follows. For every color $i \in [k]$, we introduce one selection gadget Q_i as described above. Then, for every edge

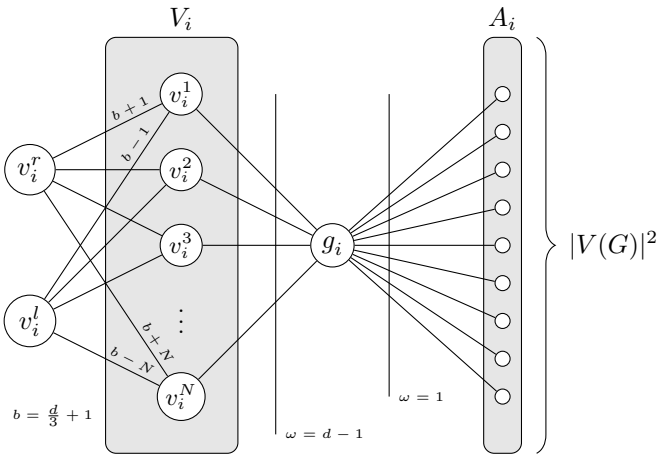

Figure 3.1 An illustration of the selection gadget for selection of a vertex colored $i \in [k]$ to the solution used in the proof of Theorem 10.

$\{v, w\} \in E(G)$, where $v = v_i^j$ and $w = v_{i'}^{j'}$, we add one *edge-vertex* $e_{v,w}$. This edge-vertex is then connected to

1. vertices v_i^r and v_i^l with edges of length $\frac{2d}{3} - j$ and $\frac{2d}{3} + j$, respectively, and
2. vertices $v_{i'}^r$ and $v_{i'}^l$ with edges of length $\frac{2d}{3} - j'$ and $\frac{2d}{3} + j'$, respectively.

Our intention is to have one cluster for each color class and select the vertex of the corresponding color as the cluster center. Thus, we set the size bounds as follows. In total, the new graph G' has $(|V(G)|^2 + N + 3) \cdot k + |E(G)|$ vertices, and we set $\ell = |V(G)|^2 + N + 3$ and $u = \ell + |E(G)|$. The lower bound ℓ is greater than $|E(G)|$, therefore, the solution does not have more than k clusters. At the same time, there must be at least k clusters due to the auxiliary vertices A_i which are too far from v_i^r and v_i^l, for every $i \in [k]$. Finally, to ensure that there are only integer and non-negative edge lengths, we set $d = 9N$.

Let $\mathcal{I}$ be a Yes -instance of MIS and $I = \{v_1^{j_1}, \ldots, v_k^{j_k}\}$, where $\forall i \in [k]\colon c(v_i^{j_i}) = i$, be a solution. We construct the solution for $\mathcal{J}$ as follows. For every $i \in [k]$, the cluster C_i contains the entire selection gadget Q_i and the center c_i of this cluster is an element corresponding to the vertex $v_i^{j_i}$. So far, in every cluster are exactly ℓ vertices. The distances to the cluster center are d for auxiliary vertices, $d-1$ for the guard vertex, at most $\frac{d}{3} + \frac{d}{3} - N + 1 < d$ for the vertices of V_i, and, finally, at most $\frac{d}{3} + 1 + N$ for the r-ports and the l-ports. What remains is to assign the edge-vertices to clusters. Let $e_{v,w}$, where $v = v_i^j$ and $w = v_{i'}^{j'}$, be an edge-vertex. Since the cluster centers form an independent set in G, it holds that at least one of v and w is not a cluster center. Without loss of generality, let $v \neq c_i$. Then we add $e_{v,w}$ to the cluster C_i. If neither v nor w are cluster centers, we place $e_{v,w}$ in an arbitrary cluster C_i or C_j. This increases the size of each cluster to at most $\ell + |E(G)| = u$ and, for every edge vertex, the distance to the cluster center is exactly d, due to the independence of the cluster centers.

In the opposite direction, let $\mathcal{J} = (G', \ell, u, d)$ be a Yes -instance of NMA and $\Pi = (C_1, \ldots, C_m)$ together with $\mathcal{C} = (c_1, \ldots, c_m)$ be a solution for $\mathcal{J}$. We already discussed that, due to our size bounds, it holds that $m = k$.

Now, we show that all cluster centers are elements of some V_i. Moreover, for every V_i, where $i \in [k]$, there is exactly one center c_i between its elements, as otherwise, we broke either the cluster size condition or the distances to the center.

▶ **Claim 8.** *Let $(\Pi, \mathcal{C})$ be a solution for $\mathcal{J}$. For each set V_i there is a cluster center $c_i \in \mathcal{C}$.*

Proof. The vertices that are less than d far from $a_i \in A_i$ are the vertices from V_i, guard vertex g_i and other auxiliary vertices from A_i. This forces each cluster center to be in a distinct selection gadget Q_i in either V_i or to be the guard vertex g_i. The distance between any two vertices from different selection gadgets is more than d, because the shortest the goes through an edge vertex which is connected only with edges of length $> \frac{d}{2}$. Therefore, the whole selection gadget is inside the same cluster where its center is, thus each cluster contains its own selection gadget. Since the distance from l-port and r-port to the guard vertex is at least $\frac{d}{3} + 1 - N + d - 1 > d$, the only remaining option for the cluster center is to be inside V_i. ■

We already verified that each vertex inside Q_i is at most d far from any vertex in V_i. What remains to show is the placement of edge-vertices. It is not hard to verify that, due to edge lengths, an edge-vertex representing an edge $e \in E_{i,i'}$ has to be part of either the cluster containing v_i^r or $v_{i'}^r$, as otherwise, we break the condition on the distance d to cluster center.

▶ **Claim 9.** *Let $(\Pi, \mathcal{C})$ be a solution for $\mathcal{J}$ and $e_{v,w}$, where $v = v_i^j$ and $w = v_{i'}^{j'}$, be an edge-vertex. Then $e_{v,w}$ is an element of a cluster $C \in \{C_i, C_{i'}\}$ with cluster center $c \notin \{v, w\}$.*

Proof. Let $e_{v,w}$ be an edge-vertex that is an element of a cluster C such that, without loss of generality, $v = v_i^j$ is the cluster center c. Then the distance to the center of the cluster is

$$\text{dist}(e_{v,w}, c) = \frac{2d}{3} - j + \frac{d}{3} + 1 + j = d + 1.$$

This contradicts that $(\Pi, \mathcal{C})$ is a solution. Hence, edge-vertex cannot be in a cluster where the center corresponds to one end-point of this edge. ■

We assumed that $\mathcal{J}$ is a Yes -instance. By Claim 9, we have that every edge-vertex is a member of a cluster where the cluster center is not one of the edge endpoints. Moreover, such a cluster exists for every edge-vertex. Hence, we obtain a solution for a MIS instance $\mathcal{I}$ by taking as I the vertices forming cluster centers. There are exactly k of these and every cluster center corresponds to a vertex of a different color thanks to Claim 8 and these vertices form an independent set in G.

It is easy to see that the set $\bigcup_{i=1}^{k} \{v_i^r, v_i^l, g_i\}$ is of size $3k$ and forms a vertex cover of G', as $\{v_i^r, v_i^l, g_i\}$ is vertex-cover for the selection gadget Q_i, $i \in [k]$, and the only connections outside the selection gadget are realized via v_i^r and v_i^l. Hence, the presented reduction is indeed a parameterized reduction and the theorem follows. ◀

We note that a highly similar construction as the one used above also works for the connected variant of the problem.

Chapter 4

Treedepth hardness for ECP

In this chapter, we improve the result shown in Theorem 9 and start classifying the complexity of Connected Network Microaggregation w.r.t. to other parameters between vc and tw. We ask a question if Connected Network Microaggregation is W[1]-hard parameterized by tw $+d$, and FPT when parameterized by vc $+d$ Theorem 7 when does the problem becomes W[1]-hard? Surprisingly, the answer can be easily answered by providing a tighter parameterized intractability result for the problem we have reduced from.

First, we recall the hardness reduction of Enciso et al. [29, Theorem 1] as it serves as a basic building block for other W[1]-hardness reductions presented in this section. The reduction is reducing from the problem Multicolored Clique. For each one of the k colors, there is a vertex selection gadget made of $2(k-1)$ anchors, each enforces its own partition by having many pendant leaves. Anchors send between themselves signals through choice gadgets, which is just a path with pendant leaves appended in such a way that only wanted values of the signals are possible. To agree on a single vertex in one vertex selection gadget, the anchors are arranged on a cycle If one anchor takes x vertices from a choice gadget, it forces the neighboring anchor to take x vertices from the next anchor to have its partition of the correct size. Two vertex selection gadgets agree on an edge by dedicating two anchors for each color and putting them with two corresponding anchors of that color on a cycle made of choice gadgets. Because this cycle shares one choice gadget with both vertex selection gadgets, they are able to filter only the signals corresponding to a vertex incident to an edge to the other color. Which in the complete picture forces the solution to represent a multicolored clique.

Now, we observe that ECP is W[1]-hard with respect to the feedback edge set number of G. This resolves negatively the question from the introduction of our paper. The actual statement shows an even stronger intractability result.

▶ **Observation 11.** *Equitable Connected Partition is* W[1]*-hard with respect to the path-width* pw*, the feedback-edge set number* fes*, and the number of parts* p *combined.*

Proof. Enciso et al. [29] show in their basic reduction that the ECP problem is W[1]-hard parameterized by the pathwidth, the feedback-vertex set, and the number of parts combined. One can observe, that the same reduction also shows W[1]-hardness for the pathwidth, the feedback-edge set, and the number of parts combined. To see this, we recall that there are k selection gadgets, each containing $2(k-1)$ anchors. That is, there are $2k(k-1)$ anchors in total. Moreover, the number of parts p is equal to the number of anchors. By removing the roots of every anchor, we obtain a graph which is a collection of paths with pending vertices, thus, the pathwidth is at most number of roots plus one. Finally, for the feedback-edge set number, recall that every selection gadget for color i consists of $2(k-1)$ anchors N^i_j and P^i_j, where $1 \leq j \leq k$ and $i \neq j$ connected into a cycle using choice gadgets. By removing the edges on the cycle

adjacent to the anchor's roots, we obtain a collection of trees, each made of a choice gadget and two anchors. Therefore, the feedback-edge set of the construction is at most $4k(k-1)$. ◀

Note that the result from Observation 11 can be, following the same arguments, strengthened to show a similar result for planar graphs.

Next, we show that the d-pvcn parameter cannot be relaxed anymore while keeping the problem tractable. Our reduction is from the Partitioned Subgraph Isomorphism problem, which is defined as follows.

Partitioned Subgraph Isomorphism (PSI)	
Input:	Two undirected graphs G and H with $\|V(H)\| \leq \|V(G)\|$ and a mapping $\psi\colon V(G) \to V(H)$.
Question:	Is there an injective mapping $\phi\colon V(H) \to V(G)$ such that $\{\phi(u), \phi(v)\} \in E(G)$ for each $\{u, v\} \in E(H)$ and $\psi \circ \phi$ is the identity?

Since special case of PSI called Partitioned Clique, where H is a complete graph, is known to be W[1]-complete parameterized by the clique size [31], it follows that PSI is W[1]-hard parameterized by $|E(H)|$.

▶ **Theorem 12.** *Equitable Connected Partition is* W[1]*-hard parameterized by the 4-*pvcn.

Proof. Our reduction is, from the high-level perspective, the same as the one of Enciso et al. [29]. However, to be able to show hardness for the parameter 4-path vertex cover number, the concrete realization of gadgets is quite different. For the sake of completeness, let us present the reduction in its entirety.

A basic building block of our construction is an *anchor*. An anchor, denoted by $\text{anch}(r, \ell)$, is a simple star with a root r and $\ell - 1$ leaves. Every anchor is connected with the rest of the graph by its root r only, hence, by setting the number of leaves of an anchor we can affect the number of additional vertices in the part together with this anchor.

Anchors will be interconnected via so-called *choice gadgets*. A choice gadget for a multiset $S = \{s_1, \ldots, s_t\}$, denoted $\text{choice}(S)$, is a bipartite graph. We start the construction of $\text{choice}(S)$ with two *ports* $v^\top$ and $v^\bot$. Next, we create an *element-vertex* v_i for every $s_i \in S$. For every $i \in [t]$, an element-vertex v_i is neighbor of both $v^\top$ and $v^\bot$ and v_i has $s_i - 1$ pendant leaves.

Let A and B be two anchors with roots r_A and r_B, respectively. We connect A to B by a choice-gadget $\text{choice}(S)$ by identifying $v^\top$ with r_A and $v^\bot$ with r_B. For an illustration of anchors and choice gadgets, we refer the reader to Figure 4.1.

Let $\mathcal{I} = (G, H, \psi)$ be a PSI instance. We construct an equivalent ECP instance $\mathcal{J} = (G', p)$ as follows. Let $k := |V(H)|$, $n := |V(G)|$, $N := n + 1$ and let ID be a function that assigns a unique ID from $[n]$ to every vertex in $V(G)$ and a unique ID from $[\binom{n}{2}]$ to every unordered pair of vertices of G. Furthermore, we define a *successor* of $j \in [k]$ to be $s(j) := (j + 1)$ if $j < k$ and $s(k) := 1$.

We start by creating k *vertex selection gadgets*, one for each vertex $i \in [k]$. These gadgets will select one vertex from $\psi^{-1}(i)$. Each of these gadgets is made of $2(k-1)$ anchors labeled as $A^\top_{i,j} = \text{anch}(v^\top_{i,j}, \ell^\top_{i,j})$, $A^\bot_{i,j} = \text{anch}(v^\bot_{i,j}, \ell^\bot_{i,j})$ for $j \in [k] \setminus \{i\}$. We label their roots as $v^\top_{i,j}$ and $v^\bot_{i,j}$, respectively.

Similarly to Enciso et al. [29], each anchor forces its own partition and anchors communicate with other anchors through signals carried by the choice gadgets. There are 2 different strengths of signals being carried. Between the ports $v^\bot_{i,j}, v^\top_{i,s(j)}$ there is a $C^V_{i,j} := \text{choice}(\{1 \mid v \in V(G)\})$, where the vertex is selected by picking that many stars in the gadget as is the vertex ID. Similarly, there is a $C^E_{i,j} := \text{choice}(\{N \mid e \in E(G)\})$ between $v^\bot_{j,i}, v^\top_{i,j}$. These choice gadgets enable two different vertex selection gadgets to agree on an edge or a non-edge.

Finally, the third type of choice gadget $C^C_{i,j}$ is between $v^\top_{i,j}$ and $v^\bot_{i,j}$ which allows both the vertex and edge selecting signal to be carried simultaneously and at the same time verify that

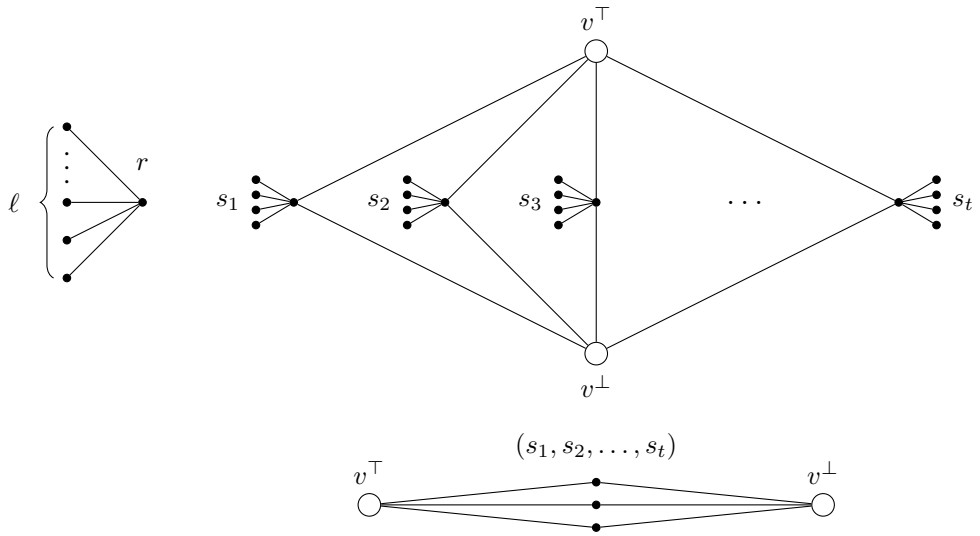

Figure 4.1 An anchor (left) and a choice gadget (right) with its schematic representation used in further construction illustrations (bellow).

there is an edge between chosen vertices if $\{i,j\} \in E(H)$. The vertex and edge selecting signals do not interfere because the smallest change of the bigger signal is larger than the maximum size of the small signal. We use the following choice gadget $C^C_{i,j} := \text{choice}(\{\text{ID}(u) + N \cdot \text{ID}(u,v) + N^3 \mid u,v \in V(G) \wedge \psi(u) = i \wedge \psi(v) = j \wedge (\{i,j\} \in E(H) \implies \{u,v\} \in E(G))\})$. The first part $\text{ID}(u) + N \cdot \text{ID}(u,v)$ allows only the desired vertex and edge selection signals to pass through to the next anchor. With the extra N^3 we ensure that we select only one value from the choice gadget $C^C_{i,j}$ by the partition containing $A^\top_{i,j}$ as we show now.

The size of each partition is chosen to be $r := n + N \cdot \binom{n}{2} + S$, where $S := \sum_{(u,v) \in V(G)^2} (\text{ID}(u) + N \cdot \text{ID}(u,v) + N^3)$. Based on that, we set the sizes of anchors $A^\top_{i,j}$ as $\ell^\top_{i,j} := S - N^3$ and size of $A^\perp_{i,j}$ as $\ell^\perp_{i,j} := S - S_{i,j} + N^3 + n + N \cdot \binom{n}{2}$, where $S_{i,j} := \sum_{\{u,v\} \in \phi^{-1}(\{i,j\})} (ID(u) + N \cdot \text{ID}(u,v) + N^3)$. Finally, we notice that there are $k(2k-2) \cdot (r)$ vertices in $V(G')$.

The connections of the anchors via choice gadgets can be seen in Figure 4.2.

▶ **Claim 10.** *If $\mathcal{I} = (G, H, \psi)$ is a* Yes *-instance then $\mathcal{J} = (G', p)$ is a* Yes *-instance.*

Proof. We create $p = 2k(k-1)$ partitions, one for each anchor labeled as $V^\top_{i,j}$ for anchor $A^\top_{i,j}$ and $V^\perp_{i,j}$ for $A^\perp_{i,j}$.

For each $i,j \in V(H), i \neq j$ we assign $\text{ID}(\phi(i))$ vertices in choice gadget $C^V_{i,j}$ to a partition $V^\perp_{i,j}$. The rest of the vertices in the choice gadget we assign to partition $V^\top_{i,s(j)}$. Similarly, for each $i,j \in V(H)$, where $i \neq j$, we assign $\text{ID}(\phi(i), \phi(j))$ stars with pendant vertices from the choice gadget $C^E_{i,j}$ to the partition $V^\perp_{i,j}$. The rest we will assign to $V^\perp_{i,j}$.

For the last type of the choice gadget $C^C_{i,j}$ with $i,j \in V(G), i \neq j$, we choose a single star that contains $\phi(i) + \text{ID}(\phi(i), \phi(j)) \cdot N + N^3$ vertices and assign it to $V^\top_{i,j}$.

We know there is such a star since $C^C_{i,j}$ contains all possible value combinations of a vertex in $\psi(i)^{-1}$ and an edge $\psi(i,j)^{-1}$, or just the vertex if the edge does not exist. Every other star in this choice gadget is assigned to $V^\perp_{i,j}$.

Now we verify that there are exactly $r = \frac{V(G')}{p}$ vertices in each partition. Each partition $V^\top_{i,j}$ contains $(n - \phi(i)) + N \cdot (\binom{n}{2} - \text{ID}(\phi(i), \phi(j))) + \phi(i) + N \cdot \text{ID}(\phi(i), \phi(j)) + N^3 = n + N \cdot \binom{n}{2} + N^3$ vertices from choice gadgets incident to its anchor. With the remaining $\ell^\top = S - N^3$ vertices from being an anchor, the partition size is exactly $n + \binom{n}{2} \cdot N + S = r$. For the partition $V^\perp_{i,j}$

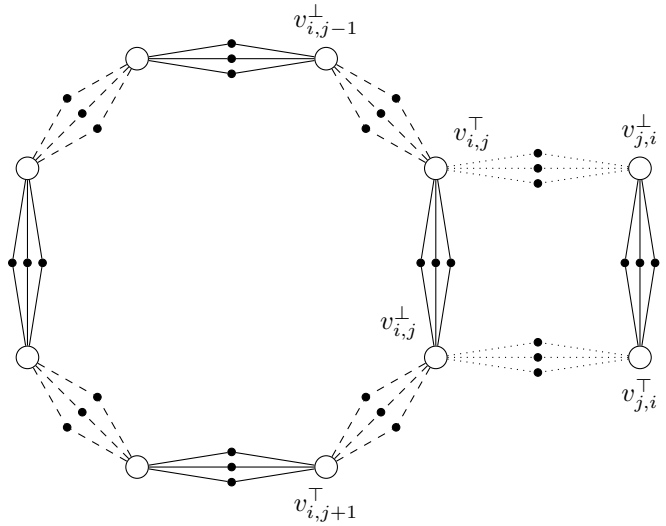

Figure 4.2 An overview of the construction used in the proof of Theorem 12. The dashed line represents a choice gadget carrying the signal selecting a vertex, the dotted line carries the signal selecting an edge and the full line is used for the combined signal.

there are $\phi(i) + N \cdot \mathrm{ID}(\phi(i), \phi(j)) + (S_{i,j} - (i + N \cdot \mathrm{ID}(\phi(i), \phi(j)) + N^3)) = S_{i,j} - N^3$ vertices from the incident choice gadgets. Together with $\ell^{\perp} := S - S_{i,j} + N^3 + n + N \cdot \binom{n}{2}$ vertices from the anchor, we get exactly $n + \binom{n}{2} \cdot N + S = r$ vertices as well.

Therefore, all partitions have the correct size, and they are connected, therefore $\mathcal{J}$ is a Yes -instance. ■

▶ **Claim 11.** *If $\mathcal{J} = (G', p)$ is a* Yes *-instance then $\mathcal{I} = (G, H, \psi)$ is a* Yes *-instance.*

Proof. Let $\pi = \{V_1, \ldots, V_p\}$ be the partitioning of $V(G')$. Every partition contains exactly one anchor. It is easy to see that every pendant leaf is in the same partition as his only neighbor with $r > 1$. Every value in a choice gadget is a star that is connected only to two anchors. Since it contains at most $N^3 + N \cdot \binom{n}{2} + n < 2N^3$ vertices and $r > S > |V(G)| \cdot N^3 > 2N^3$, it has to share partition with one of the neighboring anchors. In fact, there is exactly one anchor per partition, because two anchors sharing a choice gadget would have at least $\ell^{\top} + \ell^{\perp} = S - N^3 + S - S_{i,j} + N^3 + n + N \cdot \binom{n}{2} = 2S - S_{i,j} + n + N \cdot \binom{n}{2} > r$ since $S - S_{i,j} > 0$.

Let $V^{\top}_{i,j}, V^{\perp}_{i,j}$ be the partitions containing the anchors $A^{\top}_{i,j}$ and $A^{\perp}_{i,j}$, respectively. Since $r - \ell^{\top}_{i,j} > N^3$, at least one value from the choice gadget $C^C_{i,j}$ has to be inside $V^{\top}_{i,j}$. In fact only a single star from the choice gadget $C^C_{i,j}$ can be inside the partition $V^{\top}_{i,j}$, otherwise the size of $V^{\top}_{i,j}$ would be at least $\ell^{\top}_{i,j} + 2N^3 = S - N^3 + 2N^3 > r$. The selected star determines corresponding vertices $u \in V_i, v \in V_j$ from the value $X = \mathrm{ID}(u) + N \cdot \mathrm{ID}(u, v) + N^3$ because $X \mod N = \mathrm{ID}(u)$ and $\frac{X - \mathrm{ID}(u)}{N} \mod N^2 = \mathrm{ID}(u, v)$. The remainder of $r - \ell^{\top}_{i,j}$ when divided by N is n and values in $C^E_{i,j}$ are multiples of N, therefore they cannot fill the offset $\mathrm{ID}(u)$ from selecting X. The only possible way for $r \mod N = |V_i| \mod N = n$ is to have exactly $n - \mathrm{ID}(u)$ many vertices from $C^V_{i,j-1}$ in $V^{\top}_{i,j}$. Similarly, $\frac{X - \mathrm{ID}(u)}{N} mod N^2 = \mathrm{ID}(u, v)$ and $\frac{r - \ell^{\top}_{i,j} - n}{N} \mod N^2 = \binom{n}{2}$. There are only $n < N$ vertices in $C^V_{i,j}$, therefore $\binom{n}{2} - \mathrm{ID}(u, v)$ many stars from $C^E_{j,i}$ are present in $V^{\top}_{i,j}$. The neighboring partition $V^{\perp}_{i,j}$ is forced to choose $\mathrm{ID}(u)$ and $\mathrm{ID}(u, v)$ many vertices in $C^V_{i,j}$ and $C^E_{i,j}$, respectively, with the same argument about the modulo of r. With the additional $\ell^{\top}_{i,j}$ or $\ell^{\perp}_{i,j}$ vertices from their anchor, there are exactly r vertices in $V^{\top}_{i,j}$ and $V^{\top}_{j,i}$ as calculated in previous claim.

Every other partition in the i-th vertex selection gadget selects $u \in \psi^{-1}(i)$ as well, because the preceding partition forces it to take $n - \mathrm{ID}(u) \mod N$ vertices and thus it has to take $\mathrm{ID} \mod N$ vertices in the next choice gadget. So we specify that $\phi(i) = u$. Similarly, the edge selection gadget selects a pair of vertices $u \in \psi^{-1}(i), v \in \psi^{-1}(j)$ such that $\phi(i) = u$ and $\phi(j) = v$. For two vertices $i, j \in V(H)$ the edge $\{i, j\} \in E(H) \implies \{\phi(i), \phi(j)\} \in E(G)$ because only these pairs of vertices were allowed in the choice gadget $C^C_{i,j}$. Therefore, $\mathcal{I}$ is a Yes -instance. ■

The reduction runs in polynomial time since the graph contains $\mathcal{O}(k^2n^5)$ vertices.

◀

It is not hard to see that if the 4-pvcn is bounded, then so is the treedepth td of the graph. The graph without paths of length more than k has treedepth at most k [19, Proposition 6.1]. Removing 4-pvcn vertices from G could decrease td of G by at most 4-pvcn as every removed vertex can remove only one layer in the tree depth decomposition. Therefore, $\mathrm{td} \leq k + \text{k-pvcn}$ and hence, we directly obtain.

▶ **Corollary 13.** *Equitable Connected Partition is* W[1]*-hard parameterized by the treedepth* td.

Recall, that in our reduction from ECP in Theorem 9 we added a single vertex v connected to every vertex in G and then added pendant vertices to it. The new graph G' has 4-pvcn increase by at most one from G since removing the same vertices in G plus the vertex v gives us a graph without paths on 4 vertices. Finally, we present our last result.

▶ **Corollary 14.** *Connected Network Microaggregation is* W[1]*-hard parameterized by* 4-pvcn *or* td*, even if* $d = 1$.

Conclusion

The thesis presented in this work focuses on two clustering problems, namely Network Microaggregation and Connected Network Microaggregation, and explores their parameterized complexity. By applying various techniques, we were able to provide both tractability and intractability results for these problems.

In terms of tractability, we showed that Connected Network Microaggregation is in FPT when parameterized by the treewidth, distance to cluster center, and upper-bound on the cluster size. Moreover, we demonstrated that Connected Network Microaggregation is in XP when parameterized only by the treewidth with a dynamic programming approach. We also designed FPT algorithms for both Connected Network Microaggregation and Network Microaggregation using exhaustive search and ILP for the vertex cover number and distance to the cluster center. However, there is still room for improvement with these algorithms, as it is unclear whether they can be further optimized or their tightness can be proven using tools such as the exponential time hypothesis.

On the other hand, we presented W[1]-hardness reductions to show the intractability of the problems with certain parameterizations. Specifically, we demonstrated that Connected Network Microaggregation is W[1]-hard with respect to the treewidth and the distance to the cluster center using a reduction from Equitable Connected Partition, and that Network Microaggregation is W[1]-hard parameterized by the vertex cover number using a reduction from Multicolored Clique. Additionally, we showed that the well-known problem Equitable Connected Partition is W[1]-hard when parameterized by treedepth, which immediately carried over to our reduction from Equitable Connected Partition and established W[1]-hardness when parameterized by treedepth and distance to cluster center.

Similarly, in many other cases of the problem, there is an FPT algorithm for vertex cover plus additional parameters, but when we exchange vc for tw we get W[1]-hardness. Thus, a natural question arises. How does the problem behave between those two parameters in other cases? Other parameterization might be interesting as well, depending upon what parameters happen to be small in the real world.

From the problem specific parameters, we only considered the distance to the cluster center and upper-bound on the size of the cluster. We did not consider the lower-bound as it is smaller or equal to the upper bound and in Network Microaggregation u is effectively at most $2\ell - 1$,

Lastly, we have shown two versions of the problem, but other slight variations might make sense. For example, we might want the cluster centers to be in its own cluster center to better represent the group. Alternatively, there does not have to be centers at all and the coherency of clusters may be maintained by a maximal pairwise distance between members of each cluster.

Bibliography

1. DOMINGO-FERRER, Josep; TORRA, Vicenç. Ordinal, continuous and heterogeneous k-anonymity through microaggregation. *Data Mining and Knowledge Discovery*. 2005, vol. 11, pp. 195–212.
2. RODRIGUEZ-HOYOS, Ana; ESTRADA-JIMÉNEZ, José; REBOLLO-MONEDERO, David; PARRA-ARNAU, Javier; FORNÉ, Jordi. Does k-Anonymous microaggregation affect machine-learned macrotrends? *IEEE access*. 2018, vol. 6, pp. 28258–28277.
3. THAETER, Florian; REISCHUK, Rüdiger. Hardness of k-anonymous microaggregation. *Discrete Applied Mathematics*. 2021, vol. 303, pp. 149–158. Available from DOI: 10.1016/j.dam.2020.10.005.
4. DOMINGO-FERRER, Josep; TORRA, Vicenc. A quantitative comparison of disclosure control methods for microdata. *Confidentiality, disclosure and data access: theory and practical applications for statistical agencies*. 2001, pp. 111–134.
5. DOMINGO-FERRER, Josep; SEBÉ, Francesc; SOLANAS, Agusti. A polynomial-time approximation to optimal multivariate microaggregation. *Computers & Mathematics with Applications*. 2008, vol. 55, no. 4, pp. 714–732. ISSN 0898-1221. Available from DOI: https://doi.org/10.1016/j.camwa.2007.04.034.
6. SUN, Xiaoxun; WANG, Hua; LI, Jiuyong; ZHANG, Yanchun. An Approximate Microaggregation Approach for Microdata Protection. *Expert Systems with Applications*. 2012, vol. 39, no. 2, pp. 2211–2219. ISSN 0957-4174. Available from DOI: 10.1016/j.eswa.2011.04.223.
7. ABU-KHZAM, Faisal N.; BAZGAN, Cristina; CASEL, Katrin; FERNAU, Henning. Clustering with Lower-Bounded Sizes. *Algorithmica*. 2018, vol. 80, no. 9, pp. 2517–2550. ISSN 0178-4617. Available from DOI: 10.1007/s00453-017-0374-5.
8. CASEL, Katrin. Resolving Conflicts for Lower-Bounded Clustering. In: *Proceedings of the 13th International Symposium on Parameterized and Exact Computation, IPEC '18*. 2019, vol. 115, 23:1–23:14. LIPIcs. ISSN 1868-8969. Available from DOI: 10.4230/LIPIcs.IPEC.2018.23.
9. BLAŽEJ, Václav; GANIAN, Robert; KNOP, Dušan; POKORNÝ, Jan; SCHIERREICH, Šimon; SIMONOV, Kirill. The Parameterized Complexity of Network Microaggregation. In: *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023*. AAAI Press, 2023.
10. ARORA, Sanjeev; BARAK, Boaz. *Computational complexity: a modern approach*. Cambridge University Press, 2009.
11. TURING, Alan M. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc.* 1937, vol. s2-42, no. 1, pp. 230–265. Available from DOI: 10.1112/plms/s2-42.1.230.

12. TURING, Alan M. Computability and λ-Definability. *J. Symb. Log.* 1937, vol. 2, no. 4, pp. 153–163. Available from DOI: 10.2307/2268280.
13. COOK, Stephen A. The Complexity of Theorem-Proving Procedures. In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. Shaker Heights, Ohio, USA: Association for Computing Machinery, 1971, pp. 151–158. STOC '71. ISBN 9781450374644. Available from DOI: 10.1145/800157.805047.
14. KARP, Richard M. Reducibility Among Combinatorial Problems. In: MILLER, Raymond E.; THATCHER, James W. (eds.). *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*. Plenum Press, New York, 1972, pp. 85–103. The IBM Research Symposia Series. Available from DOI: 10.1007/978-1-4684-2001-2_9.
15. CYGAN, Marek; FOMIN, Fedor V.; KOWALIK, Łukasz; LOKSHTANOV, Daniel; MARX, Dániel; PILIPCZUK, Marcin; PILIPCZUK, Michał; SAURABH, Saket. *Parameterized Algorithms*. Springer, 2015. ISBN 978-3-319-21274-6. Available from DOI: 10.1007/978-3-319-21275-3.
16. DOWNEY, Rodney G.; FELLOWS, Michael R. Fixed-Parameter Tractability and Completeness II: On Completeness for W[1]. *Theor. Comput. Sci.* 1995, vol. 141, no. 1&2, pp. 109–131. Available from DOI: 10.1016/0304-3975(94)00097-3.
17. FLUM, Jörg; GROHE, Martin. *Parameterized Complexity Theory*. Springer, 2006. Texts in Theoretical Computer Science. An EATCS Series. ISBN 978-3-540-29952-3. Available from DOI: 10.1007/3-540-29953-X.
18. BODLAENDER, Hans L.; DRANGE, Pål Grønås; DREGI, Markus S.; FOMIN, Fedor V.; LOKSHTANOV, Daniel; PILIPCZUK, Michał. A $c^k n$ 5-Approximation Algorithm for Treewidth. *SIAM Journal on Computing*. 2016, vol. 45, no. 2, pp. 317–378. Available from DOI: 10.1137/130947374.
19. NEŠETŘIL, Jaroslav; OSSONA DE MENDEZ, Patrice. *Sparsity: Graphs, Structures, and Algorithms*. Vol. 28. Berlin, Heidelberg: Springer, 2012. Algorithms and Combinatorics. ISBN 978-3-642-27874-7. ISSN 0937-5511. Available from DOI: 10.1007/978-3-642-27875-4.
20. SORGE, Manuel. *The graph parameter hierarchy*. Available also from: https://manyu.pro/assets/parameter-hierarchy.pdf. Accessed on 26.06.2023.
21. KIRKPATRICK, David G.; HELL, Pavol. On the Completeness of a Generalized Matching Problem. In: *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, STOC '78*. San Diego, California, USA, 1978, pp. 240–245. Available from DOI: 10.1145/800133.804353.
22. FREDMAN, Michael L.; TARJAN, Robert Endre. Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *Journal of the ACM*. 1987, vol. 34, no. 3, pp. 596–615. ISSN 0004-5411. Available from DOI: 10.1145/28869.28874.
23. BODLAENDER, Hans L. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM Journal on Computing*. 1996, vol. 25, no. 6, pp. 1305–1317. Available from DOI: 10.1137/S0097539793251219.
24. KORHONEN, Tuukka. A Single-Exponential Time 2-Approximation Algorithm for Treewidth. In: *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS '21*. 2021, pp. 184–192. Available from DOI: 10.1109/FOCS52979.2021.00026.
25. LENSTRA JR., H. W. Integer programming with a fixed number of variables. *Mathematics of Operations Research*. 1983, vol. 8, no. 4, pp. 538–548.
26. KANNAN, Ravi. Minkowski's Convex Body Theorem and Integer Programming. *Mathematics of Operations Research*. 1987, vol. 12, no. 3, pp. 415–440. Available from DOI: 10.1287/moor.12.3.415.

27. FRANK, András; TARDOS, Éva. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica*. 1987, vol. 7, no. 1, pp. 49–65.

28. CHEN, Jianer; KANJ, Iyad A; XIA, Ge. Improved parameterized upper bounds for vertex cover. In: *Mathematical Foundations of Computer Science 2006: 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006. Proceedings 31*. Springer, 2006, pp. 238–249.

29. ENCISO, Rosa; FELLOWS, Michael R.; GUO, Jiong; KANJ, Iyad; ROSAMOND, Frances; SUCHÝ, Ondřej. What Makes Equitable Connected Partition Easy. In: *Proceedings of the 4th International Workshop on Parameterized and Exact Computation, IWPEC '09*. 2009, vol. 5917, pp. 122–133. LNCS.

30. FELLOWS, Michael R.; HERMELIN, Danny; ROSAMOND, Frances; VIALETTE, Stéphane. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*. 2009, vol. 410, no. 1, pp. 53–61. ISSN 0304-3975. Available from DOI: 10.1016/j.tcs.2008.09.065.

31. PIETRZAK, Krzysztof. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *Journal of Computer and System Sciences*. 2003, vol. 67, no. 4, pp. 757–771. Available from DOI: 10.1016/S0022-0000(03)00078-3.