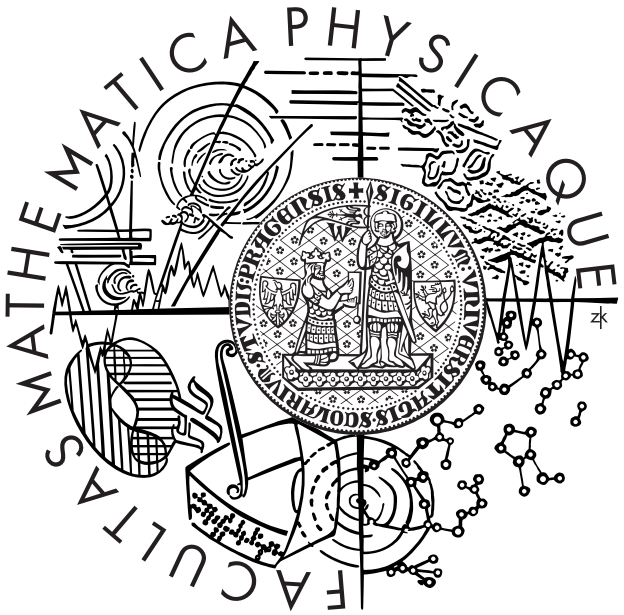
Charles University in Prague
Faculty of Mathematics and Physics

Doctoral Thesis



Ondřej Suchý

Parameterized Complexity

Nonstandard Parameterizations of Graph Problems

Department of Applied Mathematics

Advisor: **Prof. RNDr. Jan Kratochvíl, CSc.**
Study programme: **4I4 - Discrete Models and Algorithms**

Acknowledgements

I am very grateful to my advisor, Jan Kratochvíl, for many useful advises regarding the research, writing papers and many other areas. I would also like to thank the Department of Applied Mathematics and the Institute for Theoretical Computer Science for a generous support and the colleagues from there for the nice atmosphere and also for being a good company in many activities not directly related to the research. This holds also for the other institutions I have had the opportunity to stay on, especially Friedrich Schiller University of Jena and the University of Bergen and the people I met there for providing a gracious welcome and for the great time spent together. Many thanks go also to all my coauthors, in particular to Rolf Niedermeier and Mike Fellows for teaching me a lot, not only about the parameterized complexity. Last but not least I want thank to my family and especially Alena for all the support of various kind, which could hardly be expressed in words, and for being patient with me. Finally, I also want to thank all who deserve it, but I forgot to mention them.

I hereby declare, that I wrote the thesis on my own and that I have included all the sources of information which I used to the references. I also authorize the Charles University to lend this document to other institutions or individuals for academic and research purposes.

Prague, November 2010 Ondřej Suchý

Contents

Abstracts

Title: Parameterized Complexity

Author: Ondřej Suchý

Department: Department of Applied Mathematics

Advisor: Prof. RNDr. Jan Kratochvíl, CSc.

Advisor's e-mail address: honza@kam.mff.cuni.cz

Abstract: This thesis deals with the parameterized complexity of NP-hard graph problems. We explore the complexity of the problems in various scenarios, with respect to miscellaneous parameters and their combinations. Our aim is rather to classify in this multivariate manner whether the particular parameters make the problem fixed-parameter tractable or intractable than to present the algorithm achieving the best running time. In the questions we study typically the first-choice parameter is unsuccessful, in which case we propose to use less standard ones.

The first family of problems investigated provides a common generalization of many well known and studied domination and independence problems. Here we suggest using the dual parameterization and show that, in contrast to the standard solution-size, it can confine the inevitable combinatorial explosion. Further studied problems are analogues of the Steiner problem in directed graphs. Here the parameterization by the number of terminals to be connected seems to be previously unexplored in the directed setting. Unfortunately, the problems are shown to be intractable with respect to this parameter. Finally, the problems of partitioning the graph into classes of the same size, satisfying some further constraints, are considered. The problems turn out to be one of a few which are polynomial-time solvable on graphs of bounded treewidth, but not fixed-parameter tractable. More fine-grained structural parameterizations are then employed and proved to be successful.

Keywords: parameterized complexity, graph, Steiner problem, generalized domination, equitable partitions

Název práce: Parametrizovaná složitost

Autor: Ondřej Suchý

Katedra (ústav): Katedra aplikované matematiky

Školitel: Prof. RNDr. Jan Kratochvíl, CSc.

e-mail školitele: honza@kam.mff.cuni.cz

Abstrakt: Tato práce se zabývá parametrizovanou složitostí NP-těžkých grafových problémů. Zkoumáme složitost problémů v různých scenářích, vzhledem k rozličným parametrům a jejich kombinacím. Naším cílem je spíše rozlišit v tomto mnohorozměrném smyslu, zda daný parametr dělá problém parametrizovaně dostupným, nebo nedostupným, než představit algoritmus, který dosahuje nejlepší možné časové složitosti. V otázkách, které studujeme, je typicky parametr první volby neúspěšný a tak využíváme méně standardních parametrů.

První zkoumaná množina problémů je společným zobecněním mnoha dobře známých a prostudovaných problémů dominance a nezávislosti. Navrhujeme zde použít duální parametrizaci a ukážeme, že narozdíl od standardní parametrizace velikostí řešení, tato parametrizace dokáže ohrančit nevyhnutelnou kombinatorickou explozi. Další studované problémy jsou analogií Steinerova problému v orientovaných grafech. Parametrizace pomocí počtu terminalů se jeví jako dříve neprobádaná alternativa v orientovaném prostředí. Bohužel ukážeme, že zkoumané problémy jsou nedostupné vzhledem k tomuto parametru. Na problémy dělení grafu na stejně velké části, které splňují nějaké další podmínky, se zaměřujeme v poslední části. Ukazuje se, že tyto problémy jsou jedněmi z mála, které jsou na grafech omezené stromové šířky sice řešitelné v polynomialním čase, ale nejsou parametrizovaně dostupné. Využijeme tedy jemnějších strukturálních parametrizací a ukážeme, že tyto jsou úspěšné.

Klíčová slova: Parametrizovaná složitost, graf, Steinerův problém, zobecněná dominance, vyrovnaná rozdělení

List of My Research Publications

This thesis is mainly based on the following papers:

1. J. Guo, R. Niedermeier, and O. Suchý: *Parameterized Complexity of Arc-Weighted Directed Steiner Problems.*
   ISAAC 2009, LNCS vol. 5878, pp. 544-553, 2009
   journal version to appear in SIAM J. on Discrete Math

2. P. Golovach, J. Kratochvíl and O. Suchý: *Parameterized Complexity of Generalized Domination Problems.*
   WG 2009, LNCS vol. 5911, pp. 133-142, 2010
   journal version to appear in Disc. App. Math.

3. R. Enciso, M. R. Fellows, J. Guo, I. Kanj, F. Rosamond and O. Suchý: *What Makes Equitable Connected Partition Easy.*
   IWPEC 2009, LNCS vol. 5917, pp. 122-133, 2010

Further papers, not being a part of the thesis:

4 O. Suchý: *Some Parameterized Problems Related to Seidel's Switching.*
IWOCA 2007, pp. 148-157, College Publications, United Kingdom, 2008.
journal version submitted

5 E. Jelínková, J. Kára, J. Kratochvíl, M. Pergel, O. Suchý, and T. Vyskočil: *Clustered Planarity: Small Clusters in Eulerian Graphs.*
GD 2007, LNCS, vol. 4875, pp. 303-314, 2008.
journal version: JGAA, Special Issue on GD 2007, Vol. 13, no. 3, pp. 379-422, 2009.

6 V. Jelínek, O. Suchý, M. Tesař and T. Vyskočil: *Clustered Planarity: Clusters with Few Outgoing Edges.*
GD 2008, LNCS vol. 5417, pp. 102-113, 2009.

Chapter 1

Introduction

Classical complexity treats the problems according to whether they admit an algorithm solving them in polynomial time in terms of the input size or not. Unfortunately, many (if not most) interesting problems turned to be NP-complete, which means that a polynomial algorithm optimally solving them is fairly improbable.

This turned the attention to approximation algorithms which should be able to output a solution to the problem that is reasonably close to the optimal one in time polynomial in the input size. But for many applications such an approximate solution is not suitable. Also measuring the running times only in terms of the input size effectively ignores any structure of the instances.

Abrahamson, Ellis, Fellows, and Mata [AEFM89] were the first to propose to study the problems also with respect to some additional measure, the parameter, distinguishing whether there is an algorithm that can optimally solve all instances which have this parameter bounded by k in time $O(f(k) \cdot n^c)$, where f is some function, n is the input size and c is a constant independent of k or whether such an algorithm would require time $O(n^{f(k)})$ for some function f, that is, the exponent depends on k.

This idea was further elaborated by Downey and Fellows [DF92b, DF92a] in a series of papers in which they established the theory of fixed-parameter tractability and completeness. The series culminated in the ground-breaking textbook by Downey and Fellows [DF98], which attracted many people to the field and started a rapid development of the field.

Although since 2004 the International Symposium (formerly workshop) on Parameterized and Exact Computation (IPEC) devoted to results on parameterized complexity and exact moderately-exponential algorithms is organized, the papers involving parameterized complexity are accepted on a wide range of conferences devoted to graph problems, algorithms and complexity.

With the growing number of papers it was more and more clear, that the algorithms used to show fixed-parameter tractability often use techniques specific for the field. The 2006 Niedermeier's book [Nie06] summarizes such techniques,

while the monograph by Flum and Grohe [FG06] from the same year concentrates more to the intractability theory and maps various emerging complexity classes.

Since then the advancement in the field did not stop. Since the beginning it was noted by Downey and Fellows that parameterized complexity can be understood as a framework to measure the effectiveness of polynomial preprocessing. The method most related to this perspective, the kernelization, has now its own theory, and quite recently, in 2009, a tradition of workshops solely devoted to this method was started.

The big advantage of parameterized complexity is that a single problem can be studied from various points of view, using the virtually infinite set of possible parameters. The intractability shown for a problem with respect to a particular parameter does not mean that parameterized complexity was unsuccessful for the problem, but, instead, that more work should be done to reveal more suitable parameters for the problem which can possibly capture its hardness.

Also parameterized complexity enables us to study the complexity of a problem with respect to various parameters and their combination in a multivariate manner as suggested recently by Fellows [Fel09] and Niedermeier [Nie10]. Such a study then helps to understand where the complexity of the problem comes from and why it is so hard to solve.

This thesis presents such a multivariate analysis of several problems in the graph theory for which the parameter that is considered to be the standard one, or the first to try fails to capture the hardness of the problem.

The first part of the thesis is devoted to a brief state-of-the-art overview of parameterized complexity. As the field is already very wide, it is out of scope of this thesis to completely summarize it. Instead, we only include topics most related to the further chapters and those that we find crucial to give the reader an overview of the area.

After giving the basic definitions in Chapter 3 and summarizing various measures used as parameters of problems in Chapter 4, we concentrate on the most important method of the parameterized algorithmics, the kernelization, and the theory related to it in Chapter 5. Chapter 6 presents other important algorithmic methods used to show fixed-parameter tractability. Finally, Chapter 7 presents the methods and classes of parameterized intractability.

The second part of the thesis is devoted to the multivariate analysis of several concrete NP-hard problems in the graph theory. In Chapter 8 we study Steiner problems in directed graphs. As these were known to be intractable with respect to the solution-size, we examine the influence of another parameter, the number of terminals. Unfortunately, we are not able to mimic the undirected case, where this parameter leads to fixed-parameter tractability. Instead, we show that the problems are intractable even with respect to a combination of the mentioned parameters. This is complemented by algorithmic results for some special cases.

Chapter 9 is devoted to the study of domination-type problems. We first show that for a wide class of such problems the standard parameterization by a size

of the solution fails to confine the combinatorial explosion. Then the situation is examined with respect to the dual parameterization, for which the tractability is achieved. Additionally, we study closely related problems which pose parity-type constraints on the number of neighbors in the sought set, also with respect to the dual parameterization.

In the last Chapter 10 we aim on partitioning a graph into parts that are as close in size as possible, that is, their sizes differ by at most one. Such partitions are called equitable. In the two studied problems we make the partitions produced satisfy two natural conditions—either we require every partition to induce connected subgraph, or to induce an independent set. As the problems are known to be hard with respect to the most obvious parameter — the number of partition classes—we examine them with respect to various structural measures. Namely the problems turn out to be intractable with respect to the treewidth, the pathwidth and the feedback vertex set number, while tractable with respect to the vertex cover number and the max leaf number. The hardness result for the first problem holds even for planar graphs.

Chapter 2

Preliminaries

2.1 Sets, numbers, languages, formulas

Through the thesis we use standard notations. By $\mathbb{N}$ we denote the set of all positive integers (*natural numbers*), whereas $\mathbb{N}_0$ denotes all non-negative integers. We also denote EVEN $= \{0, 2, 4, 6, \ldots\}$ the set of all even non-negative integers and ODD $= \{1, 3, 5, \ldots\}$ the set of all odd positive integers. We call a set $A \subseteq \mathbb{N}_0$ *cofinite* if its *complement* $\overline{A} = \mathbb{N}_0 \setminus A$ is finite and *recursive* if there is a deterministic algorithm that given $n \in \mathbb{N}_0$ correctly decides whether $n \in A$ in a finite time.

Set of all subset of a set S is denoted $\mathcal{P}(S)$. If S is a set and $r \in \mathbb{N}_0$ then $\binom{S}{r} := \{A \subseteq S \mid |A| = r\}$ denotes the set of all r-element subsets of S. As usual, we let $|A|$ denote the cardinality of the set A. We say that $V_1, V_2, \ldots, V_r$ is *a partition of a set* V if and only if $\bigcup_{i=1}^{r} V_i = V$, and $\forall i, j, 1 \leq i < j \leq r : V_i \cap V_j = \emptyset$. A partition is *equitable* if $\forall i, j, 1 \leq i < j \leq r : ||V_i| - |V_j|| \leq 1$.

For a function h and a subset S of its domain, we denote the restriction of h to S by $h|_S$. We use the standard $O()$ notation to compare asymptotical growth of functions, although when talking about exponential functions sometimes the $O^*()$ notation is used, which suppresses the polynomial factors of the functions.

An *alphabet* is any finite set. Most often the alphabet $\{0, 1\}$ can be used for our purposes. Elements of the alphabet are called *symbols*. A *word* or a *string* in an alphabet is a finite sequence of symbols. We denote by Σ^* the set of all words in the alphabet Σ and by $|y|$ the length of the word y, i.e. the number of (occurrences of) symbols in it. A *language* is a set of words.

In a *decision problem* we are given some input and a guestion and the task is to decide whether the answer to this question is yes or no. When studying decision problems we encode them in some suitable alphabet. This way the set of all inputs to the decision problem with the positive answer corresponds to some language over this alphabet. Hence we use the terms decision problem and language as synonyms. We also assume that it is easy to recognize whether a string

constitutes an encoding of an input of the problem or not. More details on the decision problems and their encodings can be found in any standard computational complexity textbook, e.g. [AB09].

The *running time* or *time complexity* of an algorithm is the maximum number of steps it performs on inputs of given length. We do not examine the space complexity of algorithms in this thesis. When talking about algorithms we describe them and measure their complexity on the RAM (Random Access Machine) model with unit cost per arithmetic operation, with the restriction that the binary encoding of every number involved in the computation must be polynomial in the input size (so we cannot abuse this model). For complexity considerations the Turing Machine (TM) model is used more often. Although the time complexity with respect to this models generally differs, if the question is only whether there is an algorithm running in polynomial time (*polynomial algorithm*), the model chosen does not matter [CR72]. The class of all decision problems having a polynomial algorithm that correctly decides whether the input string is in the corresponding language or not is called P. Such problems are called *polynomially solvable*.

The class NP (stands for nondeterministic polynomial time) contains problems having certificates of solution that can be checked in polynomial time. More precisely for such languages there is a polynomial-time checker algorithm taking inputs formed by two strings, such that for every string in the language there is a *witness* string of size polynomial in the size of the original string that makes the checker accept this pair of strings. By contrast, for a string not in the language, there is no complementary string of polynomial size that would make the checker accept.

A problem is *NP-hard* if every problem in NP can be reduced to it in polynomial time and *NP-complete* if it is NP-hard and in NP. A language $L \subseteq \Sigma^*$ is in coNP if its complement $\Sigma^* \setminus L$ is in NP. Finally, if there is a function $a : \mathbb{N} \to \Sigma^*$ (*advice*), a polynomial $p : \mathbb{N} \to \mathbb{N}$, and a polynomial algorithm $A(x, y, z)$ such that for every $n \in \mathbb{N}$ the length of the string $a(n)$ is at most $p(n)$ and x is in the language L if and only if there is no $y \in \Sigma^*, |y| \leq p(|x|)$ that would make $A(x, y, a(|x|))$ accept, then the language L is in the class coNP/poly. In this case we also say that it is in coNP with a polynomial advice, or it has coNP circuits. Refer to [AB09] for a deeper account on computational complexity.

2.2 Graphs

Our notation in the graph theory is standard, for the terms that we forgot to mention here, we refer the reader to any basic graph theory textbook as Matoušek and Nešetřil [MN09] or West [Wes96].

In most of the thesis we speak about undirected graphs. A *graph* G is a pair (V, E), where $V = V(G)$ is the set of *vertices* and $E(G) = E \subseteq \binom{V}{2}$ is the set

of edges. When we talk about several graphs, we sometimes call the vertices of some of them *nodes* to distinguish them from the vertices of the others. We consider only simple finite loopless graphs thorough the thesis. We denote the set of all such graphs $\mathcal{G}$. If there is an edge between two vertices u and v, that is $\{u, v\} \in E$, we say that the vertices u and v are *adjacent* or *neighbors* and also that they are endpoints of the edge $\{u, v\}$. A *complement* of a graph G is a graph $\overline{G} = (V, \binom{V}{2} \setminus E)$.

A graph H is a *subgraph* of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$ and it is an *induced subgraph* if $E(H)$ equals $E(G) \cap \binom{V(H)}{2}$. Conversely, H is a *spanning* subgraph if $V(H) = V(G)$. If S is a subset of the vertex set $V(G)$ then we denote the subgraph $(S, E(G) \cap \binom{S}{2})$ induced by the set S by $G[S]$, while $G \setminus S$ denotes the graph $G[V(G) \setminus S]$ (especially if $v \in V(G)$ then we use $G \setminus v$ in the meaning of $G \setminus \{v\}$). Similarly, if e is an edge of G, then $G \setminus e$ denotes the graph $(V(G), E(G) \setminus \{e\})$.

A *k-clique* is a graph or a subgraph on k vertices having an edge between any two of them. Conversely an *k-independent set* is an induced subgraph on k vertices with no edges. A *path* of length $t \in \mathbb{N}_0$ is a (sub)graph formed by distinct vertices $v_0, v_1, \ldots, v_t$ and edges $\{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{t-1}, v_t\}$. The vertices v_0 and v_t are its endpoints. Finally, a *cycle* of length $t \geq 3$ is a graph or subgraph formed by distinct vertices $v_1, \ldots, v_t$ and edges $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{t-1}, v_t\}, \{v_t, v_1\}$. If weights on edges are given then the length of a path refers to the sum of weights of the involved edges.

We call a graph *connected* if there is a path between any two of its vertices. Maximal connected subgraphs of a graph are called *connected components*. If a graph does not contain a cycle as a subgraph then it is called a *forest*. A *tree* is a connected forest. A *distance* $dist_G(u, v)$ between two vertices u and v is the length of a shortest path between them. A *radius* of a graph is the minimum number r such that there is a vertex that is in distance at most r from any other vertex of this graph.

A (proper) *k-coloring* of a graph G is mapping $c : V(G) \to \{1, \ldots, k\}$ such that no two adjacent vertices receive the same color (number). The set of all vertices that receive a particular color in the coloring is called a *color class*. A graph is *k-colorable* if there is at least one proper k-coloring of it. Graphs that are 2-colorable graphs are also called *bipartite* and denoted (V_1, V_2, E) or $(V_1 \cup V_2, E)$, where V_1 and V_2 are the color classes. We also say that V_1 and V_2 are the partitions of a *bipartition* of the vertices.

The set of all vertices adjacent to a vertex v is called the *(open) neighborhood* of v and denoted $N(v)$ while $N[v] = N(v) \cup \{v\}$ denotes the *closed neighborhood* of v. The *degree* $\deg(v)$ of a vertex v is the size of its (open) neighborhood. A vertex is *isolated* if it has empty neighborhood. A graph is *d-regular* if every vertex has degree d. A 3-regular graphs are also called *cubic*. A *matching* is a 1-regular subgraph and it is *perfect* if it is spanning. A graph is *d-degenerate* if every its (non-empty) subgraph (including itself) has a vertex of degree at most

d.

An *edge contraction* is an operation which removes an edge from a graph while simultaneously merging together its two endpoints, removing the possibly arisen paralel edges. The graph obtained from a graph G by contacting an edge e is denoted by $G \cdot e$. Note that any connected graph can be contracted to a (graph having) single vertex by means of edge contractions. A graph H is a *minor* of a graph G, if it can be obtained from G by a sequence of vertex deletions, edge deletions and edge contractions. A graph H is an *induced minor* of G if it can be obtained by a sequence of vertex deletions and edge contractions only. A graph class $\mathcal{C}$ is *(induced) minor closed* if any (induced) minor of a graph in $\mathcal{C}$ is again in $\mathcal{C}$, respectively.

By *subdividing an edge* we mean replacing it by a path of length 2. Note that if we subdivide an edge and then contract any of the two edges of the newly introduced path, we obtain the original graph. A graph G is a subdivision of a graph H if it can be obtained from H by subdividing edges. A graph is *planar* if it can be embedded into a plane without edge crossings. It is well known that a class of planar graphs is proper minor closed and also closed under taking subdivisions. A class of graphs is *proper* if it is nonempty and does not contain all graphs.

In some parts of the thesis (especially in Chapter 8) we deal with directed graphs. A *directed graph (digraph)* is a pair $D = (V, A)$, where $V = V(D)$ is again the set of vertices (or nodes more often this time) and $A(D) = A \subseteq V \times V$ is the set of *arcs*. Although we generally allow loops in this case, they play no role in our consideration - they are of no use nor make any obstacle for the solution of our problems. We use (u, v) to denote the arc directed from the vertex u to the vertex v (*starting* in u and *ending* in v) and in this case we also say that u has an arc *to* v (it is an *in-neighbor* of v) and v has an arc *from* u (it is an *out-neighbor* of u).

The notion of (induced) subgraph works analogously as in the undirected case. A *directed path* from a vertex u to a vertex v is a subgraph of D formed by vertices $u = v_0, v_1, \ldots, v_t = v$ and arcs $(v_0, v_1), (v_1, v_2), \ldots, (v_{t-1}, v_t)$ for some $t \in \mathbb{N}_0$, while a *directed cycle* is a subgraph $(\{v_1, \ldots, v_t\}, \{(v_1, v_2), \ldots, (v_{t-1}, v_t), (v_t, v_1)\})$ for some $t \in \mathbb{N}$. We say that u is *connected to* v and v is *reachable* from u if there is a directed path from u to v. A graph $D = (V, A)$ is *strongly connected* if between each pair of vertices u and v there is a path from u to v and a path from v to u. It is *(weakly) connected* if its *underlying graph* $(V, \{\{u, v\} \mid (u, v) \in A \text{ or } (v, u) \in A\})$ is connected.

In a directed graph $D = (V, A)$, the *in-neighborhood* $N^-(u)$ (*out-neighborhood* $N^+(u)$) of a vertex u is the set of vertices which have arcs directed to u (from u), and the *in-degree* $\deg^-(u)$ (*out-degree* $\deg^+(u)$) of a vertex u denotes the size of the in-neighborhood (out-neighborhood) of the vertex u. The terms neighborhood and degree refer to the union of the in- and the out-neighborhood and the sum of in- and out-degree in this case, respectively. A vertex with $\deg^-(u) = 0$ is a

source, and a vertex with $\deg^+(v) = 0$ is a *sink*.

2.3 Graph Widths

In this section we introduce some of the graph width measures that are used later in the thesis. We start by the most renowned one, the treewidth introduced by Robertson and Seymour [RS84].

A tree decomposition of a graph $G = (V, E)$ is a pair (T, σ), where T is a tree and $\sigma : V(T) \to \mathcal{P}(V)$ is a mapping assigning to each node x of the tree a subset V_x of vertices of the graph G called *a bag*, that satisfies the following:

(i) Every vertex $u \in V$ is in some bag,

(ii) for every edge $\{u, v\} \in E$ there is an $x \in V(T)$ such that $\{u, v\}$ is a subset of the bag V_x, and

(iii) for every vertex $u \in V$ if there are two bags V_x and V_y containing u then for every z on the unique path from x to y in T, u is contained in V_z.

The width of the tree decomposition (T, σ) is the size of the largest bag minus one.
The treewidth of a graph $tw(G)$ is the minimum width of a decomposition over all tree decompositions of G.

Restricting the tree involved in the tree decomposition to be a path we obtain path decompositions and the *pathwidth* $pw(G)$ of a graph, another measure introduced by Robertson and Seymour.

Although the treewidth is probably the most widely used measure for sparse graphs, it is not suitable for dense graph, although they can also have simple structure. For that purpose, the clique-width was introduced by Courcelle and Olariu [CO00]. The definition we give is inspired by one given in [HOSG08].

Let k be a positive integer. We call (G, λ) a *k-labeled graph* if G is a graph and $\lambda : V(G) \to \{1, 2, \ldots, k\}$ is a mapping. The number $\lambda(v)$ is called *label* of a vertex v. We introduce the following operations on labeled graphs:

(1) For every i in $\{1, \ldots, k\}$, we let $\bullet_i$ denote the graph with only one vertex that is labeled by i (a constant operation).

(2) For every distinct i and j from $\{1, 2, \ldots, k\}$, we define a unary operator $\eta_{i,j}$ such that $\eta_{i,j}(G, \lambda) = (G', \lambda)$, where $V(G') = V(G)$, and $E(G') = E(G) \cup \{vw \mid v, w \in V, \lambda(v) = i, \lambda(w) = j\}$. In other words, the operator adds all edges between label-i vertices and label-j vertices.

(3) For every distinct i and j from $\{1, 2, \ldots, k\}$, we let $\rho_{i \to j}$ be the unary operator such that $\rho_{i \to j}(G, \lambda) = (G, \lambda')$, where $\lambda'(v) = j$ if $\lambda(v) = i$, and $\lambda'(v) = \lambda(v)$ otherwise. The operator only changes the labeling so that the vertices that originally had label i will now have label j.

(4) Finally, $\oplus$ is a binary operation that makes the disjoint union, while keeping the labels of the vertices unchanged. Note explicitly that the union is disjoint in the sense that $(G, \lambda) \oplus (G, \lambda) \neq (G, \lambda')$ for any λ' as the former has twice the number of vertices of the later.

A *k-expression* is a well-formed expression t written with these symbols. The k-graph produced by performing these operations in order therefore has vertex set the set of occurrences of the constant symbols in t; and this k-graph (and any k-graph isomorphic to it) is called the *value* $val(t)$ of t. If a k-expression t has value (G, λ), we say that t is a *k-expression of* G. The *clique-width* of a graph G, denoted by $cwd(G)$, is the minimum k such that there is a k-expression of G.

Finally the following notions capture classes of graphs that do not have bounded treewidth (for any number there is a graph in the class with treewidth larger than that number), while they are still sparse and exhibit some structure. The concept of classes of graphs of bounded expansion was introduced by Nešetřil and Ossona de Mendez in [NOdM05] and in the series of journal papers [NOdM08b, NOdM08c, NOdM08d]. The concept of nowhere dense graphs was introduced by the same authors in [NOdM08a] and [NOdM08e].

An *r-shallow minor* of a graph G is a graph that can be obtained from a subgraph of G by contracting a family of disjoint connected subgraphs, each having radius bounded by r. The subgraph must be taken so that the result is a simple graph. The *density* of a graph is the ratio between the number of its edges and the number of its vertices. The *grad (greatest reduced average density) of rank r* of a graph G equals to the largest density of an r-shallow minor of G. We denote the grad of rank r of G by $\nabla_r(G)$. A class $\mathcal{C}$ of graphs has *bounded expansion* if there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that for every $G \in \mathcal{C}$ and every $r \geq 0$ integer $\nabla_r(G) \leq f(r)$.

A class of graphs $\mathcal{C}$ is said to be *nowhere dense* if for every r there is a graph H such that H is not an r-shallow minor of any $G \in \mathcal{C}$.

Examples of classes of graphs with bounded expansion include proper minor closed classes of graphs, classes of graphs with bounded maximum degree, classes of graphs excluding a subdivision of a fixed graph, classes of graphs that can be embedded on a fixed surface with bounded number of crossings per each edge and many others [NOdM08b, NOdM08c, NOdM08d]. It can be shown, that every class of graphs with locally bounded tree-width or locally excluding a minor is nowhere-dense [NOdM08a, NOdM08e].

Part I

Parameterized Complexity

Chapter 3

Basic Definitions of Parameterized Complexity

In this chapter we present the basis of parameterized complexity with focus on tractability. As the fixed-parameter tractability became well known approach for solving NP-complete problems, there are several textbooks presenting comprehensively the field (except for some recent developments). The corner-stone was laid by Downey and Fellows in [DF98], more recently Niedermeier [Nie06] focused on the algorithmics and Flum and Grohe [FG06] on the complexity theory.

3.1 Parameterization and Parameterized Problem

Parameterized complexity is a framework to measure hardness of instances of computational problems in a multi-dimensional manner. To this end, we first need a notion of another measure apart from the size of the instance.

Definition 3.1. *A parameterization* of a decision problem $\mathcal{P} \in \Sigma^*$ is a computable function $\kappa : \Sigma^* \to \Sigma^*$.

Example 3.2. Suppose that we are going to investigate the problem of Vertex Cover. Here one is given a graph G and $k \in \mathbb{N}$ and the question is whether there is a set of vertices of size at most k, such that each edge of the graph has at least one endpoint in this set (such a set is called a *vertex cover*). Hence we assume, that an instance of this (decision) problem is an encoding of a pair (G, k) in some alphabet Σ, G being a graph and $k \in \mathbb{N}$. Then the so-called solution-size parameterization gives (the encoding of) k if the input represents a pair (G, k) and (the encoding of) 1 otherwise. If a parameterization is, as in this example, a projection of the input to one of its components, we will usually use the name of this component to denote the parameterization.

Parameterized complexity deals with parameterized problems:

Definition 3.3. By a *parameterized problem* $\mathcal{L}$ we mean any subset $\mathcal{L} \subseteq \Sigma^* \times \Sigma^*$, where Σ is some finite alphabet. An *instance* of the parameterized problem $\mathcal{L}$ is any pair $(x, k) \in \Sigma^* \times \Sigma^*$, where x is called the *main part* and k is the *parameter* (of the instance). We call (x, k) a *yes-instance* for the parameterized problem $\mathcal{L}$ if $(x, k) \in \mathcal{L}$ and a *no-instance* otherwise.

A parameterization provides a connection between classical decision problems and parameterized problems:

Definition 3.4. If $\mathcal{P} \subseteq \Sigma^*$ is a decision problem, and $\kappa : \Sigma^* \to \Sigma^*$ a parameterization, then we denote by κ-$\mathcal{P}$ the parameterized problem κ-$\mathcal{P} := \{(x, \kappa(x)) \mid x \in \mathcal{P}\}$. If the parameterization is clear from the context (especially for the so-called standard parameterization) we often omit its specification and use $\mathcal{P}$ also for the parameterized problem.[1]

Remark 3.5. Some authors [FG06] define the parameterized problem as a pair $(\mathcal{P}, \kappa)$ and require the parameterization to be a polynomial-time computable function $\kappa : \Sigma^* \to \mathbb{N}$. We prefer to use definition as stated above, as we neglect the possible time needed to evaluate the parameterization. There are also some parameterizations for which it is not possible to compute them in polynomial time, justifying our decision (see Section 4.4).

We defer the discussion on whether the parameter must be an integer after the main definition of parameterized algorithmics.

3.2 Fixed-Parameter Tractability

The main goal of parameterized complexity was always to decide which parameterized problems are fixed-parameter tractable and which are not.

Definition 3.6. We say that a parameterized problem $\mathcal{L} \subseteq \Sigma^* \times \Sigma^*$ is *fixed parameter tractable* if there is an algorithm that correctly decides whether an instance (x, k) is in $\mathcal{L}$ in time $f(k) \cdot |x|^c$ for some function $f : \Sigma^* \to \mathbb{N}$ and some constant c. Such an algorithm is called a *parameterized algorithm* or *fpt-algorithm*. The class of all fixed parameter tractable problems is denoted FPT.

Remark 3.7. Distinguishing the cases $f(k) \leq |x|$ and $f(k) > |x|$ one can see that an $(f(k) \cdot |x|^c)$-algorithm also runs in time $(f(k))^2 + |x|^{2c}$. On the other hand, an $(f(k) + |x|^c)$-algorithm also runs in time $2f(k) \cdot |x|^c$, as we assume both $|x|$ and $f(k)$ to be at least one. Hence FPT is also a class of par. problems having $(f(k) + |x|^c)$-algorithms.

[1] As there is no standardized way to denote the parameterization of the problem, some traditional names of decision problems already use the dash. In these exceptional cases we keep the traditional names and, thus, the part before the dash is then not necessarily a parameterization.

Definition 3.8. A parameterized problem $\mathcal{L} \subseteq \Sigma^* \times \Sigma^*$ is in the class XP if there is an algorithm that correctly decides whether an instance (x, k) is in $\mathcal{L}$ in time $f(k) \cdot |x|^{g(k)}$ for some functions f and $g : \Sigma^* \to \mathbb{N}$.

Definition 3.9. We say that a (decision) problem $\mathcal{P}$ is (in)[2] FPT (in XP) with respect to the parameterization κ (or parameterized by κ) if the parameterized problem κ-$\mathcal{P}$ is in FPT (in XP), respectively. Again, if clear from context we sometimes omit the parameterization.

Example 3.10. Vertex Cover is FPT with respect to k. We will prove that in Section 5.1 (and also 6.1).

The fundamental difference between the FPT and XP algorithmic running-times laid the corner-stone of parameterized complexity. It is obvious that FPT $\subseteq$ XP, while it is known that XP $\not\subseteq$ FPT [DF98].

Remark 3.11. We have allowed the parameter to be an arbitrary string. But in some cases (e.g. when talking about polynomial functions of the parameter) it is necessary to restrict the parameter to be formed by a (single) integer. Therefore some authors require the parameter to be an integer [FG06]. On the other, hand it is sometimes natural to consider for example a graph or a pair of numbers as a parameter. It is easy to check that if there is an fpt-algorithm deciding whether $(x, k) \in \Sigma^* \times \Sigma^*$ is a yes-instance of the problem running in time $f(k) \cdot |x|^c$ for some function $f : \Sigma^* \to \mathbb{N}$ and some constant c, then the algorithm also runs in time $f'(|k|) \cdot |x|^c$ for some $f' : \mathbb{N} \to \mathbb{N}$. Hence, we can replace the parameter $k \in \Sigma^*$ by the parameter $|k| \in \mathbb{N}$ if necessary, without affecting the membership of a particular problem in FPT (or XP). However, if the parameter particularly is formed by an r-tuple $(a_1, a_2, \ldots, a_r) \in \mathbb{N}^r$ of natural numbers, we prefer to replace it by the sum $a_1 + a_2 + \cdots + a_r \in \mathbb{N}$, as this preserves the polynomiality of functions of the parameter.

[2]The letters FPT are often used as an acronym for fixed-parameter tractable. Therefore we often say that a problem is FPT, instead of saying that it is *in* FPT

Chapter 4

Parameterizations

In the previous chapter we have said that parameterized complexity relies strongly on having further measure on the instances of the problem apart from the input size. In this chapter we discuss some of the measures used. We would like to remark that when talking about complexity of graph problems mostly the number of vertices n and sometimes the number of edges m are used to measure the size of the input. These quantities, although closely related to, are also quite different than the actual bit-size of the input. But they are quite unsuitable for our purpose by the following reason.

It is easy to see, that if a parameterization grows monotonically with the input size and is unbounded (or it is lower bounded by such a function) then any decidable problem is FPT with respect to this parameterization. Hence we head for parameterizations unrelated to the input size. Also, as the dependence of running time of fpt-algorithms on the parameter is mostly exponential or worse, there is a strong need for parameterizations that remain small on some reasonable part of the instances that are (though to be) practical.

4.1 Solution-Size and its Dual

As the vast majority of decision problems originate from optimization problems, their instances are usually equipped with a number k representing the size of the sought solution. Or, in view of the definition of the class NP, the size of the witness we search for. This number is widely used; such a parameterization is called the *standard parameterization* or the *parameterization by the size of the solution*.

The disadvantage of this parameterization is that we sometimes cannot expect it to be small. On the other hand, there is usually an input-size related upper bound for the parameter, the answer being trivial above it. Then we can use the parameter saying how far from this upper bound we can get. This is called *the dual parameterization* and the corresponding parameterized problem is called *the*

parametric dual of the original parameterized problem. For example, if we have a graph problem where we search for a vertex subset S of size k satisfying some condition, and the size of the vertex set V is n, then the dual parameterization assigns to such an instance the number $n - k$, that is the size of $V \setminus S$. Or, equivalently, we can keep the parameter to be k and change the question to be "Is there a set of size $n - k$?"

Special case of the solution-size parameterization is when the input is weighted, i.e., we are given weights on the elements that we can use in the solution an the question is, whether there is a solution with total weight not exceeding the given bound (budget). As a rule, in this case, it is necessary to normalize the weights in some sense, as otherwise (in most cases) any instance can be transformed to an instance with the budget 1 (dividing all the weights by the budget), effectively ruling out the possibility of the problem being FPT (unless it is in P) with respect to the parameterization by the budget. Hence we either allow only integer weights, or take the ratio between the budget and the minimum weight used as a parameter (or force the minimum to be 1).

4.2 Parameterization Above Tight Lower Bound

Sometimes there is an unbounded growing function f of the size of the input $|x|$, such that whenever the parameter k is less than this function $f(|x|)$, then the answer is trivial. This means always yes or always no, depending only on the problem considered. As an example consider Max d-Sat. Here one is given a propositional formula in form of conjunction of m clauses, each formed by exactly d literals and the question is whether it is possible to satisfy at least k clauses simultaneously. By a simple probabilistic argument one can show, that it is always possible to satisfy at least $1-2^{-d}$ fraction of the clauses. Hence whenever we are asked whether it is possible to satisfy $k \leq m(1-2^{-d})$ clauses we can simply answer yes. Otherwise there is less than $k/(1-2^{-d})$ clauses containing less than $d \cdot k/(1-2^{-d})$ variables and any brute force algorithm can be used to show the fixed-parameter tractability of the problem with respect to the parameter k.

Of course this algorithm is somewhat unsatisfactory. The question arises, wheather we can get a bit more, than what we are quaranteed. That is to pose a question: "Is it possible to satisfy at least $m(1-2^{-d})+k$ clauses?" or, equivalently, parameterize the problem by $\max\{k - m(1-2^{-d}), 1\}$. If the lower bound used is tight, then such a parameterization is called the *parameterization above tight lower bound*. To see that the bound from our example is tight it is enough to consider a formula formed by a blocks of all 2^d possible clauses on some d variables, taking different variables for different blocks. As often the lower bound is obtained by some probabilistic argument, this kind of parameterization

is also often called the *parameterization above average*.

There are several positive results for such parameterizations, to see the one for the above example refer to [AGK+10].

4.3 Further Natural Parameters

Definition of some other problems already shows off some natural parameters. Graph problems often include several "request" that should be satisfied simultaneously. As the number of the request can be often assumed to be small it forms a natural measure to parameterize with.

A typical example of such problem is Steiner Tree, where we are given an edge weighted graph $G = (V, E)$, a subset of vertices T (called *terminals*) and a natural number $p \in \mathbb{N}$. The question is whether there is a connected subgraph of G of weight at most p containing all vertices of T. While p can be regarded as the size of the solution, $|T|$ provides a further natural parameter—we will see further in the thesis that even much more useful.

Another example can be found in the more recently studied k-Local Search for Traveling Salesperson [Mar08]. In this problem we are given a graph with positive weights on edges and a Hamiltonian cycle in it and the question is whether we can find a Hamiltonian cycle which uses at most k edges not used by the original cycle and its weight is smaller than the weight of the original one. Here the number k can be hardly called size of the solution. Similar parameter is involved in Conservative Coloring where we are given a graph which has all vertices except for one properly colored by k colors and we search for a proper coloring of that graph with k colors which differ from the original one on at most c places [HN10]. Again, c is a natural parameter which does not represent the solution-size.

Further natural parameters appear in the problems where we search for a consensus. For example in one problem in voting systems we are given several linear orders on the candidates representing the votes and we search for a linear order that is reasonably close to the given orders in a given distance measure. Such a problem offer several natural parameters as the number of voters, number of candidates, the sum of the distances to the consensus or the maximum distance and many further. Moreover for each of them there is a natural scenario, where it is reasonable to assume that this parameter will be small [BGN08].

Natural parameters appear also in many other areas, for example for geometrical problems in higher dimensions it is natural to consider the dimension of the problem as a parameter [Kna10]. For string problems the size of the alphabet is often considered, etc.

4.4 Structural Parameters - Graph Widths

In praxis, many graph problems come with instances formed by sparse graphs, often having some additional structure. The most obvious sparsity measure is the average degree or the closely related degeneracy (see Section 2.3 for the definitions). Although these parameters are quite useful for many problems, for many others they provide too little structure.

The situation improves when we restrict our attention to the most studied class of sparse graphs — planar graphs. Or more generally we parameterize our problems be the genus of the graph, which is FPT to determine [Moh99]. Although many hard problems become tractable on graphs of bounded genus, still there are many more, that are NP-complete even on planar graphs Hence many other measures — graph widths — were introduced to really catch the structure of graphs. . Nowadays there are dozens if not hundreds of such widths.[1]

If we should present some of the widths, we have to start with *the treewidth*. Introduced by Robertson and Seymour [RS84] in 1984 it is the most widely recognized and definitely the most successful of them.

Interestingly it is NP-hard given G and k to decide whether the treewidth of G is at most k [ACP87], while it is FPT parameterized by k as there is an algorithm, whose running time is linear for every fixed k [Bod96]. Since the algorithm actually finds the decomposition in case one exists, this implies the existence of an optimal decomposition with linear number of nodes. While this algorithm is completely impractical due to the huge function $f(k)$ involved in the running time, there are practical algorithms constructing a decomposition of width $3k+2$ if the treewidth of G is at most k [Ree92] and, if this is not sufficient, heuristic approaches are used [BK10].

Hence, if we parameterize the problem by the treewidth, it is convenient (but should be mentioned) to assume that we are given a decomposition, and the complexity of the algorithm is actually measured with respect to the width of the decomposition given. We just have to bear in mind that if we don't provide the algorithm with the optimal decomposition, the bound on the running time of the algorithm being an expression of the treewidth should be actually considered as an expression of the width of the decomposition given.

After the great success of the treewidth, the research aimed both to extend the tractability results to wider classes of graphs as well as to further restrict the class of graphs considered to achieve tractability for further problems. To compare the measures we use the following notion. For two graph measures $\kappa, \kappa' : \mathcal{G} \to \mathbb{N}$ we say that κ is *more restrictive* than κ' (κ' is *less restrictive* than κ) if there is a function $g : \mathbb{N} \to \mathbb{N}$ such that for every graph $G \in \mathcal{G} : \kappa'(G) \leq g(\kappa(G))$. If κ is both less and more restrictive than κ' then we say that they are *equivalently*

[1] Over 40 were mentioned on the 2009 workshop on Graph Classes, Optimization, and Width Parameters (GROW).

restrictive.

The intuitive meaning is that if we have a subclass of graphs such that the values of some width are bounded by a constant on that class, then the values of any less restrictive width are bounded by a constant too. For example the pathwidth is more restrictive than the treewidth. Also if a problem is FPT parameterized by κ (without giving the decomposition) and κ' is more restrictive than κ then the problem is automatically also FPT with respect to κ'.

One important branch of the research was that although the cliques are easy instances for most problems, their treewidth grows linearly with their order, treating them as being the hardest instances. Hence there was a call for a graph parameter less restrictive than treewidth. This issue is addressed by *the clique-width, the rank-width, the module-width*, or *the boolean-width* all of them being equivalently restrictive. The definition of the clique-width is given Section 2.3. We omit the definitions of the other as we do not need them for any of our results. See [HOSG08] for a survey on the properties of the clique-width and the rank-width and [Rao08] and [BXTV09] for some properties of the module-width and the boolean-width, respectively.

More related to our results are some widths that are more restrictive than the treewidth. First, restricting the tree involved in the tree decomposition to be a path we obtain *the pathwidth* $pw(G)$ of a graph. The restricted structure of the decompositions can help to devise algorithms, but there are very few examples of problems being intractable parameterized by the treewidth and tractable parameterized by the pathwidth.

Another approach is represented by *the feedback vertex set number* $fvs(G)$, which is the size of the minimum feedback vertex set in the graph G. A subset of vertices S is a *feedback vertex set* for the graph G if the graph $G \setminus S$ is a forest. It may seem strange at first sight to use the result of one optimization problem as a parameter for another problem, but such parameterizations turn to be useful. A natural restriction in this case for the decision version of the "parameterizing" optimization problems is to be FPT with respect to the solution size. We also need it to provide some structure that we can further use. In the case of feedback vertex set number, we can start by doing the brute force on the (small) feedback vertex set and afterwards the rest is just a forest that we can hopefully treat easily. The feedback vertex set number is more restrictive than the treewidth as it can be seen that always $tw(G) \leq fvs(G)$. To see that, consider the width-1 tree decomposition of the forest $G \setminus S$ and add S to every bag.

As further examples of this kind let us mention *the vertex cover number* $vc(G)$, which is the size of the minimum vertex cover of the graph and *the max leaf number* $ml(G)$, the maximum number of leaves in a spanning tree of a graph. Both of them are more restrictive than both the feedback vertex set number and the pathwidth. To see the later, the result of Kleitman and West [KW91] can be used showing that if the maximum number of leaves in any spanning tree of a graph G is k then G is a subdivision of a graph on at most $4k - 2$ vertices.

Hence, maybe, the minimum number of vertices and the minimum number of edges of a graph our graph is a subdivision of would be a better candidate for a parameterization. But $ml(G)$ is used more often, maybe because there is no short name for the other two.

Finally when talking about classes of graphs with locally bounded treewidth, locally excluding a minor, of bounded expansion or nowhere dense, the sparsity is actually measured by some function $f : \mathbb{N} \to \mathbb{N}$ or an infinite sequence of excluded minors. Hence it can be hardly used as a parameter, even though some authors talk about parameterized algorithms also in this case. Note in this direction, that when talking about H minor free graphs, H or $|H|$ is sometimes indeed used as a parameter [AG08].

4.5 Multivariate Approach

Parameterized complexity tends not only to treat the problems under different parameterizations, but also in different scenarios, where the interplay between more parameterizations influences the complexity of the problems. We are always interested in the existence of an algorithm with a certain running time. In such a scenario a parameterization κ can play several different roles (the roles are ordered from those putting the most restrictions on the parameterization to those putting no restrictions):

- It is a constant with a *known value* k_0 - we restrict our attention to instances having the value of κ equal to k_0. Our algorithm quite possibly does not work at all for instances having κ different then k_0 and thus no dependence of the running time on κ is to be examined. As an example consider the parameterization by the graph genus. Devising an algorithm just for planar graphs equals restricting the genus to be 0.

- It is a *constant* but the value is unknown - the algorithm works for all values of κ but the running time can depend on κ quite arbitrarily, in particular κ can appear in an exponent of the polynomial in the running time. A typical example is the number d in d-Hitting Set[2] when we examine families of sets, each having at most d elements. It is supposed that this measure will be very small when applying the scenario, say definitely less than 10.

- It is a *parameter* - then κ can only influence the multiplicative factor in the polynomial running time. With such an influence the algorithm can grow much higher, for some parameterized problems, values of parameter larger than 100 can still give reasonable running times.

[2] d-Hitting Set: Given a family of sets, each with at most d elements and $k \in \mathbb{N}$, decide, whether there are at most k elements, such that each set in the family contains at least one of them (is *hit*).

- It is just a *variable* without any influence on the running time of the algorithm. This is included only for completeness, as this rather means that κ plays no role in this scenario.

To get better the idea, consider a scenario in which κ_1 has known value k_0, κ_2 is a constant, κ_3 forms the parameter and κ_4 is a variable. Then we are interested in the existence of an algorithm which works for all instances having κ_1 equal to k_0 and its running time on an instance x is bounded by $f(\kappa_2(x), \kappa_3(x)) \cdot |x|^{g(\kappa_2(x))}$ for some functions f and g.

It is important to note, that an FPT result for some scenario also applies to scenarios where each parameterization plays the same or more restricted role. On the other hand, a hardness result for a particular scenario also translates to scenarios that are less restrictive. In particular, an fpt-algorithm with respect to a single parameter also shows that the problem is in FPT with respect to the combination of this parameter and any other. Conversely, a hardness result for a combined parameterization means also hardness with respect to each single parameter involved in the combination. This way the number of scenarios we need to study to get the full picture can be significantly decreased.

Also note that it makes no sense to use several structural parameterizations with the same role if one of them is more restrictive than the other (see Section 4.4). Although there are several hardness results for a combination of two incomparable structural parameters such as the pathwidth and the feedback vertex set number, it seems complicated to use structures provided by two parameters in combined matter to develop an algorithm.

The main purpose of studying different scenarios is not only to provide people solving the problem the best suited tool for their particular case, but also to understand where the hardness of the problem comes from, which is very important from the theoretical point of view. Hopefully understanding of the problem hardness can lead to even better tractability results for it. More ideas about how to examine the problem in the fully multivariate manner can be found in [Fel09] and [Nie10].

Chapter 5

Kernelization Point-of-View

Kernelization is a natural formalization of a notion of effective polynomial preprocessing in terms of parameterized complexity. It is known by many names such as data reduction or reduction to a problem kernel. Of course efficient preprocessing is not only a domain of the parameterized algorithms. As the use of a kernelization makes no harm, it can be used prior to almost any approach for solving the problem, such as heuristics or approximation algorithms.

5.1 Basic Ideas

We present the basic ideas on an example of the now legendary kernelization for Vertex Cover, which is attributed to Samuel R. Buss in [DF98], but nowadays is considered rather folklore. Recall than in Vertex Cover we are given a graph G and a natural k and the question is whether there is a set of at most k vertices in which every edge has at least one endpoint. The standard parameter (considered here) is k.

First observation that can help to reduce the instance is that in an instance of Vertex Cover isolated vertices play no role. Hence we can replace G by $G \setminus I$ in our considerations, where I is the set of isolated vertices in G. This is usually called *a reduction rule*. To see its *correctness (soundness)* it is necessary to check that the instance produced by the rule is a yes-instance if and only if the original one is. In our case this means that G has a vertex cover of size at most k if and only if $G \setminus I$ does so, which is obvious.

Further consider a vertex v having more than k neighbors in G. It has to take part in any vertex cover of size at most k as the cover cannot contain all neighbors of the vertex. Thus G has a vertex cover of size k if and only if $G - v$ has a vertex cover of size at most $k-1$, immediately leading to another reduction rule. Note that, in contrast with the previous rule, this rule uses the value of the parameter - it is *parameter dependent*. Of course parameter independent rules are more desirable, as they can be used also on the optimization problem itself,

not only on its (parameterized) decision variant.

To complete the kernelization we need a *boundary lemma* saying that if the instance is *reduced* (none of the reduction rules can be applied to it any more) then either the answer is trivial, or the size of the instance is bounded in terms of the parameter. If the VERTEX COVER instance is reduced with respect to the above two rules, then either it is a no-instance or it has at most $k^2 + k$ vertices, as each vertex is non-isolated and thus must be in the cover or a neighbor of one of (at most) k cover vertices, each of them having degree at most k. Obviously, reduced yes-instance has also at most k^2 edges and thus can be described by $O(k^2 \cdot \log k)$ bits.

Now we are ready to give a formal definition of kernelization:

Definition 5.1. *A kernelization* of a parameterized problem[1] $\mathcal{P} \subseteq \Sigma^* \times \mathbb{N}$ is a polynomial time evaluable function A that on the input $(x, k) \in \Sigma^* \times \mathbb{N}$ produces an instance $(x', k') := A((x, k)) \in \Sigma^* \times \mathbb{N}$ such that

- (x', k') is in $\mathcal{P}$ if and only if (x, k) is in $\mathcal{P}$, and
- there is a function $g : \mathbb{N} \to \mathbb{N}$ such that $|x'| \leq g(k)$ and $k' \leq g(k)$.

The instance (x', k') is called *problem kernel* and the function g is called *the size of the kernel*. If we talk about $g(k)$-kernel for some function g we always mean, that there is a kernelization with g being the size of the kernel.

5.2 Further examples

First we note that that our first example translates also to a generalization of VERTEX COVER called d-HITTING SET: Given a family of sets, each with at most d elements and $k \in \mathbb{N}$, decide whether there are at most k elements, such that each set in the family contains at least one of them (is *hit*). Here, in conflict with our notion (but in accordance with the literature), we consider k to be the parameter and d to be a fixed constant, the case of $d = 2$ is exactly VERTEX COVER.

Of course it doesn't make sense to consider elements not contained in any set of the family. Similarly a superset of another set in the family is redundant in the instance. The mentioned "high degree" rule can be generalized to this case by a notion of a sunflower:

Definition 5.2. Sets $S_1, \ldots, S_r$ form *a sunflower* if there is a (possibly empty) set $A := \bigcap_{i=1}^{r} S_i$ (*center*) such that the intersection of any two sets S_i and $S_j, j \neq i$ is equal to A or, equivalently, if the sets $S_i \setminus A$ (*petals*) are disjoint.

[1] We restrict ourselves here to parameters formed by a single integer, as we want to later speak about polynomial functions of the parameter; see also Remark 3.5.

If an instance of d-Hitting Set contains a sunflower with more than k petals then in order to hit each set in the sunflower we have to select an element from the center. This can be represented by adding the center into the instance and removing all supersets of the center (in particular all sets of the sunflower).

Using the so-called Sunflower lemma or the Erdös-Rado Lemma (see [FG06, Lemma 9.7, p. 211]) one can show that an instance which has more than $k^d \cdot d \cdot d!$ sets contains a sunflower with more than k petals that can be found in polynomial time in both k and the size of the instance. Thus if a reduced family contains more sets, it is a no-instance. With the $k^d \cdot d \cdot d!$ sets containing altogether at most $k^d \cdot d^2 \cdot d!$ elements we arrive at an $O(k^d \cdot d^2 \cdot d! \cdot (2 \log d + d \cdot (\log k + \log d)))$-kernel. We mention in one of the next sections that this is asymptotically optimal.

The above example is included since most kernelizations in fact use some kind of high degree rule. It is a *local rule* in the sense that it only examines an (r-)neighborhood of a vertex (or an element) an tries to replace it with something simpler with the same function. On the other hand there are also *global rules* examining the structure of the whole graph at once. We present an example of a crown rule in one less ordinary, although well known application on the parametric dual of Graph Coloring. Here the question is given a graph G on n vertices and $k \in \mathbb{N}$ can G be properly colored by at most $n - k$ colors? It is known as "How to Save k Colors in $O(n^2)$ Steps" [CFJ04].

Definition 5.3. *A crown decomposition* of a graph $G = (V, E)$ is a partition $C \cup H \cup B = V$ of the vertex set, such that

- C is an independent set,
- there are no edges between C and B, and
- there is a matching of size $|H|$ between H and C.

The sets C, H, and B can be thought of as *the crown, the head*, and *the body*, respectively.

Lemma 5.4. *There is an algorithm, that in polynomial time finds either*

- *a matching of size at least $k + 1$, or*
- *a crown decomposition $C \cup H \cup B$ such that $|B| \leq 3k$.*

The proof can be found in [CFJ04] and we omit it here.

To solve the dual of coloring, we run the algorithm on the the complement $\overline{G}$ of the graph G. If we find a large matching, then obviously we can color the matched vertices by a same color, obtaining a coloring with less than $n - k$ colors. If $C \cup H \cup B$ is a crown decomposition of $\overline{G}$ then C is a clique in G that is connected to every vertex of B. Hence each vertex of C needs its own color that, furthermore, cannot be used on B. On the other hand, due to the matching

between H and C (in $\overline{G}$) it is possible to use colors of C for the vertices of H. Thus G can be colored with $n-k$ colors if and only if $G[B]$ can be colored by $|B|-(k-|C|)$ colors. Since B has at most $3k$ vertices, this directly yields a kernel.

We remark, that via the crown decomposition we can similarly get a kernel with $3k$ vertices (of bitsize $O(k^2)$) also for Vertex Cover. A survey of other results on kernelization can be found in [GN07].

5.3 FPT means Kernelization

The following theorem gives a notification of the importance of kernelization for parameterized complexity.

Theorem 5.5. *A decidable parameterized problem is FPT if and only if it has a kernelization.*

Proof. First suppose that $\mathcal{P}$ is FPT, that is, there is an algorithm running in time $f(k)\cdot n^c$. The promised kernelization works as follows: If $f(k)\leq n$ then it solves the instance in $f(k)\cdot n^c \leq n^{c+1}$-time and outputs some (constant size) trivial instance being in the language if and only if the input one is. Otherwise $n<f(k)$ and it outputs the same instance without a modification.

For the other direction it is enough to use any algorithm (ensured by the decidability) on the output of the kernelization. □

Having the above theorem in hand it may seem that having a kernelization is just another name for being FPT. The important thing here is the size of the kernel. From the above theorem we only get super-polynomial kernels (for NP-hard problems). In fact, as noted by Bodlaender [Bod09], we cannot get a constant size kernel for NP-complete problem (unless P=NP). What we can get are kernels of polynomial size as we have seen in our example and such kernels are of broad interest, as not only they often lead to the best known fpt-algorithms for a particular problem, but as we have mentioned earlier, they can be used in virtually any approach to solve the problem.

5.4 On the Non-Existence of Polynomial Kernels

As we have already said, a (polynomial) kernelization for a problem is more valued than just fixed-parameter tractability. The theory of parameterized hardness is capable of showing fixed-parameter intractability (see Chapter 7) but it is not fine-grained enough to provide any evidence that for a fixed-parameter tractable problem there is presumably no polynomial kernel, or to even show that some

kernel is asymptotically optimal in terms of its size. In this section we present the framework that is capable of such results (under certain complexity theoretic assumptions). We need the following definition of an algorithm doing an "OR" of several instances:

Definition 5.6. *A composition algorithm* for a parameterized problem $\mathcal{P} \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that receives as input a sequence $((x_1, k), \ldots, (x_t, k))$, with $(x_i, k) \in \Sigma^* \times \mathbb{N}$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^{t} |x_i| + k$, and outputs $(y, k') \in \Sigma^* \times \mathbb{N}$ with

- $(y, k') \in \mathcal{P}$ if and only if there is an $1 \leq i \leq t$ such that $(x_i, k) \in \mathcal{P}$ and
- k' is polynomial in k.

A parameterized problem is *compositional* if there is a composition algorithm for it.

We also need the following notion:

Definition 5.7. *An unparameterized version* of the parameterized problem $\mathcal{P} \subseteq \Sigma^* \times \mathbb{N}$ is the language $\tilde{\mathcal{P}} := \{x \Diamond 1^k \mid (x, k) \in \mathcal{P}\}$, where $\Diamond \notin \Sigma$, and 1^k is a unary encoding of k with 1 being an arbitrary symbol of Σ.
It is important that in the unparameterized version of the problem the parameter forms a lower bound for the input size, as it is encoded in unary. For most graph problems this is the case anyway, thus taking the unparameterized version makes no difference.

The main theorem of the framework states:

Theorem 5.8 (Bodlaender, Downey, Fellows, and Hermelin [BDFH09]; Fortnow and Santhanam [FS08])**.** *Let $\mathcal{P}$ be a compositional parameterized problem whose unparameterized version $\tilde{\mathcal{P}}$ is NP-complete*[2]*. Then, if $\mathcal{P}$ has a polynomial kernel then NP$\subseteq$coNP/poly. this would imply a collapse of the polynomial hierarchy to the third level.*

To use the framework, it is necessary to show the composionality of the problem considered. For many graphs problems this is easy, the composition algorithm can simply return the disjoint union of the input graphs and leave $k' := k$. This works for example for k-Path, where one asks, whether a given graph G contains a path of a given length $k \in \mathbb{N}$, parameterized by k. But there are also problems for which the compositional algorithm is fairly complicated (cf. [DLS09]), sometimes surprisingly also using the positive results known for the problem.

It is also possible to transfer the result obtained on one problem to another problem for which we were unable to design compositional algorithm directly. For that purpose the following transformation is used:

[2]As we mostly deal with NP-hard problems, this requirement is fulfilled as long as the parameter can be upper-bounded by some polynomial of the input size.

Definition 5.9 (Bodlaender, Thomassé, and Yeo [BTY08])**.** Let $\mathcal{P} \subseteq \Sigma^* \times \mathbb{N}$ and $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems. We say that $\mathcal{P}$ is *polynomial parameter reducible* to $\mathcal{Q}$, written $\mathcal{P} \leq_{Ptp} \mathcal{Q}$, if there exists a polynomial time computable function $f : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ and a polynomial p, such that for all $(x, k) \in \Sigma^* \times \mathbb{N}$

- $(x, k) \in \mathcal{P}$ if and only $(x', k') := f(x, k) \in \mathcal{Q}$ and
- $k' \leq p(k)$.

The function f is called *polynomial parameter transformation.*

Proposition 5.10 (Bodlaender, Thomassé, and Yeo [BTY08])**.** *Let $\mathcal{P}$ and $\mathcal{Q}$ be the parameterized problems and $\tilde{\mathcal{P}}$ and $\tilde{\mathcal{Q}}$ be the unparameterized versions of $\mathcal{P}$ and $\mathcal{Q}$ respectively. Suppose that $\tilde{\mathcal{P}}$ is NP-complete and $\tilde{\mathcal{Q}}$ is in NP. If there is a polynomial parameter transformation from $\mathcal{P}$ to $\mathcal{Q}$, then if $\mathcal{Q}$ has a polynomial kernel then $\mathcal{P}$ also has a polynomial kernel.*

We believe that the usage of this proposition does not need any example, but we mention that such a reduction exists showing that k-Leaf Out-Branching has no polynomial kernel [FFL$^+$09]. In k-Leaf Out-Branching we are given a directed graph and $k \in \mathbb{N}$ and the question is whether the directed graph contains a subgraph in which every vertex except for one has in-degree exactly 1 and k vertices has out-degree 0. Another complicated transformation can be found in Dom et al. [DLS09], where it is shown that for certain problems, the suitably colored version of the problem has a kernel if and only if the uncolored version does.

Recently Dell and van Melkebeek [DvM10] proved by a method to some extent similar to the above framework, that the $O(k^d \log k)$-kernel for d-Hitting Set mentioned in Section 5.2 is presumably optimal up to a logarithmic factors. Namely they have shown, that there is no kernel of size $O(k^{d-\epsilon})$ for any $\epsilon > 0$, unless NP$\subseteq$coNP/poly. Furthermore they have shown that there are no $O(k^{2-\epsilon})$-kernels for a class of graph deletion problems, where the task is to delete at most k vertices from a given graph to obtain a graph that fulfill some fixed subgraph-hereditary graph property. Feedback Vertex Set, Bounded-Degree Deletion or Planar Deletion are examples of such a problems (Here the task is to delete at most k vertices to obtain a forest, a graph with bounded-degree, and a planar graph, respectively). We believe that this approach will provide many further tight kernel lower bounds in the future.

5.5 Notion of Kernelization Relaxed

As the number of problems unlikely to have polynomial kernels grows, several approaches to relax the notion of kernelization were made. The easiest way to do this is to desist from the somewhat artificial requirement, that the result of the procedure must be an instance of the same problem. More formally:

Definition 5.11. *A bikernelization* from a parameterized problem $\mathcal{P}$ to a parameterized problem $\mathcal{Q}$ is a polynomial time evaluable function A that on the input $(x,k) \in \Sigma^* \times \mathbb{N}$ produces an instance $(x',k') := A((x,k)) \in \Sigma^* \times \mathbb{N}$ such that

- (x',k') is in $\mathcal{Q}$ if and only if (x,k) is in $\mathcal{P}$, and
- there is a function $g : \mathbb{N} \to \mathbb{N}$ such that $|x'| \leq g(k)$ and $k' \leq g(k)$.

The instance (x',k') is called *a problem bikernel* and the function g is called *a size of the bikernel*.

Note that, similarly to Theorem 5.5, existence of a bikernel from $\mathcal{P}$ to a decidable problem implies that $\mathcal{P}$ is FPT. Moreover, if $\mathcal{P}$ and $\mathcal{Q}$ are NP-complete and the parameters are upper bounded by the size of the input, then $\mathcal{P}$ has a polynomial kernel if and only if there is a polynomial bikernel from $\mathcal{P}$ to $\mathcal{Q}$. Hence, this notion does not bring too much new.

In fact, the notion of bikernelization is very close to the so-called *annotated kernels*. *An annotation* is additional information added to handle partially solved instances during the run of an algorithm. An instance of the original problem can be viewed as a special instance of the annotated problem. On the other hand, although no polynomial kernel is known for the original problem, there can be a polynomial kernel for the annotated problem (sometimes called *annotated kernel*), as in this case some information can be stored in the annotation.

A well known example is the case of Dominating Set in Planar Graphs where we are given a planar graph and $k \in \mathbb{N}$ and the question whether there is a set of at most k vertices, such that each vertex not in this set has a neighbor in it (is dominated). Here the annotation divides vertices into two groups — those that are already dominated by vertices taken into a solution and removed from the graph — these vertices are left in the graph as they can still dominate some of the other vertices — those not dominated yet, which form the other group. This way we can obtain a linear kernel for (Annotated) Dominating Set in Planar Graphs [DF98]

Another approach is based on the observation that for praxis it is still desirable to produce polynomially many (in terms of the input size) polynomial (in terms of the parameter) kernels. Such a reduction is often called *Turing kernelization*. Fernau et al. [FFL$^+$09] showed that there is a cubic kernel for Rooted k-Leaf Out-Branching a variant of k-Leaf Out-Branching, where the root of the sought out-branching is given. Hence k-Leaf Out-Branching can be transformed into n Turing kernels, each of size $O(k^3)$, although it was shown unlikely to have a standard polynomial kernel.

There is an active research in the area. The already mentioned results of Dell and van Melkebeek [DvM10] show that the existence of a Turing kernelization asking $n^{1-\epsilon}$ queries each of size polynomial in k, for some $\epsilon > 0$ would imply the collapse of the polynomial-time hierarchy.

5.6 Similar approach - Win/Win

After applying the reduction rules, kernelization basically branches into two cases according to the size of the reduced input: If the reduced input is too large than the answer is trivial. If it is small, then the running time of an algorithm we have will be hopefully affordable. It is not necessary to lower exactly the input size. The only requirement is that the low value of the decisive measure provides us with a (known) reasonable algorithm. This approach is sometimes called *Win/Win* that is we "win" if the measure is high and also if it is low. The measure used is mostly some graph width.

We give an easy illustrative example for Max Leaf, where given a graph and $k \in \mathbb{N}$ we search for a spanning tree of the graph with at least k leaves, k being the standard parameter. We start by doing (arbitrarily) a bread-first search from some vertex. If the tree obtained has at least k leaves, we won. Otherwise each layer contains at most $k-1$ vertices. Note that the edges only appear inside layers and between two consecutive layers. Hence we can easily obtain a path-decomposition of width $2k-2$, forming a bag from each two consecutive layers. As Max Leaf is FPT with respect to the pathwidth (by a dynamic programming on the path decomposition [Bod93], see Section 6.2 for such algorithms), we win in this case also.

The very powerful "bidimensionality" approach that also significantly relies on the win/win paradigm is described in Section 6.7 as it is also based on the theory of minors.

Chapter 6

Further Algorithmic Methods

In this chapter we present the most important methods that are used to show fixed-parameter tractability.

6.1 Bounded Search Trees

If there is a method, that is similarly important in showing fixed-parameter tractability with respect to the solution size (or its dual) as kernelization, it is definitely the method of Bounded Search Trees. Or, if you want, you can call it *Branching* or *Recursive algorithms*. As there is not so much theory related to this method, we cut its presentation in space.

The basic idea is very simple: Identify a set of objects, one of which must be in the solution and try all possibilities to add one of them to a solution set. Continue recursively searching for a one-smaller solution on each of the partially solved instances. This gives a *search tree* of the recursive calls. This works for the primal parameterization, if the dual parameterization is used, we talk about objects that will not be a part of the solution

The important thing making this a special case of brute-force algorithms is that the size of the search tree can be bounded in terms of the parameter. As the depth of the tree is usually bounded by the solution size, it remains to bound the number of solution possibilities in each call of the procedure. The set of objects is usually constant size, but sizes bounded by the parameter are good as well. The number of leaves of the search tree is then bounded by the k-th power of the size of the set, while the number of internal vertices in the tree is upper-bounded by the number of leaves. The time needed inside one recursive call (corresponding to one node of the tree) is usually polynomial.

More involved algorithms usually add more than one object at once into the solution (at least in some branches) doing the analysis of the size of the tree more complicated. Also several different branching rules for different situation usually form the whole algorithm. As the branching algorithms are widely used

in the area of moderately exponential exact algorithms for hard problems, the ways to compute (an upper bound for) the running time can be found in any standard textbook on algorithms (see for example [KT05]). For most of the complicated branching algorithms it is hard to come with lower bound matching (or approaching) the upper bounds and thus the actual time complexity is rather unknown, which makes it harder to compare different branching algorithms.

The easiest example is the folklore $O(2^k \cdot n)$-algorithm for Vertex Cover. It is directly suggested by the definition of the problem — as each edge has to have at least one endpoint in the cover, we try both possibilities, delete the appropriate vertex from the graph (together with the incident edges), and continue recursively on the rest, searching for a cover of size at least by one smaller. Obviously there is no cover of size 0 for a graph with at least one edge, while for an edgeless graph the empty set is a cover of size 0. It is immediate that such a search tree has at most 2^k leaves and, thus, $O(2^k)$ nodes, the time needed to process each node being proportional to the number of vertices.

The algorithm can be improved by employing the idea that either a vertex is a part of the cover, or all its neighbors are. In this case, the bigger the degree of the vertex, the larger the progress we make (in one of the branches). Hence it is preferable to process the vertices of the highest degrees first. Once there is no vertex of degree at least 3 anymore, the minimum vertex cover for the graph can be found in linear time. Hence whenever we branch, one branch has the parameter reduced by at least 3, while the other by one. The resulting tree has $O(1.4656^k)$ nodes, the time spent in each node is still linear, yielding an $O(1.4656^k \cdot n)$-algorithm. This can be further improved to $O(1.4656^k \cdot k^2 + k \cdot n)$ by first using the Buss' Kernelization. Also note that $1.4656^k \cdot k^2$ is $O(1.4657^k)$ and therefore we can omit the factor polynomial in k as the base of the exponent was already rounded up.

Further example is d-Hitting Set, which also admits a natural search tree algorithm. Here for $d \in \mathbb{N}$ fixed constant and $k \in \mathbb{N}$ parameter we are given a family of sets, each with at most d elements and we should find at most k elements that hit every of the given sets. That is, the solution must contain at least one element out of each set. There is nothing easier than to take one set and try each element as the one that hits this set. Delete all sets hit by this element, decrease the parameter and continue recursively. We arrive at an $O(d^k)$ search tree for the problem.

Observe that a solution containing more elements of the set is considered in several branches of algorithm — in each branch that corresponds to some of the elements finally taken into the solution. We can actually partially avoid this inefficiency. To this end, we need the following reduction rule: Whenever an element u is contained in each set, where v is contained, delete v from all sets (without changing the value of the parameter), as it is never worse to take u whenever v should be taken. Now we force our algorithm to only consider the solutions containing the first element of the set (in some order) in the first

branch and delete it (as unusable) in all other branches. For each of the elements u different from the first one, there is always a set that contains the first element but not the element u. This set is not hit by the element u and has at most $d-1$ elements as the first element was deleted from it. Therefore it allows for better branching in the recursive call. This approach suggested in [NR03] brings the base of the exponent down to

$$\alpha(d) = \frac{d-1}{2}\left(1+\sqrt{1+\frac{4}{(d-1)^2}}\right),$$

which is a significant improvement at least for small values of d. For example $\alpha(3)$ roughly equals to 2.41.

It is important to note, that in the above example, the rule has to be applied at the beginning of each recursive call, as otherwise the condition can be violated during the execution. This increases the polynomial time needed in each node. On the other hand, this is greatly balanced by the improvement of the exponential factor. Furthermore this time can be decreased using the kernelization from Section 5.1. Usually (as in this case) the reduction rules are also used in between the branchings, which is sometimes called *interleaving* (of the kernelization and the search tree). This usually improves the performance both practically and theoretically. Quite typically an fpt-algorithm is formed by a set of rules, some of them being reduction rules (hopefully yielding a kernelization) and some of them being branching rules.

6.2 Dynamic Programming and Algorithmic Meta-Theorems

Dynamic programming is definitely the most successful technique for problems parameterized by something else than the solution-size or its dual. It is based on finding the solutions for the subproblems of the original problem, storing them in a table and then combining the solution for smaller subproblems to obtain a solution for larger subproblems.

Although the success of the method is very much connected with the success of the treewidth and other structural measures, we prefer to start by an example for Steiner Tree parameterized by the number of terminals to be connected, as this is very close to our results presented in the second part of the thesis. The algorithm we present is a modification of the famous Dreyfus-Wagner Algorithm [DW72] as presented in [DYW+07].

In Steiner Tree we are given a graph $G = (V, E)$ with integral weights on edges $w : E \to \mathbb{N}$, a set of terminals $T \subseteq V$ and integers $p \in \mathbb{N}$. The question is whether there is a tree of cost at most p containing all the terminals.

We use two tables S and D: For any $X \subseteq T$ and $r \in V$ the table entry $S(r, X)$ will store the smallest weight of a tree that contains vertices of $X \cup \{r\}$ (for a better imagination it can be viewed as rooted in r) and the table $D(r, X)$ will store an auxiliary number that in most cases equals the weight of a smallest tree (rooted in r) that contains vertices of $X \cup \{r\}$ in which either r has degree at least two or $r \in X$. As the answer is trivial for $|T| \leq 1$, we assume $|T| \geq 2$.

The algorithm proceeds through all sets $\emptyset \neq X \subseteq T$ from the smaller to the larger ones and for each of them does the following two things: First, if X is a singleton then we set $D(r, X) := S(r, X) := 0$ for $\{r\} = X$ and $D(r, X) := S(r, X) := \infty$ for each $r \in V \setminus X$. Otherwise, for every $r \in V$, we set

$$D(r, X) := \min_{\emptyset \neq Y \subsetneq X} (S(r, Y) + S(r, X \setminus Y)). \tag{6.1}$$

Second, we obtain the values of $S(r, X)$ for every $r \in V$ as

$$S(r, X) := \min_{v \in V} (D(v, X) + dist_G(r, v)), \tag{6.2}$$

where $dist_G(r, v)$ is the length of the shortest path between r and v in G. This is done by running Dijkstra's Algorithm [Dij59] on $S(r, X)$ initializing $S(r, X) := D(r, X)$ for every $r \in V$. The result of the whole algorithm is obtained as $S(r, T \setminus \{r\})$ for an arbitrary $r \in T$.

The overall correctness of the algorithm follows from the claim, that the algorithm correctly computes $S(r, X)$ for every $r \in V$ and $\emptyset \neq X \subseteq T$. This is easily seen if $X = \{x\}$ as in this case Dijkstra's Algorithm in fact computes the shortest path from r to x. We further proceed by the induction on the size of the set X.

Next we show that whenever the algorithm assigns a value t to a cell $S(r, X)$ or $D(r, X)$ of the tables, there is a connected subgraph of weight at most t containing the vertices $X \cup \{r\}$ justifying that. If the value of $D(X, r)$ is set according to the equation 6.1 then we obtain such a graph as the union of the graphs for $S(r, Y)$ and $S(r, X \setminus Y)$. Similarly if the recurrence 6.2 is used, the graph is obtained as the union of the shortest path from r to v with the graph for $D(v, X)$.

Finally suppose that $|X| \geq 2$, the claim holds for every nonempty proper subset of X and there is a tree T' containing $X \cup \{r\}$ of weights smaller than $S(r, X)$. Denote by v the vertex closest to r in T' that is either in X or of degree at least tree in T' and P_v the (possibly trivial) path between r and v.

If $v \in X$ then $T' \setminus (V(P_v) \setminus v)$ forms a tree for $\{v\} \cup (X \setminus \{v\})$ and hence is lower bounded by $S(v, (X \setminus \{v\})$, due to our assumptions, as $\emptyset \neq (X \setminus \{v\}) \subsetneq X$. Thus $w(T') \geq S(v, (X \setminus \{v\}) + w(P_v) = S(v, (X \setminus \{v\}) + S(v, \{v\}) + w(P_v) \geq D(v, X) + dist_G(r, v) \geq S(v, X)$ — a contradiction.

Otherwise $T' \setminus V(P_v)$ has more components. Then denote Y the subset of X contained in the first component. The subtree of T' induced by the first component together with v is a tree for $Y \cup \{v\}$ and therefore is lower bounded

by $S(v, Y)$ due to our assumptions, as $\emptyset \neq Y \subsetneq X$. Similarly, removing the first component and $P_v \setminus v$ we obtain a tree for $(X \setminus Y) \cup \{v\}$ lower bounded by $s(v, X \setminus Y)$. Thus $w(T') \geq s(v, Y) + s(v, X \setminus Y) + w(P_v) \geq D(v, X) + dist_G(r, v) \geq S(r, X)$, contradicting our assumptions.

As to the time complexity, the equation 6.1 yields two table lookups for each combination of $\emptyset \subsetneq Y \subsetneq X \subset T$ and each $r \in V$, hence the total time needed to evaluate the recurrence 6.1 can be bounded by $O(3^{|T|} \cdot n)$. Further, in each of the $2^{|T|}$ iterations of the cycle we execute once Dijkstra's Algorithm, which can be implemented to run in time $O(n \log n + m)$ [FT87b]. Hence the whole algorithm runs in time $O(3^{|T|} \cdot n + 2^{|T|}(n \log n + m))$.

As we have said, dynamic programming is, among parameterized algorithmics, mostly used in connection with the treewidth. To this end, usually the following modified decomposition is used:

Definition 6.1. A tree decomposition (T, σ) of a graph $G = (V, E)$ is called *nice* if it has a distinguished root and each node x is either a leaf or

- it has exactly one child y and there is a vertex $v \in V$ such that $V_x = V_y \cup \{v\}$ (such a node is called *introduce* node), or
- it has exactly one child y and there is a vertex $v \in V_y$ such that $V_x = V_y \setminus \{v\}$ (*forget* node), or
- it has exactly two children y and z such that $V_x = V_y = V_z$ (*join* node).

We use G_x to denote a subgraph of G induced by the vertices of V_x and all vertices that appear in the bags of the subtree of T rooted in x.

Due to the following observation we can always assume that the decomposition we are given is a nice decomposition with $O(k \cdot n)$ nodes.

Observation 6.2. *Given a tree decomposition (T, σ) of width k with $O(n)$ nodes, we can modify it to a nice tree decomposition with $O(k \cdot n)$ nodes in $O(k \cdot n)$ time.*

We demonstrate the use of dynamic programming on a problem parameterized by the treewidth on Independent Set. In Independent Set we are given a graph and search for a maximum size independent set. A subset of vertices is called *independent* if no two of the vertices are connected by an edge. We also assume that the tree decomposition of width $tw(G)$ is given on the input.

We associate with each node x a table A_x indexed by all subset of V_x. The number stored in the table on index S will represent the maximum independent set I in G_x that intersect V_x exactly in S or $-\infty$ if S itself is not an independent set.

We fill the tables from leaves of the decomposition to the root assuming that by the time we process a node, all its children were already processed (this is

usually called *bottom-up fashion*). For the leaves we set $A_x(S) := |S|$ if S is an independent set and $A_x(S) := -\infty$ otherwise. If x is an introduce node with child y, where the vertex v is introduced, we set again $A_x(S) := -\infty$ if S is not an independent set. Otherwise if v is contained in S, we set $A_x(S) := A_y(S \setminus \{v\}) + 1$ and for $v \notin S$ we let $A_x(S) := A_y(S)$. Similarly for a forget node forgetting vertex v and S an independent set we set $A_x(S) := \max\{A_y(S), A_y(S \cup \{v\})\}$. Finally, let x be a join node with children y and z and $S \subseteq V_x$. If either $A_y(S) = -\infty$ or $A_z(S) = -\infty$ then set $A_x(S) := -\infty$. Otherwise we let $A_x(S) := A_y(S) + A_z(S) - |S|$. The size of the maximum independent set in the graph is then the maximum number stored in the table of the root.

The algorithm obviously runs in time $O(2^k \cdot (k^2 + k \cdot n))$ as there are at most 2^{k+1} subsets of each V_x. The correctness follows from that the algorithm correctly fills all the tables. This is obvious for the leaves. Now we prove that for the other nodes under the assumption that it was already proven for all their children. For a forget node the independent set in G_x intersecting V_x in S intersects V_y in $S \setminus \{v\}$ and, hence, the correctness of the table directly follows from the correctness of the tables of the children. Similarly for the introduce node, the maximum independent set in G_x either contains v or not.

For the join node, assume that there is an independent set I in G_x intersecting V_x in S such that $|I| > A_x(S)$. Then $I \cap V(G_y)$ is an independent set in G_y intersecting V_y in S and thus $|I \cap V(G_y)| \leq A_y(S)$. Similarly $|I \cap V(G_z)| \leq A_z(S)$ and hence $|I| \leq A_y(S) + A_z(S) - |S| = A_x(S)$ –a contradiction. To finish the proof, note that if I_y is an independent set in G_y and I_z is an independent set in G_z both intersecting $V_x = V_y = V_z$ in S, then $I_y \cup I_z$ is an independent set in G_x as there are no edges between $G_y \setminus V_x$ and $G_y \setminus V_x$ (vertices of an edge are contained in one bag and, hence, can appear in only one subtree rooted at x).

There are many results similar to the example we gave. In fact, for any problem, that is expressible in the so-called Monadic Second-Order Logic (MSOL), a similar algorithm exists.

The *Monadic Second Order Logic (MSOL)* over graphs uses the union of the vertex set and the edge set as its domain and there are two unary predicates — $V(x)$ and $E(x)$ to distinguish, whether the object x is a vertex or an edge, respectively — and one binary $I(v, e)$, which is *true*, if and only if v is a vertex and e an edge incident to it. The quantification can be done over objects and sets of objects (unary predicates), but not over relations of higher arity.

Often this variant is called MSO_2 as there is another variant MSO_1, where the domain is only formed by the vertex set and only one binary relation $adj(u, v)$ is available, coding the adjacency of the vertices. This variant is less powerful, as it is unable to quantify over sets of edges.

The following result justifies the importance of the concept of tree decompositions:

Theorem 6.3 (Courcelle [Cou92])**.** *Given an MSO_2 formula φ, there is an al-*

gorithm that, given a graph G together with its tree-decomposition of width w, decides whether $G \models \varphi$, that is whether G is a model for φ, in time $O(f(\varphi, w) \cdot n)$, where the function f is independent of G (it only depends on w and φ) and n is the number of vertices of the graph G.

The function f involved in the theorem is so huge, that it makes the theorem mainly of theoretical interest. One-purpose dynamic programming algorithms as the one for Independent Set are to be used in praxis. The result can be also strengthened in the sense, that in same linear time we can actually also find the maximum of some linear objective function on the cardinalities of the involved set variables. Recall that the tree-decomposition of the graph can be obtained in linear time due to the result of Bodlaender [Bod96].

A similar result can be achieved with respect to the clique-width, we only have to give up the quantification over sets of edges. Even this result can be strengthened so that we can optimize a linear objective function of the cardinalities.

Theorem 6.4 (Courcelle, Makowsky, and Rotics [CMR00])**.** *Given an MSO_1 formula φ, there is an algorithm that, given a graph G together with its w-expression, decides whether $G \models \varphi$, in time $O(f(\varphi, w) \cdot (n + m))$, where the function f is independent of G, n is the number of vertices and m the number of edges of the graph G.*

Finally if our problem can be defined in terms of the First Order Logic (FO), where the quantification over sets is not possible at all, it can be decided on a fairly general graph classes. The following results were proved in [DK09], but similar results also appeared independently in [DKT09]. The results generalize some older results on graphs of locally bounded treewidth and locally excluding a minor.

Theorem 6.5 (Dawar and Kreutzer [DK09])**.** *Let $\mathcal{C}$ be a nowhere dense class of graphs. For every $\epsilon > 0$ there is a computable function $f : \mathbb{N} \to \mathbb{N}$ and an algorithm which, given $G \in \mathcal{C}$ and $\varphi \in FO$, decides whether G satisfies φ in time $f(|\varphi|) \cdot |G|^{1+\epsilon}$.*

Theorem 6.6 (Dawar and Kreutzer [DK09])**.** *Let $\mathcal{C}$ be a class of graphs of bounded expansion. There is a computable function $f : \mathbb{N} \to \mathbb{N}$ and an algorithm which, given $G \in \mathcal{C}$ and $\varphi \in FO$, decides whether G satisfies φ in time $f(\varphi) \cdot |G|$.*

6.3 Color Coding

It is no surprise that it is often easier to search only for solutions of a certain type. The color coding technique does that by first coloring the vertices of the

graph (adjacent vertices can receive the same color) and searching for a *rainbow* solution, that is, a solution in which no two vertices are of the same color. As we no longer search for a solution of certain size, but rather for a solution using once each of the colors, dynamic programming can be usually used for the search. Of course we do not always have to color vertices, there are many other parts of the instances to be colored, such as edges.

The following clever way of coloring ensures that each possible solution is colored in a rainbow manner at least in one of the colorings tried.

Definition 6.7. Let $k, n \in \mathbb{N}$. We say that $\mathcal{H}$ is a *k-perfect family of hash functions* from $\{1, 2, \ldots, n\}$ to $\{1, 2, \ldots, k\}$ if for every $S \subseteq \{1, 2, \ldots, n\}$ of size k there is an $h \in \mathcal{H}$ such that $h|_S$ is a bijection between S and $\{1, 2, \ldots, k\}$.

Lemma 6.8 (Naor; Alon, Yuster, and Zwick [AYZ95])**.** *There is a k-perfect family of hash functions from $\{1, 2, \ldots, n\}$ to $\{1, 2, \ldots, k\}$ of size $|\mathcal{H}| = 2^{O(k)} \cdot \log n$ that can be constructed in time $2^{O(k)} \cdot n \cdot \log n$.*

To illustrate the technique we present the algorithm for k-Path as stated in the original paper of Alon, Yuster, and Zwick [AYZ95]. Recall that in k-Path we search for a path of given length k (parameter) in a given graph. We start by constructing the hash functions as guaranteed by the previous lemma. For each h of the functions constructed we assign to each vertex $v \in V$ the color $h(v)$ in $\{1, \ldots k\}$ and search for a path formed by one vertex of each color.

This is done by a dynamic programming. With each vertex $v \in V$ we associate a table $S_v(A)$ indexed by all non-empty subsets A of the color set $\{1, \ldots k\}$. The value on position $S_v(A)$ determines whether there is a path in G having an endpoint in v an using one vertex of each color in A. The table is initialized by setting $S_v(\{h(v)\}) := \mathit{true}$ and $S_v(A) = \mathit{false}$ for every other singleton set A. Then the sets A are processed from smaller to larger, and $S_v(A)$ is set to *true* if and only if $h(v)$ is in A and there is a neighbor u of v with $S_u(A \setminus \{h(v)\}) = \mathit{true}$. There is a rainbow path if and only if $S_v(\{1, \ldots, k\})$ is *true* for some $v \in V$.

It is not hard to see that the dynamic programming works correctly and runs in time $O(2^k \cdot k \cdot |E|)$, we omit the proof here. The k-perfectness of the hash family ensures that if a path of length k exists, then it becomes a rainbow path by at least one function in the family and, hence, is found by the dynamic programming. Therefore the whole algorithm runs in time $O(2^{O(k)} \cdot m \cdot \log n)$.

The idea used in the example can be generalized to any class of graphs with low treewidth as follows:

Theorem 6.9 (Alon, Yuster, and Zwick [AYZ95])**.** *Let H be a directed or undirected graph on k vertices with treewidth t. Let $G = (V, E)$ be a (directed or undirected) graph. A subgraph of G isomorphic to H, if one exists, can be found in time $2^{O(k)}|V|^{t+1} \log |V|$.*

6.4 Integer Linear Programming

In 1983 Lenstra [Len83] has shown that Integer Linear Programming (or ILP shortly) is FPT with respect to the number of variables of the instance. In Integer Linear Programming, given a $p \times n$-matrix A and an n-vector b, one asks for a vector $x \in \mathbb{N}^p$ such that $Ax \leq b$ (coordinatewise). The result was later improved by several authors (see [AWW02] for survey) to obtain the following theorem:

Theorem 6.10 (Kannan [Kan87])**.** Integer Linear Programming *can be solved using $O(p^{2.5p+o(p)} \cdot L)$ arithmetic operations and space polynomial in L, where L is the number of bits of the input.*

Theorem 6.10 is very powerful to be used as a subroutine in other fpt-algorithms, often yielding somewhat surprising results. Recently it is frequently used in connection with structural parameterizations such as the vertex cover number. Our example is not an exception from that. We present the result of Fiala, Golovach, and Kratochvíl [FGK09a] showing that Equitable Coloring, where we ask for an r-coloring of the given graph with the sizes of any two color classes differing by at most one, is FPT with respect to the vertex cover number $vc(G)$.

Let C be a vertex cover of size $vc(G)$ of the graph G, and let $\{I_1, \ldots, I_t\}$ be the partition of the remaining vertices according to their neighborhoods ($t \leq 2^{vc(G)}$ as each vertex not in C has neighbors only in C). Set $s := \lfloor \frac{n}{r} \rfloor$ and $l := n - rs$. We search for a coloring with l color classes of the size $s+1$ and $r-l$ classes of the size s.

Assume that only $r' := \min\{vc(G), r\}$ colors can be used on the vertices of C, namely the colors $1, \ldots, r'$. Even more specifically, we assume that among these color classes $1, \ldots, l'$, where $l' \leq \min\{r', l\}$, are of size $s+1$ and the others are of size s. Whenever there is an equitable r-coloring of G one can easily derive an equitable coloring fulfilling the above assumptions by renaming the colors. Note that (in the case $r > r'$) the colors $r'+1, \ldots, r$ can be used on $G \setminus C$ arbitrarily and, hence, we do not have to care about these color classes once there is the right number of vertices left for these classes.

For each $l' \leq \min\{r', l\}$ and each proper coloring $V_1, \ldots, V_{r'}$ of C we construct a system of linear integer inequations with tr' variables $x_{i,j}$, for $i \in \{1, \ldots, t\}$ and $j \in \{1, \ldots, r'\}$. The variable $x_{i,j}$ will denote the number of vertices of the color j in the set I_i. The inequations are as follows:

$$\begin{aligned} x_{i,j} &\geq 0, \\ x_{i,j} &= 0, \text{ if color } j \text{ is used in the neighborhood of } I_i, \\ \sum_{i=1}^{t} x_{i,j} &= s + 1 - |C \cap V_j|, \text{ if } j \in \{1, \ldots, l'\}, \end{aligned}$$

$$\sum_{i=1}^{t} x_{i,j} \;=\; s - |C \cap V_j|, \text{ if } j \in \{l+1, \ldots, r'\}.$$

It is not hard to see that there is an integer solution to the above inequations if and only if there is an equitable coloring of G extending the initial coloring of C. Also the ILP instance can be constructed in polynomial time given the coloring of C. Since C has at most $(r')^{vc(G)}$ colorings and the number of variables is at most $r' \cdot 2^{vc(G)}$, Equitable Coloring can be solved in FPT-time.

6.5 Iterative Compression

Iterative Compression is a method used almost exclusively for vertex deletion problems parameterized by the size of the solution, where the task is to delete some vertices from the graph to achieve some property. It iteratively produces a solution for subproblems that is already one vertex bigger than the solution we search for. Then it compresses the solution to hit the bound. If this is not possible then there is no solution for the whole instance. Otherwise some more vertices are considered in the subproblem so that the solution is again too big and the next iteration is executed.

Our example is for Odd Cycle Transversal, where we aim to delete at most k (parameter) vertices from a given graph G in order to make it bipartite. The algorithm was first proposed by Reed et al. [RSV04] and the simplified version we present appeared in [LSS09].

The main part of an algorithm based on iterative compression is the so-called *compression step*. Here, given a solution of size $k+1$ we are trying to find a solution of size at most k or to show that no such solution exists. Before presenting it, let us show how to solve the whole problem once we have an algorithm A for the compression step.

Assume $V(G) = \{v_1, \ldots v_n\}$ and for $i \in \{k+2, \ldots n-1\}$ denote by G_i the graph $G[\{v_1, \ldots, v_i\}]$. First observe that for the graph G_{k+2} any set of size k constitutes a solution. Let us denote one of them S_{k+2}. Now for $i \in \{k+2, \ldots n-1\}$ assume we have a solution S_i of size at most k for G_i and we want to find one of size at most k for G_{i+1}. The set $S'_{i+1} := S_i \cup \{v_{i+1}\}$ is a solution for G_{i+1} of size at most $k+1$. If it is of size at most k we denote $S_{i+1} := S'_{i+1}$. Otherwise we use the compression step to obtain a solution S_{i+1} of size at most k for G_{i+1} being given the solution S'_{i+1} of size $k+1$. If there is no such solution for G_{i+1} then obviously there is no solution of size k also for the whole graph G. At the end we either obtain a size-k solution S_n for $G_n = G$ or we know that there is no solution of size at most k.

It remains to solve the compression step. Assume that we are given a solution S' of size $k+1$ for a graph G and we search for solution S of size k. We start be trying all possible partitions of S' into $T \cup L \cup R$. The vertices in T will be a

part of the new solution, while the vertices of L and R will be on the left and the right side of the bipartition provided by the new solution, respectively (they will definitely not be a part of the solution). If there is an edge inside $G[L]$ or $G[R]$ then there is no chance to find a solution corresponding to this partition and we continue with a further partition. Otherwise we make use of the following fact:

Fact 6.11. *If $H = (V_1 \cup V_2, E)$ is a bipartite graph with the partitions V_1 and V_2 then*

- *any trail from V_i to V_i has even length ($i = 1, 2$) and*
- *any trail from V_1 to V_2 has odd length.*

The graph $G \setminus S'$ is bipartite. Let us call the partitions A and B, and denote A_L, B_L the neighbors of the set L in A and B, respectively. Similarly A_R and B_R denote the neighbors of R. We further need the following lemma:

Lemma 6.12. *If X is a subset of $V \setminus S'$ such that $G \setminus (T \cup X)$ is bipartite with partitions V_L and V_R such that $L \subseteq V_L$ and $R \subseteq V_R$, then in $G \setminus (S' \cup X)$ there are no paths between A_L and B_L, A_R and B_R, A_L and A_R, and between B_L and B_R.*

Proof. Assume there is a path between A_L and B_L, then Fact 6.11 implies it has an odd length as it goes from one partition of $G \setminus S'$ to the another. Hence it can be prolonged to an odd trail from L to L in $G \setminus (T \cup X)$ which is a contradiction with $G \setminus (T \cup X)$ being bipartite. Similarly for the other combinations. □

Lemma 6.13. *If X is a subset of $V \setminus S'$ such that in $G \setminus (S' \cup X)$ there are no paths between A_L and B_L, A_R and B_R, A_L and A_R, and between B_L and B_R, then $G \setminus (T \cup X)$ is bipartite with partitions V_L and V_R such that $L \subseteq V_L$ and $R \subseteq V_R$.*

Proof. First note that in this case each path from L to L with internal vertices from $V \setminus (S' \cup X)$ is even. The same holds for such a path from R to R, while any such path from R to L is odd. Thus if $G \setminus (T \cup X)$ is bipartite then the partitions V_L and V_R can be taken such that $L \subseteq V_L$ and $R \subseteq V_R$.

Suppose that there is a cycle C in $G \setminus (T \cup X)$. If $C \cap (L \cup R) \neq \emptyset$ then C is even, as the graph $G \setminus S'$ is bipartite. Otherwise denote $v_1, \ldots, v_t$ vertices of $C \cap (L \cup R)$ in order as they appear on the cycle, and for simplicity set $v_0 := v_t$. The length of the cycle can be counted as $|E(C)| = \sum_{i=0}^{t-1} d_C(v_i, v_{i+1})$, where $d_C(v_i, v_{i+1})$ is the distance between v_i and v_{i+1} taken along the cycle C. It remains to observe that the number of indices i with $v_i \in L$ and $v_{i+1} \in R$ equals the number of i's with $v_i \in R$ and $v_{i+1} \in L$ and hence C is again even. □

Due to the above lemma, to finish the compression step it is enough to find X of size at most $k - |T|$ such that in $G \setminus (S' \cup X)$ there are no paths between the mentioned pairs of sets. But this is equal to finding a cut of size $k - |T|$ in $G \setminus S'$ between $A_L \cup B_R$ and $A_R \cup B_L$. This can be done in time $O(k \cdot m)$

using standard flow techniques [FF56]. As this is done at most 3^{k+1} times in the compression step and the compression step is executed at most n times, we obtain an $O(3^k \cdot k \cdot n \cdot m)$ running-time for the whole algorithm.

Iterative compression was several times the break-through tool on problems resisting the research attacks for a long time before. A survey of known results based on iterative compression can be found in [GMN09].

6.6 Greedy Localization

Similarly to the previous technique, greedy localization uses some solution that is not good enough as a starting point for the search for the desired one. In contrast to the previous method, this time we use an inclusion maximal solution that is found greedily.

The idea is best illustrated on Packing 3-Sets where we are given a system $\mathcal{C}$ of three-element subsets of a finite set S and an integer $k \in \mathbb{N}$. The task is to find a subsystem $\mathcal{C}'$ of at least k mutually disjoint sets. The result is due to Jia et al. [JZC04] and actually uses the idea twice.

The algorithm first greedily locates an inclusion maximal subsystem $\mathcal{C}'_0$ of system $\mathcal{C}$ formed by pairwise disjoint sets. We assume $|\mathcal{C}'_0| < k$ as otherwise we are done. Each set in an optimal solution must contain at least one element of $\bigcup \mathcal{C}'_0$ as $\mathcal{C}'_0$ is maximal. Hence an optimal solution fits into one of the following patterns:

$$\mathcal{C}^* = \{\{a_1, *, *\}, \{a_2, *, *\}, \ldots, \{a_k, *, *\}\}, \text{ where } a_i \in \bigcup \mathcal{C}'_0 \text{ are distinct.}$$

Our algorithm will try all such patterns. Now assume that we have a pattern P with some positions filled and some still carrying a wild-card symbol $*$. We greedily try to fill the pattern into a pattern P'. That is we take the incomplete sets in the pattern one by one and look for a 3-set of $\mathcal{C}$ that contains the elements prescribed by the pattern and the other elements in it are not used anywhere else in the partially filled pattern — that is they are nor in the prescribed pattern P neither they have been already used to fill other set in P'.

If we succeed to fill the whole pattern, we have a solution. Otherwise consider the set S which we were unable to fill. If every its completion contains elements of some other part of P, that means the pattern P cannot be realized. Otherwise to complete this set we have to use some elements already added to the other sets in the pattern P'. As an optimal solution fitting the pattern (if it exists) must add one of these elements into S we try all the possibilities to do so. The element is added permanently to P and the algorithm recourses on it. This way a solution must be revealed if one exists.

As to the running time of the algorithm, the first greedy search runs in time $O(|\mathcal{C}|)$. As $|\mathcal{C}'_0| < 3k$ there are at most $\binom{3k}{k}$ "initial patterns". We are trying to fill each of them by the recursive algorithm, that first runs a greedy search

in $O(|\mathcal{C}| \cdot k)$ time and then possibly runs some recursive calls, each corresponding to an element added to some set. Hence there are less than $2k$ of such calls, and for each of them the pattern is filled by one more element. Hence at latest on level $2k$ of the recursion it is completely full and it either constitutes a solution or some of its sets is not in $\mathcal{C}$ and the pattern can be discarded. Thus the overall running time of the algorithm is $O((3k)^k \cdot (2k)^{2k} \cdot |\mathcal{C}|)$.

6.7 Using the Theory of Minors, Bidimensionality

The theory of minors, developed mainly by Robertson and Seymour in their famous Graph Minors series (started by [RS83]), is nowadays one of the most important parts of the whole graph theory. It is no surprise, that it can be also used in parameterized algorithmics. For the basic definitions related to the minor theory refer to Section 2.2.

The following two essential results of Robertson and Seymour are of special interest also for parameterized algorithmics:

Theorem 6.14 (Robertson and Seymour [RS04])**.** *Any class of graphs has finitely many minimal elements with respect to the minor relation. In particular, if $\mathcal{C}$ is a minor closed class of graphs, then there is a finite set $\mathcal{F}(\mathcal{C})$ of* obstructions *such that $G \in \mathcal{C}$ if and only if there is no $H \in \mathcal{F}(\mathcal{C})$ such that H is a minor of G.*

Theorem 6.15 (Robertson and Seymour [RS95]; shorter proof published recently in [KW10])**.** *There is an algorithm that decides whether H is a minor of G in time $O(f(H) \cdot |V(G)|^3)$, for some function f.*

Putting Theorems 6.14 and 6.15 together we obtain the following corollary:

Corollary 6.16. *Any minor closed graph class $\mathcal{C}$ can be recognized in a cubic time.*

Proof. Due to Theorem 6.14, graph G is in $\mathcal{C}$ if and only if there is no minor H of G in $\mathcal{F}(\mathcal{C})$. By Theorem 6.15, this can be tested in time $O\left(\left(\sum_{H \in \mathcal{F}(\mathcal{C})} f(H)\right) \cdot |V(G)|^3\right)$. It remains to note that $\sum_{H \in \mathcal{F}(\mathcal{C})} f(H)$ is a finite constant since $\mathcal{F}(\mathcal{C})$ is finite. □

It is very easy to use Corollary 6.16 in parameterized complexity. It is enough to show that a set of yes-instances with a particular value of the parameter is minor closed.

Example 6.17. Class of graphs having a vertex cover of size at most k is minor closed — hence it can be decided in $O(f(k) \cdot n^3)$-time whether an n-vertex graph G has a vertex cover of size at most k. To see the former it is enough to show

that if G has an k-vertex cover C then for every $v \in V(G)$ and every $e \in E(G)$ the graphs $G \setminus \{v\}$, $G \setminus e$, and $G \cdot e$ have vertex cover of size at most k. For the first two $C \setminus \{v\}$ and C constitute such a cover, respectively. If $e = \{x, y\}$ is an edge of G, the vertex z to corresponds to the union of x and y in $V(G \cdot e)$ and $x \in C$ or $y \in C$ then $C \setminus \{x, y\} \cup \{z\}$ is such a cover for third case. Otherwise we can use simply C.

Remark 6.18. In Example 6.17 we have shown, that it can be decided in $O(f(k) \cdot n^3)$-time whether an n-vertex graph G has a vertex cover of size at most k, but we have shown no algorithm for that. We only know that there is a cubic algorithm for each fixed k, but the result gives no way to find the algorithm. Note that our definition of the class FPT requires the existence of a single algorithm for all values of the parameter. The class of problems for which there is a constant c and for each fixed value of the parameter there is an $O(n^c)$-algorithm is often called *non-uniform FPT*.

The above result can be simply generalized to a whole class of graph problems:

Definition 6.19. Let $\mathcal{C}$ be a class of graphs. A graph G is in the class $\mathcal{C} \oplus kv$ if and only if there is a set $S \subseteq V(G)$ of size at most k such that $G \setminus S$ is in $\mathcal{C}$.

Observation 6.20. *If $\mathcal{C}$ is a minor closed class, then so is $\mathcal{C} \oplus kv$.*

It is easy to see that edgeless graphs, forests and planar graphs form minor-closed families of graphs. As a corollary of this fact together with the above observation we get that not only Vertex Cover, but also Feedback Vertex Set or Planar Deletion are in non-uniform FPT (Here the question is whether it is possible to delete at most k vertices to obtain an edge-less graph, a forest, and a planar graph, respectively).

The most powerful usage of the theory of minors is in the combination with the graph width measures for the problems restricted to planar graphs, graphs with bounded genus or graphs excluding a fixed graph as a minor [DFHT05]. We present only the planar case. The other cases, although similar in the basic ideas, are more complicated in details.

Theorem 6.21 (Robertson, Seymour, and Thomas [RST94])**.** *Let $l \geq 1$ be an integer. Every planar graph of treewidth at least $6l - 4$ contains an $(l \times l)$-grid as a minor.*

If we forbid ourselves edge deletions and instead of each vertex deletion we contract the vertex to be deleted with some of its neighbors, we arrive at the following corollary.

Corollary 6.22. *Let $l \geq 1$ be an integer. Every connected planar graph of treewidth at least $6l - 4$ can be transformed into a partially triangulated $(l \times l)$-grid using only edge contractions.*

Using the above theorem we can derive a fairly general result for problems that are bidimensional. Intuitively, a parameterized graph problem is bidimensional if it is closed under (at least some) minor operations and if the parameter for grids grows linearly with the number of vertices of the grid. More formally:

Definition 6.23 (Demaine et al. [DFHT05])**.** A parameter P assigning an integer to each graph is *minor bidimensional* with density δ if

(1) $P(H) \leq P(G)$ whenever a graph H is a minor of a graph G, and

(2) for the $(l \times l)$-grid R, $P(R) = (\delta l)^2 + o(l^2)$.

The parameter P is called *contraction bidimensional* with density δ if

(1) contracting an edge in a graph G cannot increase $P(G)$,

(2) $P(C)$ is at most $P(G)$ whenever C is a connected component of G,

(3) for any partially triangulated $(l \times l)$-grid R, $P(R) \geq (\delta l)^2 + o(l^2)$, and

(4) δ is the smallest real number for which this inequality holds.

The parameter P is called *bidimensional* if it is either minor or contraction bidimensional.

It is usually very easy to prove that some parameter is bidimensional. We have already shown that the vertex cover number satisfies the condition (1) for the minor bidimensionality. If we notice, that an $(l \times l)$-grid contains l disjoint parallel paths each of length l and, thus, a matching of size $l \cdot \lfloor l/2 \rfloor$ we arrive at the following observation.

Observation 6.24. *Vertex cover number $vc(G)$ is minor bidimensional with density $1/\sqrt{2}$.*

Similarly one can see that the minimum size of a dominating set in a graph $ds(G)$ is closed under contractions of edges and taking connected components. Furthermore, a partially triangulated $(l \times l)$-grid contains $(l-2)^2$ inner vertices, among which no vertex can dominate more than 9 (including itself). Hence $ds(G)$ is contraction bidimensional.

The following theorem is a simple corollary of Theorem 6.21.

Theorem 6.25 (Demaine et al. [DFHT05])**.** *Let P be a bidimensional parameter. Then for any planar graph G, $tw(G) = O(\sqrt{P(G)})$.*

To use the above theorem we need to able to determine the treewidth of a graph very quickly, the algorithm of Bodlaender [Bod96] is not fast enough for our purpose. Fortunately, treewidth can be $\frac{3}{2}$ approximated in planar graphs.

Theorem 6.26 (Seymour and Thomas [ST94]; Gu and Tamaki [GT05])**.** *There is an algorithm that given a planar graph G outputs in a cubic time a tree decomposition of width w with the guarantee, that the treewidth $tw(G)$ is at least $2w/3$ (the algorithm actually computes an optimal so-called branch decomposition of the graph G, from which such a tree decomposition can be derived).*

The last important ingredient in a subexponential algorithm for computing a bidimensional parameter is a fast algorithm for the computation of the parameter on graphs of bounded treewidth. We need an algorithm with running time $2^{O(tw(G))} \cdot poly(n)$ or at least $2^{o(tw(G)^2)} \cdot poly(n)$. Such an algorithm for the vertex cover number can be derived from the algorithm for Independent Set in Section 6.2 and for (Annotated) Dominating Set an algorithm can be devised similarly.

Theorem 6.27 (Demaine et al. [DFHT05])**.** *If P is a bidimensional parameter and there is an $2^{O(tw(G))} \cdot poly(|V(G)|)$-time algorithm for its computation, then it is possible to decide in time $2^{O(\sqrt{k})} \cdot poly(|V(G)|)$ for a planar graph G whether it has $P(G) \leq k$.*

Proof. We prove the theorem for a parameter P that is minor bidimensional, the case of contraction bidimensionality is similar. We assume that $P(R) = (\delta l)^2 + o(l^2)$ for every $(l \times l)$-grid R. Hence there is some l_0 and $0 < \delta_0 \leq \delta$ such that for every $l \geq l_0$ and every $(l \times l)$-grid R we have $P(R) \geq (\delta_0 l)^2$.

We use the algorithm from Theorem 6.26 to obtain a tree-decomposition of width w. Due to Theorems 6.26 and 6.21 we know that the graph contains a grid of side at least $(2w/3)/6 = \frac{w}{9}$. If $\frac{1}{9}w \geq l_0$ and $(\frac{1}{9}\delta_0 \cdot w)^2 > k$ then the answer is no, as P is greater than k already on the $\frac{w}{9} \times \frac{w}{9}$-grid contained in G. Otherwise, we have a tree-decomposition of width at most $\max\{9l_0, 9\delta_0^{-1}\sqrt{k}\}$ and, hence, the problem can be solved in time $2^{O(\max\{9l_0, 9\delta_0^{-1}\sqrt{k}\})} \cdot poly(|V(G)|) = 2^{O(\sqrt{k})} \cdot poly(|V(G)|)$. □

The running times can be sometimes further improved, if the dynamic programming itself is designed more carefully using the properties of planar graphs. The results further generalize in a certain way to a classes of graphs of higher genus and classes excluding some fixed graph as a minor. See [DFHT05] for details of such generalizations.

Recent results [FLST10] also show that if the problem satisfies some further conditions, then one can even obtain a polynomial kernel for it, by replacing large parts of the graph with low treewidth and small boundary by equivalent smaller parts.

Chapter 7

Intractability

As with the classical complexity, there is no way known to show unconditionally the non-existence of an fpt-algorithm for a certain problem (such a result would imply P≠NP). Instead we use to show that the fixed-parameter tractability of the problem considered would mean the same result for a wide class of other problems. For that purpose we first need a notion of parameterized reduction.

7.1 Reductions, Classes

Definition 7.1. A *parameterized reduction* (fpt-reduction) from a parameterized problem $\mathcal{P}$ to a parameterized problem $\mathcal{Q}$ is an algorithm that on an instance[1] $(x, k) \in \Sigma^* \times \mathbb{N}$ of $\mathcal{P}$ produces in time $f(k) \cdot |x|^{O(1)}$ an instance $(x', k') \in \Sigma^* \times \mathbb{N}$ of $\mathcal{Q}$ such that

- $(x, k) \in \mathcal{P}$ if and only if $(x', k') \in \mathcal{Q}$, and
- $k' \leq g(k)$,

where the functions f and g depend only on k. A parameterized problem $\mathcal{P}$ is *fpt-reducible* to a parameterized problem $\mathcal{Q}$ if there is a parameterized reduction from $\mathcal{P}$ to $\mathcal{Q}$.

Remark 7.2. We use the term reduction for both the classical polynomial time many:one reductions and the parameterized ones. The meaning should be clear from the context.

The essential property of parameterized reductions is that whenever $\mathcal{P}$ reduces to $\mathcal{Q}$ (by an fpt-reduction) and $\mathcal{Q}$ is in FPT, then $\mathcal{P}$ is FPT as well. Note that the second condition of Definition 7.1 is necessary for that purpose. It also worth noticing, that kernelization (Definition 5.1) and polynomial parameter transformation (Definition 5.9) are both special cases of parameterized reduction.

[1] We give the definition of the parameterized reduction only for the case when parameter is a single integer, but in the sense of Remark 3.5 it generalizes also to all other cases.

Unlike the classical complexity, in parameterized complexity the classes of intractability are primarily determined by their canonical complete problem, not by a model of computation. To this end, we first need the following classes of Boolean formulae.

Definition 7.3. A Boolean formula φ is 1-*normalized* if it is in the form of a conjunction of disjunctions of two literals (it is an instance of 2-SAT).
For $t \in \mathbb{N}, t > 1$, we say that a formula φ is t-*normalized* if it is in the form of a conjunction of disjunctions of conjunctions of.... of literals, where the conjunctions and the disjunctions alternate $t-1$ times (a formula in conjunctive normal form is 2-normalized).
A formula is called *monotone*, if it contains no negations and *antimonotone* if every literal in it is a negation of a variable.
A *weight* of a Boolean assignment $a : \{x_1, \ldots, x_n\} \to \{true, false\}$ is the number of variables set to *true*.

Fundamental problems of parameterized intractability are the following:

Weighted t-Normalized Satisfiability
Input: A t-normalized Boolean formula φ and $k \in N$.
Question: Is there a satisfying assignment for φ of weight exactly k?

Weighted Satisfiability
Input: A Boolean formula φ and $k \in N$.
Question: Is there a satisfying assignment for φ of weight exactly k?

Weighted Circuit Satisfiability
Input: A Boolean decision circuit C and $k \in N$.
Question: Is there a satisfying assignment for C of weight exactly k?

In all cases we use the weight of the sought solution k as a parameter. We will further mention Weighted Monotone t-Normalized Satisfiability and Weighted Antimonotone t-Normalized Satisfiability, the definitions of these problems should be clear.

Now we are ready to introduce the basic classes of parameterized intractability:

Definition 7.4. For every $t \in \mathbb{N}$ the class W[t] consists of all parameterized problems that are fpt-reducible to Weighted t-Normalized Satisfiability.
The classes of parameterized problems reducible to Weighted Satisfiability and Weighted Circuit Satisfiability are called W[Sat] and W[P], respectively.
We say that a parameterized problem $\mathcal{P}$ is W[t]-hard if every problem in W[t] can be FPT-reduced to $\mathcal{P}$ and W[t]-complete if it is W[t]-hard and in W[t]. Similarly for W[Sat] and W[P].

Immediately from the definitions one can get the following hierarchy of the parameterized complexity classes.

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \ldots \subseteq \text{W[t]} \subseteq \ldots \subseteq \text{W[Sat]} \subseteq \text{W[P]} \subseteq \text{XP}$$

The reduction of an FPT problem to Weighted 1-Normalized Satisfiability can be done by first solving the problem (by its fpt-algorithm) and outputting a constant-size instance with the same answer. For the last inequality it suffices to use a trivial algorithm for Weighted Circuit Satisfiability that tries all weight k assignments and checks whether any of them is satisfying.

All of the above inequalities are supposed to be strict but so far we only know that FPT $\subsetneq$ XP [DF98]. It is also interesting that it is not known, whether an eventual collapse of two classes would propagate, either upwards or downwards.

As the W-classes serve to show that some problem is presumably not in FPT, the following class can be used to show that some problem is not even in the class XP.

Definition 7.5. A parameterized problem $\mathcal{P} \subseteq \Sigma^* \times \Sigma^*$ is para-NP-complete if there is a string $k_0 \in \Sigma^*$ such that the k_0-*th slice* of $\mathcal{P}$, that is the set $\{(x, k_0) \mid (x, k_0 \in \mathcal{P}\}$, is NP-complete.

A classical example of a para-NP-complete problem is Graph Coloring parameterized by the number of colors to be used, as it is known to be NP-complete even for 3 colors [GJ79].

It is not hard to see, that if there is a para-NP-complete problem in XP, then P=NP.

7.2 Monotone/Antimonotone Collapse

The following well known theorem gives an overview what happens if we require all literals of the formula to be positive or all of them to be negative.

Theorem 7.6 (Downey, Fellows [DF95a, DF95b]; Monotone/Antimonotone Collapse)**.** *Let $t \in \mathbb{N}$. Then* Weighted Monotone t-Normalized Satisfiability *is*

- *W[t]-complete if t is even,*
- *W[$t-1$]-complete if t is odd and $t > 1$, and*
- *FPT for $t = 1$.*

Conversely Weighted Antimonotone t-Normalized Satisfiability *is*

- *W[$t-1$]-complete if t is even, and*
- *W[t]-complete if t is odd.*

It can be easily seen that every instance of WEIGHTED ANTIMONOTONE 1-NORMALIZED SATISFIABILITY can be viewed as an instance of k-INDEPENDENT SET (or k-CLIQUE equivalently) and vice versa and, hence, k-CLIQUE is W[1]-complete. It is only slightly harder to show, that DOMINATING SET (also with standard parameterization) is W[2]-complete. Hence these two problems form a graph problem basis for W[1] and W[2], the two complexity classes vast majority of intractable natural problems fall in. There are also some quite natural problems known to be W[t] for all $t \in \mathbb{N}$, W[Sat] or W[P] complete, but the other classes are mainly of theoretical interest.

To give a flavor of a parameterized reduction, we prove the theorem for WEIGHTED ANTIMONOTONE 1-NORMALIZED SATISFIABILITY, which implies that CLIQUE is W[1]-complete. Obviously it suffices to show the hardness.

Theorem 7.7 (Downey, Fellows [DF95b])**.** WEIGHTED ANTIMONOTONE 1-NORMALIZED SATISFIABILITY *is W[1]-hard.*

The proof is a modification of the original proof from [DF95b], where it was stated in a much more general way.

Proof. We will reduce WEIGHTED 1-NORMALIZED SATISFIABILITY, as expected. Let (φ, k) be an instance of this problem with variables $x_1, \dots x_n$. We will construct an equivalent instance ψ, k' of WEIGHTED ANTIMONOTONE 1-NORMALIZED SATISFIABILITY. For simplicity we assume $2 \leq k \leq n$.

The variables of ψ form k blocks $A^l = \{a^l_1, \dots, a^l_n\}$ for $1 \leq l \leq k$ and $k-1$ blocks $B^l = \{b^l_{i,j} \mid 1 \leq i, j \leq n\}$ for $1 \leq l \leq k-1$. We set $k' = 2k-1$ as from each of the blocks one variable is to be *true*. True variables in A blocks represent the variables of φ set to *true* and the variables in B blocks represent gaps between them. In particular, if a^l_i is *true*, then the *true* variable in B^l must be $b^l_{i,j}$ for some j. Conversely if $b^l_{i,j}$ is *true*, then in A^{l+1} the variable a^{l+1}_{i+j} must be *true*.

These restrictions are enforced by clauses as follows:

- There is at most one variable *true* in each block: for every $1 \leq l \leq k$ and all $1 \leq i, j, i', j' \leq n$ add to ψ the clause $(\neg a^l_i \vee \neg a^l_{i'})$ if $i \neq i'$ and the clause $(\neg b^l_{i,j} \vee \neg b^l_{i',j'})$ if $l \leq k-1$ and $(i,j) \neq (i',j')$.

- The variable selected in A^l enforces the selection in B^l: for every $1 \leq l \leq k-1$ and every $1 \leq i, i', j \leq n$ add the clause $(\neg a^l_i \vee \neg b^l_{i',j})$ if $i' \neq i$

- The variable selected in B^l enforces the selection in A^{l+1}: for every $1 \leq l \leq k-1$ and every $1 \leq i, i', j \leq n$ add the clause $(\neg b^l_{i,j} \vee \neg a^{l+1}_{i'})$ if $i' \neq i+j$

Since the construction can only handle gaps of positive size, we know that if both a^l_i and $a^{l+1}_{i'}$ are set to *true*, then $i < i'$.

The most important thing to realize is that the fact that some variable of φ is set to *false* is represented by a certain variable set to *true* in our construction. Namely setting x_r to *false* can be represented by that

- some of the variables $a^1_{r+1}, \ldots, a^1_n$ is set to *true*, or
- some of the variables $b^l_{i,j}$ with $i < r < i+j$ and $1 \leq l \leq k-1$ is set to *true*, or
- some of the variables $a^k_1, \ldots, a^k_{r-1}$ is set to *true*.

Denote the set of the above mentioned variables by S_r.

Now consider a clause C of φ. If (for some $1 \leq p, r \leq n$) the clause C is of the form

- $(x_p \vee x_r)$ then we add into ψ the clauses $(\neg y \vee \neg z)$ for every $y \in S_p$ and $z \in S_r$
- $(\neg x_p \vee x_r)$ (or $(x_r \vee \neg x_p)$) then we add into ψ the clauses $(\neg a^l_p \vee \neg y)$ into ψ for every $1 \leq l \leq k$ and $y \in S_r$
- $(\neg x_p \vee \neg x_r)$ then we add into ψ the clauses $(\neg a^l_p \vee \neg a^{l'}_r)$ for every $1 \leq l, l' \leq k$.

We claim that if the clause C is not satisfied in the assignment examined, then at least one of the added clauses is not satisfied as well. For the first case, this holds since if neither x_p nor x_r is set to *true*, then at least one variable $y \in S_p$ is set to *true* and at least one variable $z \in S_r$ is set to *true* and $(\neg y \vee \neg z)$ is not satisfied. Conversely, if all the newly added clauses are satisfied, then either no variable of S_p or no variable of S_r is set to *true* and thus either x_p or x_r is set to *true* and C is also satisfied. For the other cases it can be seen similarly.

Hence, an assignment setting exactly the variables $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ to *true*, where $i_1 \leq i_2 \leq \cdots \leq i_k$, satisfies φ if and only if the assignment setting exactly variables $a^l_{i_l}$, $1 \leq l \leq k$ and $b^l_{i_l, i_{l+1}-i_l}$, $1 \leq l \leq k-1$ to *true* is satisfying for ψ. This finishes the reduction, as the instance (ψ, k') can be clearly constructed in polynomial time. □

7.3 Characterization by Computational Models

Although we said that the parameterized hardness classes are not defined by a computational model, there is actually a way to characterize the classes in terms of Turing Machine computation [CF03]. We only show that for two most important classes W[1] and W[2]. For that purpose let us first formally define a Nondeterministic Turing Machine.

Definition 7.8. A *Nondeterministic Turing Machine* is a sextuple $(\Sigma, t, Q, s, A, \delta)$, where Σ is an alphabet, t is the number of tapes, Q is a set of internal states, s is the initial state, $A \subseteq Q$ is the set of accepting states and $\delta \subseteq (Q \times \Sigma^t \times Q \times \Sigma^t \times \{-1, 0, 1\}^t)$ is the set of transitions. A quintuple $(p, (r_1, \ldots, r_t), q, (w_1, \ldots, w_t), (m_1, \ldots, m_t))$ being in δ means that if the machine

is in the state p reading the symbol r_i on a tape i (for every $1 \leq i \leq t$) it can proceed to the state q, writing the symbol w_i on the tape i and moving the head on this tape by m_i (for every i).
Note that not only the set of transitions can contain several possible transitions for one particular state and a t-tuple of symbols read, but it is also not required to be total, that is it does not have to contain a transition for each combination of state and t symbols read. The maximum number of possible transitions from a particular state when particular symbols are read is called the *amount of nondeterminism*.

It was proven in [CI97] that the following natural parameterized analogue of the Halting Problem is W[1]-complete:

Short Nondeterministic Turing Machine Computation
Input: A single-tape nondeterministic Turing machine $M = (\Sigma, 1, Q, s, A, \delta)$; a positive integer k.
Question: Is there a computation of M (on the empty tape) that reaches an accepting state in at most k steps?
Parameter: The allowed length of a computation k.

Note that for the result it is crucial that no bound is given for the size of the alphabet, the number of states, nor to the amount of nondeterminism. The problem is FPT for any of the parameterizations obtained by combining the number of steps k with any of the aspects mentioned above [CI97].

Similarly, the following problem is complete for W[2] as shown in [Ces03]:

Short Multi-Tape Nondeterministic Turing Machine Computation
Input: A nondeterministic Turing machine $M = (\Sigma, t, Q, s, A, \delta)$; a positive integer k.
Question: Is there a computation of M on the empty input that reaches an accepting state in at most k steps?
Parameter: The allowed length of a computation k.

The result heavily depends on having unlimited number of tapes. Parameterized by k and t, the problem becomes equivalent to the previous one, that is W[1]-complete [CI97]. A characterization of other classes by means of short computations of alternating Turing Machines can be found in [CF03].

Although Turing Machine is a standard tool in complexity considerations, RAM (Random Access Machine) model seems to be used more often when talking about algorithms. We introduce the concept of non-determinism into this model in a slightly unnatural way as proposed by Chen et al. [CFG03].

In a nondeterministic random access machine (NRAM) model a single nondeterministic instruction "GUESS" is added to the standard deterministic random access machine (RAM) model. The semantics of this instruction is: *Guess a natural number less than or equal to the number stored in the accumulator and store it in the accumulator.* Acceptance of an input by an NRAM is defined as

usually for nondeterministic machines, that is, the program accepts the particular input if there is a computation on it that ends by an execution of the ACCEPT instruction. The steps of the computation of an NRAM that execute the GUESS instruction are called *nondeterministic steps*.

To stay within the class W[1], the following restrictions should by put on a program for NRAM.

Definition 7.9 (Chen, Flum, and Grohe [CFG03])**.** An NRAM program $\mathbb{P}$ is *tail-nondeterministic k-restricted* if there are computable functions f and g and a polynomial p such that on every run with input $(x, k) \in \Sigma^* \times \Sigma^*$ the program $\mathbb{P}$

- performs at most $f(k) \cdot p(|x|)$ steps;
- uses at most the first $f(k) \cdot p(|x|)$ registers;
- contains numbers $\leq f(k) \cdot p(|x|)$ in any register at any time;

and all nondeterministic steps are among the last $g(k)$ steps of the computation.

The following characterization due to Chen, Flum, and Grohe [CFG03] seems to be easier to apply then developing a reduction to a single-tape Turing Machine:

Theorem 7.10 (Chen, Flum, and Grohe [CFG03])**.** *A parameterized problem $\mathcal{P}$ is in W[1] if and only if there is a tail-nondeterministic k-restricted NRAM program deciding $\mathcal{P}$.*

The model can be further enriched by a nondeterministic instruction FORALL, to obtain alternating RAMs. By restricting the number of times the machine can switch from using the GUESS instruction to the FORALL instruction and vice versa, one can characterize the other classes of the W-hierarchy [CF03].

7.4 Multicolored Problems

In most reductions that directly reduce Weighted t-Normalized Satisfiability it is necessary to somehow represent, that some variable was not set to *true* in the solution considered. This is usually done by representing the gap between two neighboring (in a certain order) variables set to *true*. Such reductions are sometimes called *gap reductions*.

Another approach is to develop a problem in which the objects for a size-k solution are picked from k groups, each object from a different group. Then the case that an object is not picked into a solution is represented by picking another object from the same group. The groups are usually referred to as colors and we

speak about multicolored problems. The result from Section 6.3 suggests, that the multicolored problem is usually no easier than the original one.

The problem most often used for hardness reductions is probably Multicolored Clique. It can be found in older literature under the name Partitioned Clique, but it is nowadays more known under the other name.

Multicolored Clique (MCC)
Input: A graph $G = (V, E)$, positive integer $k \in \mathbb{N}$ and a proper k-coloring $c : V \to \{1, \ldots, k\}$ of G.
Question: Is there a multicolored clique in G, that is, a clique taking exactly one vertex of each color?
Parameter: The number of colors k.

Note that any clique of size k in a graph properly colored by k colors must take exactly one vertex of each color, and there is no clique of size more than k.

The following easy theorem was in fact proved in [Pie03], and recently rediscovered by [FHRV09].

Theorem 7.11. Multicolored Clique *is W[1]-complete.*

Proof. We provide an almost trivial reduction from and to Clique which we have shown to be W[1]-complete in Section 7.2. Omitting the coloring from an instance of MCC one obtain an equivalent instance of Clique, which implies that MCC is in W[1]. To show the W[1]-hardness we let $G = (V, E), k$ be an instance of Clique. Consider an instance of MCC formed by $G' = (V', E')$, where $V' = V \times \{1, \ldots, k\}$ and $E' = \{\{(u, i), (v, j)\} \mid i \neq j \wedge \{u, v\} \in E\}$, the number k, and the coloring $c : V' \to \{1, \ldots, k\}$ assigning to each vertex (v, i) its second coordinate i.

If $\{v_1, \ldots, v_k\}$ is a k-Clique in G, then $\{(v_1, 1), \ldots, (v_k, k)\}$ is a multicolored clique in G'. On the other hand if $\{(v_1, 1), \ldots, (v_k, k)\}$ is a multicolored clique in G' then for every $i \neq j$ the set $\{v_i, v_j\}$ is an edge of G and, thus, $\{v_1, \ldots, v_k\}$. Hence (G, k) is a yes-instance of Clique if and only if (G', k, c) is a yes-instance of MCC, which finishes the proof, as G' can be constructed in polynomial time and the parameter is preserved. □

Note that the constructed instance of MCC has the same number of vertices of each color and also the number of edges with endpoints colored by a particular pair of colors is the same for each pair of colors selected. This property is also often used in the reductions.

We also mention, that in many reduction from MCC, not only there are gadgets for each color, that represent a selection of a vertex of this color, but there are often gadgets for each pair of different colors representing the selection of an edge between the vertices of the particular color. This simplifies the subsequent check, whether the select objects form a clique. We just check, whether the selected edges are incident with the selected vertices, which is often much simpler

than checking whether the selected vertices are adjacent. This idea called *edge representation strategy* was first introduced by Fellows et al. in [FHRV09].

A very simple example of a reduction starting from Multicolored Clique (MCC) is to show that List Coloring is W[1]-hard parameterized by the vertex cover number. This was first observed in [FFL$^+$07] and so far constitutes one of a few, if not the only example of a problem hard with respect to the vertex cover number. We also mention that the paper is also the first one to show that some problem is W[1]-hard with respect to the treewidth for which it also uses a reduction from MCC together with the edge representation strategy.

In List Coloring we are given a graph G, a set of colors B and a mapping $L : V(G) \to \mathcal{P}(B)$ assigning to each vertex its list of available colors. The question is whether there is a proper coloring $c : V(G) \to B$ of G respecting the lists. This means that for every $v \in V(G)$ we have $c(v) \in L(v)$.

Theorem 7.12. List Coloring *is W[1]-hard parameterized by the vertex cover number.*

Proof. For the reduction, assume that we are given an instance of MCC with a graph G having n vertices of each color out of $\{1, \ldots, k\}$. The vertices of color i are denoted $v_{i,1}, \ldots, v_{i,n}$. We construct an instance of List Coloring formed by a graph G', a set of colors $B = V(G)$ and a mapping $L : V(G') \to B$. We start by introducing k vertices $a_1, \ldots, a_k$; the vertex a_i will have a list $L(a_i) = \{v_{i,1}, \ldots, v_{i,n}\}$. The colors chosen for these vertices should represent the selected vertices of particular colors.

To ensure that the selected vertices form a clique, we do the following. For each i and i', where $1 \leq i < i' \leq k$, if for some $1 \leq j, j' \leq n$ the vertices $v_{i,j}$ and $v_{i',j'}$ are not connected by an edge in G, we add a new vertex into G' which will be connected to a_i and $a_{i'}$ and will have a list $\{v_{i,j}, v_{i',j'}\}$. Hence, if the vertex a_i was assigned the color $v_{i,j}$ and the vertex $a_{i'}$ the color $v_{i',j'}$, then there would be no chance to color this new vertex. Otherwise there is always at least one color left.

It is easy to see, that this way $x_1, \ldots, x_k$ is a multicolored clique in G if and only if the partial coloring $f : \{a_1, \ldots a_k\} \to V(G)$, that assigns the color x_i to the vertex a_i for every i, can be extended to a proper list coloring of G' respecting the lists L. As the construction can be clearly done in polynomial time, it remains to note that the set $\{a_1, \ldots a_k\}$ forms a vertex cover of size k for the graph G'. Hence, the parameter of the new instance equals the parameter of the original instance and the reduction is indeed a parameterized one. □

7.5 Connections to the Exponential Time Hypothesis

Although it is hard to believe that there could an $f(k)n^{O(1)}$-time algorithm for Short Nondeterministic Turing Machine Computation (which would be implied by W[1]=FPT) still for some people less familiar with parameterized complexity the following hypothesis is more plausible than that W[1]≠FPT.

Hypothesis 7.13 (Exponential Time Hypothesis (ETH))**.** *There is no algorithm that solves n-variable 3SAT in time $2^{o(n)}$.*

In fact, ETH is a stronger hypothesis — it implies W[1]≠FPT, while it is not known whether W[1]≠FPT implies ETH. In particular Abrahamson et al. [ADF95] have shown the following:

Theorem 7.14 (Abrahamson, Downey, and Fellows [ADF95])**.** *If there is an $f(k) \cdot n^{O(1)}$ time algorithm for k-Clique, then ETH fails.*

This result was later strengthened so that it also excludes algorithms with the exponent of the polynomial running time growing slower than linear in the parameter:

Theorem 7.15 (Chen, Huang, Kanj, and Xia [CHKX04])**.** *If there is an $f(k) \cdot n^{o(k)}$ time algorithm for k-Clique, then ETH fails.*

The result stated for Clique can be easily translated to other problems. Namely, if there was a parameterized reduction transforming an instance (G, k) of k-Clique to an instance (x', k') of a problem $\mathcal{P}$ with $k' = O(k^c)$ then an algorithm for $\mathcal{P}$ with running time $f(k)n^{o(k^{1/c})}$ would imply an $f'(k)n^{o(k)}$ algorithm for k-Clique and, thus, ETH would fail. As most of the parameterized reductions known have either $k' = O(k)$ or $k' = O(k^2)$ this provides a good lower bound for many problems. Note that this way the results also translate to many problems parameterized by a structural or other parameters.

Assuming ETH it is also possible to prove, that for certain problems the dependence of the exponent of the exponential part of the running time on the parameter is asymptotically optimal.

Theorem 7.16 (Cai and Juedes [CJ03])**.** *If* Vertex Cover *can be solved in time $2^{o(k)} \cdot n^{O(1)}$, then ETH fails.*

Similar results can be proved also for problems on planar graphs, but here the known (asymptotically optimal) algorithms are only exponential in the square root of the parameter; see Section 6.7 for such algorithms.

Theorem 7.17 (Cai and Juedes [CJ03])**.** *If* Vertex Cover*,* Independent Set*, or* Dominating Set *can be solved in time $2^{o(\sqrt{k})} \cdot n^{O(1)}$ for planar graphs, then ETH fails.*

The following is a further strengthening of ETH, which is definitely not so widely believed as ETH itself.

Hypothesis 7.18 (Strong Exponential Time Hypothesis (SETH))**.** *Let $\epsilon > 0$. There is no algorithm that solves n-variable SAT in time $(2-\epsilon)^n$.*

Under this assumption, the lower bound for the running times of an algorithm for Dominating Set can be further improved as follows.

Theorem 7.19 (Patrascu and Williams [PW10])**.** *If for some $k \geq 3$ and $\epsilon > 0$, k-Dominating Set can be solved in $O(n^{k-\epsilon})$ time then SETH fails.*

This result can be again translated in a certain way to some W[2]-hard problems. By contrast, an $O(n^{0.793k})$ algorithm for k-Clique is known [NP85]. This suggests, that also some differences in the running times achievable for W[1]-complete and W[2]-complete problems are to be expected.

Part II

Case Studies

Chapter 8

Steiner Problems

8.1 Introduction to Steiner Problems

8.1.1 Definition of Studied Problems

The Steiner problem is a traditional problem of both computational geometry and theoretical computer science. Roughly the task is, given a set of points, to find the cheapest way to connect them. While the statement is simple, the complexity can be seen already on the oldest problem in the family: Given 3 points in a plane find a point that minimizes the sum of Euclidean distances to the given points. This question asked by Fermat in the 17^{th} century was first completely solved at the beginning of the 19^{th} century by Jakob Steiner, who gave the name to all these problems. The generalization to more points was studied and solved by Jarník and Kössler [JK34].

The study of Steiner Tree in graphs goes back to Hakimi [Hak71] (the problem was also independently formulated by Levin [Lev71]), who showed that Clique can be reduced to Steiner Tree (the theory of NP-hardness was not known yet). His work was complemented only a year later by Dreyfus and Wagner [DW72], who showed that the problem can be efficiently solved by their famous algorithm (presented in Section 6.2) if the number of terminals to be connected is small.

In the undirected case, the situation is simple, an optimal solution is always a tree — if a cycle was present, we could remove its heaviest edge, decreasing the weight of the solution. If one moves forward to directed graphs, as proposed already by Hakimi [Hak71], the situation becomes more complicated, as several notions of connectivity are available. We concentrate on the following three NP-complete [Fra92] Steiner problems in directed graphs.

For the Directed Steiner Tree problem (DST), the task is to connect a distinguished root vertex by directed paths to a set of given terminals. For the Strongly Connected Steiner Subgraph problem (SCSS), the task is to connect all terminals among each other, to achieve strong connectivity among

them. Finally, for the DIRECTED STEINER NETWORK problem (DSN), the task is to connect given terminal vertex pairs. Obviously, DST and SCSS are special cases of DSN, whereas they are "incomparable" to each other. Following the standard modeling, we always assume the underlying directed graph to be complete; arcs that do not exist are modeled by assigning them the weight ∞.

Formally, let W be some subset of $\mathbb{N} \cup \{\infty\}$. If V is a set of vertices, $w : V \times V \to W$ is a weight function[1], and $A \subseteq V \times V$ is a set of arcs, then we define $w(A) := \sum_{a \in A} w(a)$. The problems studied in this chapter are formally defined as follows:

DIRECTED STEINER TREE (DST):
Input: A set of vertices V, a weight function $w : V \times V \to W$, a set $T \subseteq V$ of terminals ($l := |T|$), a root $s \in V$, and a weight bound $p \in \mathbb{N}$.
Question: Is there a set of arcs $A \subseteq V \times V$ of weight $w(A) \leq p$ such that in the digraph $D := (V, A)$ for every $t \in T$ there is a directed path from s to t?

STRONGLY CONNECTED STEINER SUBGRAPH (SCSS):
Input: A set of vertices V, a weight function $w : V \times V \to W$, a set $S \subseteq V$ of terminals ($l := |S|$), and a weight bound $p \in \mathbb{N}$.
Question: Is there a set of arcs $A \subseteq V \times V$ of weight $w(A) \leq p$ such that in the digraph $D := (V, A)$ for every $s, t \in S$ there is a directed path from s to t?

DIRECTED STEINER NETWORK (DSN):
Input: A set of vertices V, a weight function $w : V \times V \to W$, l pairs of vertices $(s_1, t_1), (s_2, t_2), \ldots, (s_l, t_l)$, and a weight bound $p \in \mathbb{N}$.
Question: Is there a set of arcs $A \subseteq V \times V$ of weight $w(A) \leq p$ such that in the digraph $D := (V, A)$ for every $1 \leq i \leq l$ there is a directed path from s_i to t_i?

If an undirected graph being a part of a STEINER TREE instance contained an edge $\{x, y\}$ of weight 0, then this edge can be included in any solution to the instance. Therefore having a path to vertex x is the same as having a path to vertex y and vice versa and the edge $\{x, y\}$ can be contracted without affecting the weight of a solution. Thus an undirected graph is usually assumed to have only edges of positive weights. The situation is different for directed graphs. The digraph $(\{u, v, w\}, \{(u, w), (v, w)\})$ does not contain a directed path between u and v in any direction, but if we contract any of its arcs we obtain such a path.

Therefore, to achieve full modeling flexibility (including the cases where one wants to augment an already existing digraph), we sometimes also use arcs of weight 0 to represent already existing connection structure that comes for free. Hence, we distinguish between 0-DST and DST, indicating whether 0-weights are allowed or not (analogously, 0-SCSS, SCSS, 0-DSN, DSN). More precisely, in 0-DST, 0-SCSS, and 0-DSN the set W is a subset of $\mathbb{N}_0 \cup \{\infty\}$.

[1]Observe that in this way we implicitly deal with complete digraphs in the sense that only arc weights are specified.

Allowing only arcs of weights 0 and 1 is studied and known in the literature as *augmentation problem* [ET76], and allowing only arcs of weights 1 and ∞ models the case that one searches for a minimum-size subgraph, for example, including the Unweighted Directed Steiner Tree problem mentioned in [GHK+09]. Moreover, we consider the (maximum) *ratio* r of arc weights to be the quotient of the maximum occurring arc weight and the minimum occurring arc weight, excluding 0-weights from consideration. If there are ∞-weight arcs, then we call this *unbounded ratio*. Clearly, a bounded ratio means that in principle every arc is a candidate for being part of the connecting minimum-cost subgraph. It is important to observe that a higher ratio makes the problem harder (or at least not easier) as well as allowing arcs of weight 0 does. We set $\min_W := \min(W \setminus \{0\})$ and $\max_W := \max W$ (∞ if contained in W). The ratio is then defined as $r := \max_W / \min_W$.

8.1.2 Complexity

Steiner-type problems were among the first to be shown to be NP-complete [GJ79] and this is also the case for the three considered problems in directed graphs [Fra92]. As such, the problems are one of the most intensively studied in terms of the polynomial-time approximability. Unfortunately, in general terms, one may say that the considered problems are hard to approximate. For instance, it is known that 0-DSN cannot be approximated to within a factor of $O(2^{\log^{1-\epsilon} n})$ for any fixed $\epsilon > 0$, unless NP $\subseteq$ TIME$(2^{\text{polylog}(n)})$ [DK99]. Herein, n denotes the number of vertices and m the number of arcs of finite weight. The best known approximation factor is $O(l^{1/2+\epsilon})$ for any fixed $\epsilon > 0$ [CEGS08]. Moreover, 0-DST cannot be approximated to within a factor of $(1-\epsilon)\ln l$ for any fixed $\epsilon > 0$, unless NP $\subseteq$ DTIME$[n^{O(\log\log n)}]$ [Fei98]. The best known approximation factor for 0-DST is $O(l^{\epsilon})$ for any fixed $\epsilon > 0$ [CCC+99]. We refer to [Eve07, Khu95, KN07] for surveys on the numerous polynomial-time approximation results for Steiner-type problems, which we omit here as we are more interested in other types of results.

Much less is known about the parameterized complexity of directed Steiner problems. In the spirit of the multivariate algorithmics approach to computational intractability (Section 4.5), some meaningful parameterizations of the considered Steiner problems are

- the parameter l denoting the number of terminals to be connected;
- the weight p of the solution divided by the minimum arc weight $\min_W$ (again excluding 0), giving the parameter $p/\min_W$;[2] and
- the combined parameter $(l, p/\min_W)$.

[2]This parameter naturally reflects the number of arcs in the spanning subgraph by providing an upper bound on the number of (non-zero) arcs; it is the standard parameterization (see Section 4.1).

Note that (as already mentioned in Section 4.5) a parameterized hardness result with respect to the combined parameter clearly means hardness results for each single parameter and, by way of contrast, a fixed-parameter tractability result for a single parameter trivially extends to the combined parameter.

In the basic Steiner Tree problem in undirected graphs any solution to an instance with l terminals has at least $l-1$ edges. Therefore the question is rather how many additional edges are needed, which is equal to the number of non-terminals in the solution. The problem is known to be W[2]-complete with respect to this parameter [DF98], whereas it is FPT with respect to the number of terminals by the Dreyfus-Wagner Algorithm [DW72]. We have already presented this algorithm in Section 6.2, together with a way how to improve its running time from the original $O(3^l \cdot n + 2^l \cdot n^2 + n^3)$ to $O(3^l \cdot n + 2^l[(l + \log n)n + m])$ as proposed by [DYW+07]. If the weights used are small ($\max_W$ is bounded by a constant), then the running time can be further improved to $O((2^l \cdot n^2 + nm)polylog(n))$ using Möbius and Fourier transformations [BHKK07]. Otherwise for every $\epsilon > 0$ and $1/2 < \xi \leq 1$ running time $O((2+\epsilon)^l \cdot n^{O(((-\log \epsilon)/\epsilon)^{\xi})})$ can be achieved by guessing $1/\epsilon$ new terminals to be added to split the instance to several smaller instances [FKM+07].

All these results also transfer to the directed case in a certain way. In particular, the FPT-algorithm can also be used to solve 0-DST, yielding its fixed-parameter tractability with respect to the number of terminals. Moreover, since the Set Cover problem is W[2]-complete [DF98] and it can also be formulated as a special case of both 0-DST and 0-SCSS [Fra92], it follows that 0-DST, 0-SCSS, and 0-DSN are W[2]-hard with respect to the parameter $p/\min_W$. We sketch in Sections 8.2 and 8.3 reductions from Weighted Monotone 2-Normalized Satisfiability to make the presentation self-contained.

Finally, Feldman and Ruhl [FR06] showed that 0-DSN can be solved in $O(mn^{4l-2} + n^{4l-1} \log n)$ time using their $O(mn^{2l-3} + n^{2l-2} \log n)$-time algorithm for 0-SCSS as a subprocedure. These algorithmic results directly lead to the question whether there are polynomial-time algorithms whose polynomial degree is independent of l. Thus, Feldman and Ruhl explicitly asked for the fixed-parameter tractability of 0-DSN and 0-SCSS with respect to the parameter l.

In this chapter, which is based on [GNS09], we extend the above results by initiating a systematic study of the parameterized complexity of the Steiner problems discussed above (also see Table 8.1). First, we summarize the known results for Steiner Tree and show how to transfer them to the directed case in Section 8.2. We complement this by showing that there is no polynomial kernel even for DST (and 0-DST) with unbounded ratio, even with respect to the combined parameter, unless NP$\subseteq$coNP/poly. In Section 8.3 we focus on SCSS and 0-SCSS. We answer the question of Feldman and Ruhl in a negative way, showing that the problems are W[1]-hard with respect to combined parameter $(l, p/\min_W)$ if the arc weight ratio r is at least 9 (4 for 0-SCSS). On the other hand, we show that 0-SCSS with ratio 1 and SCSS with ratio at most 2 are fixed-

Table 8.1: Parameterized complexity results for DST, 0-DST, SCSS, 0-SCSS, DSN, and 0-DSN. Herein, r denotes the ratio of the arc weights.

Probl.:	Parameter: l	Parameter: $p/\min_W$	Parameter: combined
DST	$r \geq 1$: FPT (Sec. 6.2) $r = \infty$: no poly. kernel (Thm. 8.1)	$r \geq 1$: FPT (Sec. 6.2) $r = \infty$: no poly. kernel (Thm. 8.1)	$r \geq 1$: FPT (Sec. 6.2) $r = \infty$: no poly. kernel (Thm. 8.1)
0-DST	$r \geq 1$: FPT (Sec. 6.2) $r = \infty$: no poly. kernel (Thm. 8.1)	$r \geq 1$:W[2]-h. [DF98]	$r \geq 1$: FPT (Sec. 6.2) $r = \infty$: no poly. kernel (Thm. 8.1)
SCSS	$r \geq 9$: W[1]-h. (Thm. 8.2) $2 < r < 9$: open $r \leq 2$: FPT (Thm. 8.4)	$r \geq 9$: W[1]-h. (Thm. 8.2) $2 < r < 9$: open $r \leq 2$: FPT (Thm. 8.4)	$r \geq 9$: W[1]-h. (Thm. 8.2) $2 < r < 9$: open $r \leq 2$: FPT (Thm. 8.4)
0-SCSS	$r \geq 4$: W[1]-h. (Thm. 8.3) $1 < r < 4$: open $r = 1$: FPT (Thm. 8.5)	$r \geq 1$: W[2]-h. [DF98, Fra92]	$r \geq 4$: W[1]-h. (Thm. 8.3) $1 < r < 4$: open $r = 1$: FPT (Thm. 8.5)
DSN	$r \geq 9$: W[1]-h. (Thm. 8.2) $1 < r < 9$: open $r = 1$: in P (Obser. 8.6)	$r \geq 9$: W[1]-h. (Thm. 8.2) $1 < r < 9$: open $r = 1$: in P (Obser. 8.6)	$r \geq 9$: W[1]-h. (Thm. 8.2) $1 < r < 9$: open $r = 1$: in P (Obser. 8.6)
0-DSN	$r \geq 1$: W[1]-h. (Thm. 8.7)	$r \geq 1$: W[2]-h. [DF98, Fra92]	$r \geq 1$: W[1]-h. (Thm. 8.7)

parameter tractable with respect to the number of terminals l. The last section is devoted to DSN and 0-DSN. The hardness results which extend from the SCSS and 0-SCSS case are further strengthened here even to the augmentation case of 0-DSN with ratio 1 with respect to the combined parameter. We also mention how to use the algorithm devised for the special cases of 0-SCSS and SCSS to improve the running times of the algorithm of Feldman and Ruhl for 0-DSN and DSN in the particular cases. As indicated in Table 8.1, our work leaves several challenges for future research, particularly concerning the parameterized complexity for small arc weight ratios.

8.2 Steiner Trees

First we focus on Directed Steiner Tree (DST) and 0-Directed Steiner Tree (0-DST). The methods from Section 6.2, where we have shown that the undirected Steiner Tree is FPT with respect to the number of terminal l can be also used for the directed case of both DST and 0-DST. It suffices to consider $S(s, X)$ and $D(s, X)$ to be the minimum size of a tree with all arcs oriented from s to the (other) leaves of the tree (with further constrains in case of $D(s, X)$) as also mentioned in [DYW$^+$07]. This somehow follows from that the algorithm uses Dijkstra's Algorithm which also works for oriented graphs, even when some arcs have weight 0. As all the better algorithms mentioned in the introduction are basically still improvements of the same algorithm, they can

be used, too. Note also that as any solution to DST must use at least $l - 1$ arcs, we have $l \leq (p/\min_W) + 1$ whenever a solution is possible. Therefore DST is fixed-parameter tractable for both single parameterizations by l and $p/\min_W$, respectively, while 0-DST is fixed-parameter tractable with respect to l.

We have already mentioned that Steiner Tree is W[2]-hard with respect to the number of non-terminals in the solution [DF98] and that result can be translated to 0-DST with respect to $p/\min_W$ [Fra92]. For completeness we now show here a reduction from Weighted Monotone 2-Normalized Satisfiability, which was shown to be W[2]-complete by Downey and Fellows [DF95a, DF95b] as we have mentioned in Section 7.2, to both problems.

Suppose that we are given an instance of Weighted Monotone 2-Normalized Satisfiability formed by $k \in \mathbb{N}$ and a formula φ with clauses $C_1, \ldots, C_m$ and variables $x_1, \ldots, x_n$. For Steiner Tree we construct a graph in which there is a designated root s and one vertex for each clause and each variable. The vertices corresponding to the clauses are adjacent to vertices corresponding to variables occurring in the particular clause and nothing else and the root is adjacent with all variable vertices. One is asked to connect the root vertex with all clause vertices using at most k non-terminals — variable vertices. The correspondence with weight k satisfying assignments of φ is obvious.

For 0-DST with ratio at least 1 we use the same vertex set and from each variable vertex there is an arc of weight 0 to each clause this variable occurs in. All the other arcs have weight $\min_W$ (recall that the underlying digraph is complete). The task is again to connect the root to all clause vertices by arcs of total weight at most $k \cdot \min_W$. It is easy to see that given an optimal solution one can reroute the arcs of non-zero weight to make them originate from the root and to end in some variable vertices. As there can be at most k arcs of non-zero weight used in the solution it is again easy to see that the out-neighborhood of the root s in such a solution one-to-one correspond to a satisfying assignment of weight k for φ.

We can add to these results a proof for the "non-existence" of a polynomial-size problem kernel for DST parameterized by the combined parameter $(l, p/\min_W)$. Recall that this implies the non-existence of polynomial-size problem kernels for the more general 0-DST case as well as for the single parameter case. To this end, we use the technique of Bodlaender et al. [BDFH09] introduced in Section 5.4.

Theorem 8.1. *There is no polynomial-size problem kernel for* Directed Steiner Tree *with unbounded ratio with respect to the combined parameter* $(l, p/\min_W)$ *unless* NP⊆coNP/poly.

Proof. We show the claim by applying the lower bound technique of Bodlaender et al. [BDFH09] presented in Section 5.4 to DST. More specifically, we show that there exists a composition algorithm for DST, which implies, according to the lower bound technique, that a polynomial problem kernel for DST with respect

to the combined parameter would lead to NP⊆coNP/poly as the studied problem is NP-complete, which clearly translates to its unparameterized variant.

Let $(I_1, l, d), \ldots, (I_r, l, d)$ be a set of DST-instances where, for $1 \leq i \leq r$, I_i consists of a vertex set V_i, a weight function $w_i : V_i \times V_i \to W_i$, a set $T_i \subseteq V_i$ of terminals, a root $s_i \in V_i$, and a weight bound $p_i \in \mathbb{N}$. Moreover, $l = |T_1| = \cdots = |T_r|$ and $d = p_1/\min\{W_1 \setminus \{0\}\} = \cdots = p_r/\min\{W_r \setminus \{0\}\}$. By rescaling, without loss of generality, we can assume that $\min\{W_1 \setminus \{0\}\} = \cdots = \min\{W_r \setminus \{0\}\}$ and, thus, $p_1 = \cdots = p_r$. Now we show how a composition algorithm constructs a DST-instance $(I, l, 2d + 2l)$ which is a yes-instance if and only if there is one i such that (I_i, l, d) is a yes-instance. First, the algorithm adds a set U of $l + 1$ new vertices to $V := \bigcup_{1 \leq i \leq s} V_i$. Herein, one new vertex s is the new root and the remaining l new vertices form the new terminal set $T := \{t_1, \ldots, t_l\}$. The set of weights of the new instance contains all possible weights of the original instances. We set $p := 2p_1 + 2l \cdot \min_W$. Let $T_i := \{t_1^i, \ldots, t_l^i\}$ denote the respective terminal sets of the given instances. The new weight function w is defined as follows:

- $w((u, v)) := w_i((u, v))$ if both u and v are from V_i;
- $w((u, v)) := \infty$ if u and v are from different instances;
- $w((u, s)) := \infty$;
- $w((s, u)) := p_i + l \cdot \min_W$ if $u = s_i$ for some $1 \leq i \leq r$; otherwise, $w((s, u)) := \infty$;
- $w((t_i, u)) := \infty$ for all $1 \leq i \leq l$;
- $w((u, t_i)) := \min_W$ if $u = t_i^j$ for some $1 \leq j \leq r$; otherwise, $w((u, t_i)) := \infty$.

Clearly, the above algorithm runs in polynomial time. To show that $(I, l, 2d + 2l)$ is a yes-instance if and only if one of (I_i, l, d) is a yes-instance, observe that from the new root s we can only use the arcs between s and the old roots s_i. By the weight upper bound p, we know that we cannot afford to use more than one of such arcs. Therefore, connecting s to T can be reduced to connecting one of the old roots to its corresponding terminals. This completes the proof. □

8.3 Strongly Connected Steiner Subgraph

In this section we focus on Strongly Connected Steiner Subgraph (SCSS) and 0-Strongly Connected Steiner Subgraph (0-SCSS). We have said that DST and SCSS are "incomparable" (there is no obvious reduction in either direction), but this is not the case for 0-DST and 0-SCSS as 0-DST can be reduced to 0-SCSS by the following simple reduction.

Given an instance (V, w, T, s, p) of 0-DST one can obtain an equivalent instance (V, w', S, p) of 0-SCSS by setting $S := T \cup \{s\}$, $w'(t, s) = 0$ whenever

$t \in T$ and $w'(x, y) = w(x, y)$ for any other pair of x and y. It is easy to see that there is a path from s to every $t \in T$ in $(V, A \setminus \{(t, s) \mid t \in T\})$ if there is a path between any two vertices of S in (V, A). Conversely, if there is a path from s to every $t \in T$ in (V, A) then in $(V, A \cup \{(t, s) \mid t \in T\})$ obviously there is a path between any two vertices of S. It remains to note that for any $A \subseteq V \times V$ we have $w(A \setminus \{(t, s) \mid t \in T\}) \leq w'(A)$ and $w'(A \cup \{(t, s) \mid t \in T\}) \leq w(A)$ and the equivalence of the instances follows.

As a corollary of this reduction we get that 0-SCSS is W[2]-hard with respect to $p/\min_W$ as so is 0-DST. This can be also deduced from the NP-hardness reductions given by Frank [Fra92].

We further show in Subsection 8.3.1 that both problems are W[1]-hard with respect to the combination of parameters $p/\min_W$ and l even for rather small ratios (at least 9 for SCSS and at least 4 for 0-SCSS). Then we complement the results in Subsection 8.3.2 by showing the fixed-parameter tractability of SCSS with ratio at most 2 with respect to each single parameter and of 0-SCSS of ratio 1 with respect to the number of terminals l.

8.3.1 Hardness with Respect to the Combined Parameter

We start with SCSS. We provide an fpt-reduction from Multicolored Clique (MCC) which we have shown to be W[1]-complete in Section 7.4. Recall that in MCC we are given an undirected graph that is properly k-colored and the question is whether there is a size-k clique in it taking exactly one vertex from each color class. The parameter is k.

Theorem 8.2. Strongly Connected Steiner Subgraph *with arc-weight ratio at least* 9 *is W[1]-hard with respect to the combined parameter* $(l, p/\min_W)$.

Proof. Let an undirected graph $G = (V, E)$, an integer $k \in \mathbb{N}$, and a proper coloring $c : V \to \{1, \ldots, k\}$ form an instance of MCC. In what follows, we construct an instance (V', w, S, p) of SCSS that corresponds to the given instance of MCC in the sense that it is a yes-instance of SCSS if and only if (G, k, c) is a yes-instance of MCC. Since the instance will be constructible in polynomial time and its parameters l and $p/\min_W$ will be bounded by a function of the original parameter k, the construction provides a parameterized reduction between the problems, showing the hardness of SCSS with respect to the combined parameter. The high-level idea of the construction of the corresponding SCSS-instance is as follows:

First, for every fixed arc weight ratio $r \geq 9$, we use only two weights for the arcs between the vertices in V', the cheap arcs having weight $\min_W$ and the expensive arcs having weight $\max_W$ (∞ if the ratio is unbounded) with $r = \max_W/\min_W$. It will be shown that there is always a solution using only cheap arcs. Thus, we consider in the following only such solutions.

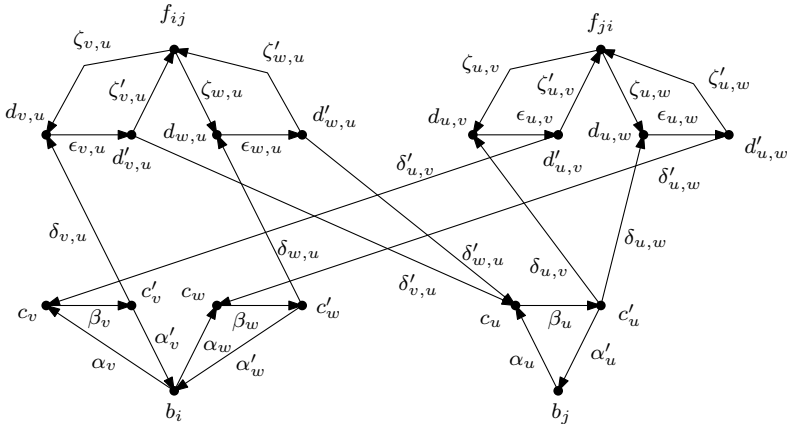


Figure 8.1: Part of the construction from Theorem 8.2 with three vertices—v and w of color i and u of color j—and two edges $\{u, v\}$ and $\{u, w\}$. Only the arcs of $\mathcal{Y} \setminus \Gamma$ are drawn for simplicity. The gadget representing the choice of a vertex of color i is in the bottom left corner, the one for a vertex of color j in the bottom right, the edge selection is represented as a selection of an arc from color-i vertices to color-j vertices in top left and as a selection of an arc from color-j vertices to color-i vertices in top right. The gadgets are interconnected according to the incidence of the vertices and the edges.

Second, for each color i there is one terminal b_i that has only cheap arcs to and from vertex gadgets (which will be described later) representing the vertices in the MCC-graph G of this color. Thus, the paths between b_i and other terminals, which consist of cheap arcs, have to pass through some arcs in some of these vertex gadgets. This corresponds to taking some vertex from this color class into the solution for the MCC-instance. A similar gadget is also used for every pair of distinct colors, representing the choice of the edge connecting the vertices of the appropriate colors (this corresponds to the edge representation strategy as described in Section 7.4).

Third, the vertex gadget for a vertex v of G consists of two vertices c_v and c'_v and a cheap arc (c_v, c'_v). This arc is the only cheap one leaving c_v and is also the only cheap arc entering c'_v. Taking this unambiguously defined arc in solutions for the SCSS-instance represents the selection of the corresponding vertex into the solution for the MCC-instance. The edges of G are encoded in a similar way. Note, however, that every edge is encoded twice. Finally, the vertex and edge gadgets are connected by cheap arcs according to the incidence so that the selected edges are between the selected vertices.

After this informal high-level description of the parameterized reduction from MCC to SCSS, we now come to the mathematical details. We construct our instance (V', w, S, p) of SCSS as follows. The set of vertices V' consists of the following six vertex subsets (see Fig. 8.1):

$$\begin{array}{rclcrcl} B & := & \{b_i \mid 1 \le i \le k\}, & & D & := & \{d_{u,v}, d_{v,u} \mid \{u,v\} \in E\}, \\ C & := & \{c_v \mid v \in V\}, & & D' & := & \{d'_{u,v}, d'_{v,u} \mid \{u,v\} \in E\}, \\ C' & := & \{c'_v \mid v \in V\}, & & F & := & \{f_{ij} \mid 1 \le i,j \le k, i \ne j\}. \end{array}$$

The following arcs are given the weight $\min_W$, that is, they are cheap arcs (see Fig. 8.1):

$$\begin{array}{rcl} \mathcal{A} & := & \{\alpha_v := (b_{c(v)}, c_v) \mid v \in V\}, \\ \mathcal{A}' & := & \{\alpha'_v := (c'_v, b_{c(v)}) \mid v \in V\}, \\ \mathcal{B} & := & \{\beta_v := (c_v, c'_v) \mid v \in V\}, \\ \Gamma & := & \{\gamma_{u,v} := (c'_u, c_v) \mid u,v \in V\}, \\ \mathcal{D} & := & \{\delta_{u,v} := (c'_u, d_{u,v}), \delta_{v,u} := (c'_v, d_{v,u}) \mid \{u,v\} \in E\}, \\ \mathcal{D}' & := & \{\delta'_{u,v} := (d'_{u,v}, c_v), \delta'_{v,u} := (d'_{v,u}, c_u) \mid \{u,v\} \in E\}, \\ \mathcal{H} & := & \{\epsilon_{u,v} := (d_{u,v}, d'_{u,v}), \epsilon_{v,u} := (d_{v,u}, d'_{v,u}) \mid \{u,v\} \in E\}, \\ \mathcal{Z} & := & \{\zeta_{u,v} := (f_{c(u),c(v)}, d_{u,v}), \zeta_{v,u} := (f_{c(v),c(u)}, d_{v,u}) \mid \{u,v\} \in E\}, \\ \mathcal{Z}' & := & \{\zeta'_{u,v} := (d'_{u,v}, f_{c(u),c(v)}), \zeta'_{v,u} := (d'_{v,u}, f_{c(v),c(u)}) \mid \{u,v\} \in E\}, \\ \mathcal{Y} & := & \mathcal{A} \cup \mathcal{A}' \cup \mathcal{B} \cup \Gamma \cup \mathcal{D} \cup \mathcal{D}' \cup \mathcal{H} \cup \mathcal{Z} \cup \mathcal{Z}'. \end{array}$$

All remaining arcs are set to be expensive arcs, that is, for $(x,y) \in ((V \times V) \setminus \mathcal{Y})$, $w((x,y)) := \max_W$ (∞ if the ratio is unbounded). The terminal set S is $B \cup F$ and hence $l = |S| = |B| + |F| = k + k(k-1) = k^2$. Finally, set $p := (3k + 5k(k-1)) \cdot \min_W$. It is clear that the instance is constructible in polynomial time and that both l and $p/\min_W$ only depend on the parameter k. Next, we show that every size-k clique of the MCC-instance one-to-one corresponds to a weight-p solution of the SCSS-instance.

"$\Rightarrow$": If K is a multi-colored clique in G, then one obtains a set A of arcs that form a solution to the SCSS-instance as follows:

$$A := \{\alpha_v, \alpha'_v, \beta_v \mid v \in K\} \cup \{\delta_{u,v}, \delta'_{u,v}, \epsilon_{u,v}, \zeta_{u,v}, \zeta'_{u,v} \mid u,v \in K, u \ne v\}.$$

To show that the vertices of S are mutually connected in the digraph (V', A), assume that $v_h \in K$ is the vertex of color h. Now, for every two terminals b_i and b_j, the arcs α_{v_i}, β_{v_i}, δ_{v_i,v_j}, ϵ_{v_i,v_j}, δ'_{v_i,v_j}, β_{v_j}, α'_{v_j} form a path from b_i to b_j. The arcs α_{v_i}, β_{v_i}, δ_{v_i,v_j}, ϵ_{v_i,v_j}, ζ'_{v_i,v_j} form a path from b_i to $f_{i,j}$ and the arcs ζ_{v_i,v_j}, ϵ_{v_i,v_j}, δ'_{v_i,v_j}, β_{v_j}, α'_{v_j} form a path from $f_{i,j}$ to b_j. Hence, the set S is strongly connected and with an easy calculation one shows $w(A) \le p$.

"$\Leftarrow$": To show that a weight-p solution of the SCSS-instance implies a size-k clique of the MCC-instance, we need the following claim:

Claim. For every set of arcs $A \subseteq V' \times V'$, there is a set of arcs $A' \subseteq \mathcal{Y}$ with $w(A') \le w(A)$ such that, for every two vertices $x, y \in V'$, if there is a path from x to y in (V', A), then there is a path from x to y in (V', A').

Proof of Claim. We construct a set A' by replacing each arc from $A \setminus \mathcal{Y}$ by a set of arcs in $\mathcal{Y}$ and show that, for every $(x, y) \in A$, there is a path from x to y in (V, A'). Since the weight of the replaced arc is $\max_W$ whereas the weight of each replacement arc is $\min_W$, if we introduce at most r replacement arcs for each replaced arc, then $w(A') \leq w(A)$ and the claim follows.

To define the set A'_e of the replacement arcs for a replaced arc $e = (x, y) \in A \setminus \mathcal{Y}$, we distinguish several cases depending on whether the endpoints represent a color, a pair of colors, a vertex, or an edge. Moreover, in order to decrease the number of cases in the analysis, we build A'_e in three steps, first the replacement arcs forming a path from x to some vertex $x' \in C'$, then the replacement arcs forming a path from some vertex $y' \in C$ to y, and, finally, the replacement arc connecting x' to y'. In the first step, we consider the following cases for x:

- If $x = b_i$ for some $1 \leq i \leq k$, then consider an arbitrary vertex v of color $c(v) = i$, add the arcs α_v and β_v to A'_e, and set $x' := c'_v$.
- If $x = c_v$ for some $v \in V$, then add the arc β_v to A'_e and set $x' := c'_v$.
- If $x = c'_v$ for some $v \in V$, then add no arc to A'_e and set $x' := x$.
- If $x = d_{u,v}$ for some $u, v \in V$, then add the arcs $\epsilon_{u,v}, \delta'_{u,v}, \beta_v$ to A'_e and set $x' := c'_v$.
- If $x = d'_{u,v}$ for some $u, v \in V$, then add the arcs $\delta'_{u,v}, \beta_v$ to A'_e and set $x' := c'_v$.
- If $x = f_{i,j}$ for some $1 \leq i, j \leq k$, then consider an arbitrary edge $\{u, v\} \in E$ such that $c(u) = i$ and $c(v) = j$, add arcs $\zeta_{u,v}, \epsilon_{u,v}, \delta'_{u,v}, \beta_v$ to A'_e and set $x' := c'_v$.

In the second step, we consider y:

- If $y = b_i$ for some $1 \leq i \leq k$, then consider an arbitrary vertex v with $c(v) = i$, add the arcs β_v and α'_v to A'_e, and set $y' := c_v$.
- If $y = c_v$ for some $v \in V$, then add no arc to A'_e and set $y' := y$.
- If $y = c'_v$ for some $v \in V$, then add the arc β_v to A'_e and set $y' := c_v$.
- If $y = d_{u,v}$ for some $u, v \in V$, then add the arcs β_u and $\delta_{u,v}$ to A'_e and set $y' := c_u$.
- If $y = d'_{u,v}$ for some $u, v \in V$, then add arcs $\beta_u, \delta_{u,v}, \epsilon_{u,v}$ to A'_e and set $y' := c_u$.
- If $y = f_{i,j}$ for some $1 \leq i, j \leq k$, then consider an arbitrary edge $\{u, v\} \in E$ such that $c(u) = i$ and $c(v) = j$, add arcs $\beta_u, \delta_{u,v}, \epsilon_{u,v}, \zeta'_{u,v}$ to A'_e and set $y' := c_u$.

In the third step, we add the arc (x', y') to A'_e. This arc exists since $x' \in C'$, $y' \in C$, and Γ has an arc from u to v for every pair of vertices u and v with $u \in C'$ and $v \in C$. Altogether, we have added at most four arcs in the first step, at most four in the second step, and the arc (x', y') in the third step to A'_e. That is, we add at most nine replacement arcs for each arc from $A \setminus \mathcal{Y}$. Moreover, it is not hard to check that the replacement arcs form a path from x to y. This finishes the proof of the claim.

Given a solution A to the SCSS-instance, we can assume, due to the above claim, that $A \subseteq \mathcal{Y}$. We take an arbitrary vertex $f_{i,j} \in F$. All the paths to and from $f_{i,j}$ in $\mathcal{Y}$ pass through some vertex in $D \cup D'$. In order to connect $f_{i,j}$ to other terminals, A must contain at least one arc from each of the sets $\mathcal{Z}, \mathcal{Z}', \mathcal{H}, \mathcal{D}, \mathcal{D}'$; each of them having indices u and v such that $c(u) = i$ and $c(v) = j$. This means that A contains a disjoint union of sets of at least five arcs, each of the sets one-to-one corresponding to an $f_{i,j}$. Now take an arbitrary vertex $b_i \in B$. Clearly, A must contain two arcs α_u and α'_u with $c(u) = i$ which connect b_i to all other terminals. Moreover, since the vertices in C are sinks and the vertices in C' are sources in the digraph induced by $\mathcal{Y} \setminus \mathcal{B}$, there is at least one β_u with $c(u) = i$ in A. This means that A also contains a disjoint union of sets of at least three arcs, each of these sets one-to-one corresponding to a b_i. Since the above mentioned arcs together already give weight $(5 \cdot |F| + 3 \cdot |B|) \cdot \min_W = p$, there is no other arc in A. Let $K := \{v \in V \mid \beta_v \in A\}$. We show that K is a multi-colored clique in G.

For each color i there is exactly one vertex of color i in K, since there is exactly one β_u in A among those with $c(u) = i$, as we have shown above. It remains to show that, for two arbitrary vertices u_0 and v_0 in K, there is an edge between u_0 and v_0 in E. As we have shown, there are exactly one $\delta_{u,v}$, exactly one $\delta'_{u,v}$, exactly one $\epsilon_{u,v}$, exactly one $\zeta_{u,v}$, and exactly one $\zeta'_{u,v}$ in A with $c(u) = c(u_0)$, $c(v) = c(v_0)$, and $\{u, v\} \in E$. It is not hard to see that the indices u, v must be the same for these five arcs. Now, if $u \neq u_0$, then β_u is not in A and c'_u is a source in $D = (V', A)$. Thus, there is a path to $f_{c(u_0),c(v_0)}$ only from c'_u, $d_{u,v}$, and $d'_{u,v}$ but from no vertex in S. This is a contradiction, since A is a solution. Hence, $u = u_0$. Similarly, $v = v_0$ and thus $\{u_0, v_0\}$ is an edge of G. □

A similar reduction as above also works for 0-SCSS; however, we can prove hardness already for a smaller arc weight ratio.

Theorem 8.3. 0-Strongly Connected Steiner Subgraph *with ratio at least 4 is W[1]-hard with respect to the combined parameter* $(l, p/\min_W)$.

Proof. Giving a parameterized reduction from MCC to 0-SCSS, we use the same construction as for the proof of Theorem 8.2 and only replace the weight function w by a function w' that is defined as follows: For any $x, y \in V'$, we set $w'((x, y)) := \min_W$ if $(x, y) \in \mathcal{B} \cup \mathcal{H}$, $w'((x, y)) := 0$ if $(x, y) \in \mathcal{Y} \setminus (\mathcal{B} \cup \mathcal{H})$,

and $w'((x,y)) := \max_W$ otherwise. We set $p := k^2 \cdot \min_W$. Again, $S := B \cup F$ and $l = k^2$.

It is not hard to check that if K is a multi-colored clique in G, then

$$A := \{\alpha_v, \alpha'_v, \beta_v \mid v \in K\} \cup \{\delta_{u,v}, \delta'_{u,v}, \epsilon_{u,v}, \zeta_{u,v}, \zeta'_{u,v} \mid u, v \in K, u \neq v\}$$

is again a solution to the instance (V', w', S, p) of 0-SCSS. It is also not hard to check that an analogue of the claim shown in the proof of Theorem 8.2 holds in this case. It suffices to take the same replacement and then check that the replacement arcs have weight at most $4 \cdot \min_W$, whereas the replaced arc has weight $\max_W$ which is at least $4 \cdot \min_W$ since the ratio is at least 4.

It remains to show that if there is a solution A such that $A \subseteq \mathcal{Y}$, then there is a multi-colored clique in G. To this end, first observe that in $(V', \mathcal{Y} \setminus (\mathcal{B} \cup \mathcal{H}))$ the vertices in C and D are sinks, while the vertices in C' and D' are sources. Thus, to connect the vertex b_i to the other terminals, there must be at least one β_v in A with $c(v) = i$ and, similarly, to connect $f_{i,j}$ to the other terminals it requires at least one $\epsilon_{u,v}$ in A with $c(u) = i$ and $c(v) = j$. Summing these up, we know that there is exactly one such arc in each of these subsets of $\mathcal{Y}$ and, by similar reasons as in the proof of Theorem 8.2, the set $K := \{v \mid \beta_v \in A\}$ forms a multi-colored clique in G. □

8.3.2 Tractability for Small Ratios

Here, we present two fixed-parameter algorithms for two variants of SCSS and 0-SCSS that restrict the allowed arc weight ratio, respectively. Recall that in Theorems 8.2 and 8.3 we have shown W[1]-hardness of SCSS for arc weight ratio $r \geq 9$ and of 0-SCSS for $r \geq 4$. Now, we complement this with fixed-parameter tractability results for $r \leq 2$ and $r = 1$, respectively, leaving a small gap of unsettled cases.

Theorem 8.4. Strongly Connected Steiner Subgraph *with ratio at most* 2 *is solvable in* $O(2^l \cdot l^2 + n^2)$ *or* $O(2^{(p/\min_W)} \cdot (p/\min_W)^2 + n^2)$ *time.*

Proof. We only consider the case that $p/(2\min_W) < l \leq p/\min_W$. Hamiltonian cycle over terminals gives a total weight at most $2l \cdot \min_W$. This means that to strongly connect the terminals in S we need an arc set with a minimum weight at least $l \cdot \min_W$ and a maximum weight at most $l \cdot \max_W \leq 2l \cdot \min_W$. Thus, $p \geq 2l \cdot \min_W$ always gives yes-instances, while $p < l \cdot \min_W$ gives no-instances. Therefore, the parameters l and $p/\min_W$ are linearly related to each other and it suffices to show that the problem is solvable in $O(2^l \cdot l^2 + n^2)$ time. To this end, we claim that there is always a Hamiltonian cycle on S having the minimum total weight among all arc sets strongly connecting S. Since a minimum-weight Hamiltonian cycle on S can be found in $O(2^l \cdot l^2)$ time [HK62], the theorem follows.

Claim. Among all arc sets strongly connecting the terminals in S, there is always one arc set A with a minimum total weight such that A induces a Hamiltonian cycle on S.

Proof of Claim. Let A denote one arc set strongly connecting S with a minimum total weight. If A induces a Hamiltonian cycle on S, then we are done; otherwise, we construct a new arc set A' from A such that A' induces a Hamiltonian cycle on S and $w(A') \leq w(A)$.

Consider the digraph D' resulting from removing isolated vertices from (V, A). Since all terminals in S are strongly connected in D', there exist for each terminal at least one incoming arc and at least one outgoing arc. For each terminal, we arbitrarily add one of its incoming arcs and one of its outgoing arcs to two initially empty sets I and O, respectively. We set $X := I \cap O$. Note that $X \subseteq (S \times S)$.

Now consider the digraph D'' induced by the arc set X. It consists of vertex-disjoint cycles and paths which only pass through terminals, since I (or O) contains exactly one incoming (or outgoing) arc for each terminal. With $A \nsubseteq (S \times S)$, D'' cannot be a single cycle over all terminals. Now, repeat the following procedure until all cycles in D'', which consists solely of terminals, are destroyed. Let C be such a cycle in D'' and let A'' denote the arc set of D''. There must be at least two arcs e and e' in $A \setminus A''$, e leaving C at a vertex u and e' entering C at a vertex v. Then, delete the outgoing arc of u in D'' and add $e = (u, w)$ to D''. If w is a terminal as well, then remove the arc in A'' that ends in w. Observe that after the procedure D'' may contain non-terminal vertices and all terminals induce a vertex-disjoint union of paths in D''.

Finally, connect these paths directly using arcs between their endpoints to construct a Hamiltonian cycle of terminals. Let A' be the arc set of this Hamiltonian cycle. It remains to show $w(A') \leq w(A)$. Note that each of these paths induced by the terminals in D'' has at least one outgoing and one incoming arc in A attached to the endpoints of this path, respectively. Summing up over all these paths, these arcs have a total weight at least $2n_p \cdot \min_W$ with n_p denoting the number of paths. To connect the paths into a Hamiltonian cycle, we need at most n_p arcs; thus, $w(A') \leq w(A)$. □

Next, we consider the "augmentation case" of 0-SCSS, namely, the case that there are only two weights and one of them is zero. We achieve fixed-parameter tractability with respect to the number of terminals l.

Theorem 8.5. 0-Strongly Connected Steiner Subgraph *with ratio* 1 *is solvable in* $O(4^{l^2} + n^3)$ *time.*

Proof. First note that in this case we have only two weights 0 and $\min_W = \max_W$. Suppose that we are given an input instance of 0-SCSS consisting of V, w, S (where $|S| = l$), and p. If $p/\min_W \geq l$, then the answer is always yes, since we can connect all terminals to a cycle that costs at most $l \cdot \min_W$. So, for the rest of the proof we will assume that $p/\min_W < l$.

We provide four polynomial-time executable data reduction rules that lead to a problem kernel with at most $2 \cdot 2^l + l$ vertices. Let $A_0 := \{a \in V \times V \mid w(a) = 0\}$. To simplify the presentation, the rules are described as modifications of the digraph $H := (V, A_0)$. The vertices of $V \setminus S$ are called *non-terminals*. In the following, we use $N_A^+(v)$ and $N_A^-(v)$ to denote the sets of vertices which have arcs in a set A directed from and to v, respectively. If clear from the context, then we omit the index A. The rules are ordered and the next rule is always applied after the previous one cannot be applied any more. Later rules never produce a situation where an earlier rule could again be applied. The correctness of the rules follows from the proven fact that an instance produced by a rule is a yes-instance if and only if the original instance is a yes-instance.

Rule 1. *Contract strongly connected components into a single vertex.*

Since the arcs in A_0 can be added to any solution, Rule 1 is clearly correct. Moreover it can be exhaustively applied in $O(n^2)$ time.

Rule 2. *For any non-terminal $v \in V \setminus S$ with both $N^-(v) \neq \emptyset$ and $N^+(v) \neq \emptyset$, delete v and connect its neighbors appropriately, that is, continue with the digraph $H' := (V \setminus \{v\}, A_0 \setminus (\{v\} \times N^+(v)) \cup (N^-(v) \times \{v\})) \cup (N^-(v) \times N^+(v)))$.*

After this rule is exhaustively applied, there remain only sources, terminals, and sinks in the digraph and the connections between them are preserved. Hence, the resulting digraph does not depend on the order in which the vertices are considered. To see the correctness of the rule, it is enough to realize that any arc $a \notin A_0$ of the solution starting in v can be replaced by an arc starting in some sink reachable from v in H. Similarly, any arc $a \notin A_0$ ending in v can be replaced by an arc ending in some source, from which v can be reached in H. This rule can be exhaustively applied in $O(n^3)$ time, since there are n vertices and to apply the rule to one vertex takes $O(n^2)$ time.

Rule 3. *Delete all weight-0 arcs between two non-terminals.*

The rule can be applied in $O(n^2)$ time. The following claim shows its correctness.

Claim. If the instance is reduced with respect to Rule 1 and Rule 2, then there is an optimal solution that uses no arc of weight 0 between two non-terminals.

Proof of Claim. Suppose, on the contrary, that each optimal solution uses some arc in $A_0 \cap ((V \setminus S) \times (V \setminus S))$. Let A be an optimal solution with the minimum number of such arcs and let $a := (x, y) \in A \cap A_0$ be such an arc, that is, $x, y \in V \setminus S$. Clearly, x is a source and y is a sink in $H = (V, A_0)$. There is some arc in A ending in x and some arc in A starting in y since $A \setminus \{a\}$ is not a solution. We can assume that there is only one arc ending in x in A for the following reason: If $|N_A^-(x)| \geq 2$, then select an arbitrary $x' \in N_A^-(x)$ and replace the arcs ending in x (except (x', x)) by arcs ending in x' and get another optimal solution that satisfies this assumption. Let us call this unique arc (x', x). Similarly, we assume that there is a unique arc starting in y and we call it (y, y').

Let V_0 denote the minimal set $S \subseteq V_0 \subseteq V$ such that $A \subseteq V_0 \times V_0$ and also assume that A is minimal in the sense that (V_0, A) is a strongly connected

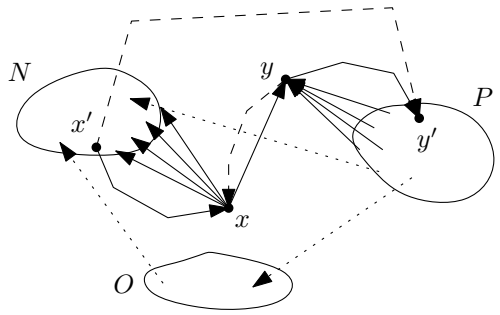


Figure 8.2: Illustration to the proof of the claim in Theorem 8.5. Solid lines represent the sure connections in A. No other connections are possible in A, except for those drawn by dotted lines. Dashed lines represent the arcs in $A' \setminus A$.

digraph. Now distinguish the following sets of vertices (see Fig. 8.2):

$$\begin{aligned} P &:= \{v \in V_0 \setminus \{y\} \mid \exists \text{ a path in } (V_0, A \setminus \{a\}) \text{ from } v \text{ to } y\}, \\ N &:= \{v \in V_0 \setminus \{x\} \mid \exists \text{ a path in } (V_0, A \setminus \{a\}) \text{ from } x \text{ to } v\}, \\ O &:= V_0 \setminus (P \cup N \cup \{x, y\}). \end{aligned}$$

Observe that in $A \setminus \{a\}$ there is no path from any vertex in N to any vertex in P (in particular, $N \cap P = \emptyset$), since otherwise there would be a path from x to y different from (x, y) and, thus, $A \setminus \{a\}$ would be a solution. There is also no path from O to P and from N to O according to the definition of N, P, and O. If N is empty, then $A' := (A \setminus \{(x', x), a\}) \cup \{(x', y)\}$ is a better solution, since $(V_0 \setminus \{x\}, A')$ is strongly connected, x is a non-terminal, and $w((x', x)) \geq w((x', y))$. Hence N is non-empty. Similarly, P is non-empty since otherwise $(A \setminus \{a, (y, y')\}) \cup \{(x, y')\}$ would be a better solution.

Since N is non-empty and (V_0, A) is strongly connected there must be some arc from some vertex in N to some vertex outside N. But, as we have shown, it can end neither in O nor in P nor in y. Hence it ends in x and thus $x' \in N$. Similarly, $y' \in P$. Now let $A' := (A \setminus \{a, (x', x), (y, y')\}) \cup \{(x', y'), (y, x)\}$. To check that $H' := (V_0, A')$ is again strongly connected, observe that there is a path from x to y in H' formed by a path from x to x' ($x' \in N$), the arc (x', y'), and a path from y' to y ($y' \in P$). The path from x' to x is formed by the arc (x', y'), a path from y' to y, and the arc (y, x), and finally the path from y to y' is formed by the arc (y, x), a path from x to x', and the arc (x', y'). Since $w((x', x)) + w((y, y')) = 2\min_W \geq w((x', y')) + w((y, x))$, we have $w(A') \leq w(A)$. Thus A' is an optimal solution which uses fewer arcs of weight 0 between two non-terminals—a contradiction.

Rule 4. *If there are several non-terminals with the same neighborhood, then delete all of them except for one.*

Rule 4 can be exhaustively applied in $O(n^3)$ time by comparing the neighborhoods of each of the $O(n^2)$ pairs of vertices in $O(n)$ time, and if they are the same

deleting one of them again in $O(n)$ time. To see the correctness, observe that we can reconnect the solution arcs with non-zero weight incident with deleted vertices to make them incident with the appropriate remaining vertex without affecting any connection between terminals.

Claim. If Rule 4 cannot be applied, then the digraph has at most $2 \cdot 2^l + l$ vertices.

Proof of Claim. The neighborhood of a non-terminal is formed only by terminal vertices. Moreover a non-terminal is always either a source or a sink. Thus, there are at most $2 \cdot 2^l$ different neighborhoods and thus by Rule 4 at most $2 \cdot 2^l$ non-terminals. Together with l terminals this gives the claimed bound on the number of the vertices.

By the above claim, we have at most $2 \cdot 2^l$ non-terminals in the reduced instance. To solve 0-SCSS on the reduced instance, try all possibilities to connect at most $p/\min_W$ sinks out of 2^l many to at most $p/\min_W$ sources out of 2^l many and check whether in the resulting digraph the terminals are mutually interconnected. This can be carried out in $O(\binom{2^l}{l} \cdot \binom{2^l}{l} \cdot l! \cdot 2^l \cdot l) = O(4^{l^2})$ time. Thus, 0-SCSS with ratio 1 can be solved in $O(4^{l^2} + n^3)$ time. □

8.4 Directed Steiner Network

Finally we look at Directed Steiner Network (DSN) and 0-Directed Steiner Network (0-DSN), the most general of the problems. As such, the hardness results given for SCSS and 0-SCSS in Theorems 8.2 and 8.3 also apply for DSN and 0-DSN. Nevertheless, recall that the problems are known to be solvable in $O(mn^{4l-2} + n^{4l-1} \log n)$ and therefore are in XP with respect to the number of terminals l, as shown by Feldman and Ruhl [FR06].

In Subsection 8.4.1 we show that in contrast to the fixed-parameter tractable case of 0-SCSS, 0-DSN is W[1]-hard even in the augmentation case. We indicate in Subsection 8.4.2 that the algorithms designed in proofs of Theorems 8.4 and 8.5 directly imply a significant running time improvement of the algorithm by Feldman and Ruhl [FR06] for these relevant cases.

Quite surprisingly even the case of DSN with ratio 1, that is the case where all arcs have the same weight is not completely clear at first sight. Nevertheless the following observation shows that this case is indeed in P.

Observation 8.6. Directed Steiner Network *with ratio 1 is in P.*

Proof. First observe that we can limit ourselves to arcs between terminals. If a non-terminal was a part of an optimal solution, then we could reroute the arcs incident with it to some terminal without increasing the weight of the solution. Let $V, (s_1, t_1), \ldots, (s_l, t_l)$ and p be an instance of DSN with ratio 1 (weight function is not necessary in this case) and consider the digraph $D = (V_t, A_t)$, where $V_t := \bigcup_{i=1}^{l} \{s_i, t_i\}$ is the set of terminals and $A_t := \{(s_1, t_1), \ldots, (s_l, t_l)\}$.

It is easy to see that each (weakly) connected component can be treated separately. If such a component C is acyclic (does not contain a directed cycle) then C can be linearly ordered in polynomial time by standard techniques so that each arc ends higher in the order than it started. Now connect each vertex except for the maximal one with its least successor in the order. It is easy to see that this way we obtain a solution of weight $(|C| - 1) \cdot \min_W$. As any solution must use at least $|C| - 1$ arcs to make this component connected, the solution is optimal.

If the considered component C contains a cycle, then at least $|C|$ arcs are needed and any cycle over the vertices of C constitutes an optimal solution. □

8.4.1 Hardness for the Augmentation Case

The W[1]-hardness result shown in Theorem 8.3 for 0-SCSS with ratio at least 4 clearly extends to the more general 0-DSN problem. In what follows, however, we strengthen the result by showing that, in case of 0-DSN, the W[1]-hardness already holds for arc weight ratio $r = 1$. The reduction for 0-DSN with ratio 1 is again from Multicolored Clique. However, with now only one value of non-zero weight allowed, a completely different construction is needed.

Theorem 8.7. 0-Directed Steiner Network *with ratio at least* 1 *is W[1]-hard with respect to the combined parameter* $(l, p/\min_W)$.

Proof. We reduce from Multicolored Clique. For each edge and each vertex color there is a pair of vertices that can be connected either directly or by one of several paths formed by two arcs each, exactly one of them of weight zero. The middle vertex of the chosen path represents the choice of the appropriate edge/vertex. Formally, let $G = (V, E)$, k, and $c : V \to \{1, \ldots, k\}$ be an instance of Multicolored Clique. We construct our instance of 0-DSN as follows. The set of vertices V' consists of the following six vertex subsets (see Fig. 8.3):

$$
\begin{aligned}
&B := \{b_i \mid 1 \le i \le k\},\\
&B' := \{b'_i \mid 1 \le i \le k\},\\
&F := \{f_{ij} \mid 1 \le i < j \le k\},\\
&F' := \{f'_{ij} \mid 1 \le i < j \le k\},\\
&C := \textstyle\bigcup_{i=1}^{k} C_i, \text{ where}\\
&C_i := \{c_v \mid v \in V, c(v) = i\},\\
&D := \textstyle\bigcup_{1 \le i < j \le k} D_{ij}, \text{ where}\\
&D_{ij} := \{d_{u,v} \mid \{u, v\} \in E, c(u) = i, c(v) = j\}.
\end{aligned}
$$

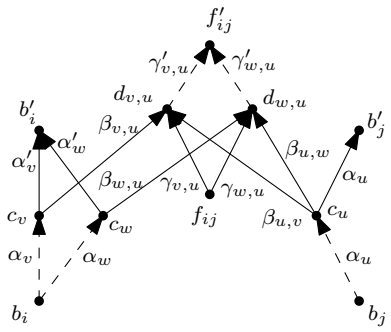


Figure 8.3: Part of the construction from Theorem 8.7 with three vertices—v and w of color i and u of color j—and two edges $\{u, v\}$ and $\{u, w\}$. The 0-weight arcs of $\mathcal{A}' \cup \mathcal{B} \cup \mathcal{C}$ are drawn as solid, the arcs of $\mathcal{A} \cup \mathcal{C}'$ with weight $\max_W$ as dashed lines.

The following arc sets are important for the construction (see Fig. 8.3):

$$\begin{aligned}
\mathcal{A} &:= \{\alpha_v := (b_{c(v)}, c_v) \mid v \in V\},\\
\mathcal{A}' &:= \{\alpha'_v := (c_v, b'_{c(v)}) \mid v \in V\},\\
\mathcal{B} &:= \{\beta_{u,v} := (c_u, d_{u,v}), \beta_{v,u} := (c_v, d_{u,v}) \mid \{u, v\} \in E, c(u) < c(v)\},\\
\mathcal{C} &:= \{\gamma_{u,v} := (f_{c(u),c(v)}, d_{u,v}) \mid \{u, v\} \in E, c(u) < c(v)\},\\
\mathcal{C}' &:= \{\gamma'_{u,v} := (d_{u,v}, f'_{c(u),c(v)}) \mid \{u, v\} \in E, c(u) < c(v)\}.
\end{aligned}$$

Now let $w((x, y)) := 0$ if $(x, y) \in \mathcal{A}' \cup \mathcal{B} \cup \mathcal{C}$, and $w((x, y)) := \max_W = \min_W$ otherwise. The solution is required to connect for every i the vertex b_i to vertex b'_i and for every $1 \leq i < j \leq k$ the vertices b_i, b_j and $f_{i,j}$ to $f'_{i,j}$. Hence $l = k + (3/2) \cdot k(k-1)$. Let the bound on the weight of arcs which are in the solution be $p = (k + (1/2) \cdot k(k-1)) \cdot \min_W$.

It is clear that the instance is constructible in polynomial time and that both l and $p/\min_W$ only depend on the parameter k. It remains to show that there is a multi-colored clique in G if and only if there is a weight-p solution to the constructed 0-DSN instance.

If K is a multi-colored clique in G, then we get a set A of arcs that form a solution to the constructed 0-DSN instance by setting

$$A := \{\alpha_v, \alpha'_v \mid v \in K\} \cup \{\beta_{u,v}, \beta_{v,u}, \gamma_{u,v}, \gamma'_{u,v} \mid u, v \in K, c(u) < c(v)\}.$$

Indeed, if we assume that $v_i \in K$ is the vertex of color i, then for each $i < j$ the arcs α_{v_i} and α'_{v_i} form a directed path from b_i to b'_i whereas the arcs $\alpha_{v_i}, \beta_{v_i,v_j}, \gamma'_{v_i,v_j}$, the arcs $\alpha_{v_j}, \beta_{v_j,v_i}, \gamma'_{v_i,v_j}$, and the arcs $\gamma_{v_i,v_j}, \gamma'_{v_i,v_j}$ form the directed paths to $f'_{i,j}$ from b_i, b_j, and $f_{i,j}$, respectively.

For the reverse direction, we make use of the following claim:
Claim. Every solution $A \subseteq V' \times V'$ to the constructed 0-DSN instance uses only arcs in $\mathcal{A} \cup \mathcal{A}' \cup \mathcal{B} \cup \mathcal{C} \cup \mathcal{C}'$. Moreover, for each vertex in $B \cup F'$, there is exactly one arc in A incident with it.

Proof of Claim. Let us concentrate on the arcs in $\mathcal{X} := (V' \times V') \setminus (\mathcal{A}' \cup \mathcal{B} \cup \mathcal{C})$, that is, the arcs of non-zero weight. The solution A contains at most $p/\min_W$ of them. Each vertex in B has out-degree at least one with respect to $A \cap \mathcal{X}$ since it is connected to a vertex from B' in (V', A). For each $1 \leq i < j \leq k$, at least one of the vertices in $D_{ij} \cup \{f_{ij}\}$ has out-degree at least one in $A \cap \mathcal{X}$ since the vertex f_{ij} is connected to f'_{ij}. But this already gives $k + (1/2)k(k-1) = p/\min_W$ arcs. Hence no vertex in $C \cup B' \cup F'$ has non-zero out-degree in $X \cap A$, the vertices in B have out-degree exactly one, and for each $1 \leq i < j \leq k$ exactly one vertex from $D_{ij} \cup \{f_{ij}\}$ has out-degree exactly one. By similar reasons the vertices in F' have in-degree exactly one, for each $1 \leq i \leq k$ there is exactly one vertex in $C_i \cup \{b'_i\}$ with in-degree one, and all the other vertices have in-degree zero.

Now suppose that for some i the solution contains arc (b_i, x) where $x \notin C_i \cup \{b'_i\}$. Then, however, either $x \in F' \cup B' \setminus \{b'_i\}$, which is not possible since this would mean that x has out-degree 0 in A and b_i is not connected to b'_i, or $x \in C$. In this case there must be an arc from some vertex in D to some vertex in C_i or to b'_i in A in order to connect b_i to b'_i. Then, by a simple counting argument there is some j such that there is an arc in A from b_j to some vertex in F' which is impossible as we have already shown. Similarly, we can show that the arc ending in f'_{ij} starts in some vertex of $D_{ij} \cup \{f_{ij}\}$. This proves the claim.

Given a solution A to the 0-DSN-instance, we can assume, due to the above claim, that $A \subseteq \mathcal{A} \cup \mathcal{A}' \cup \mathcal{B} \cup \mathcal{C} \cup \mathcal{C}'$. For each i denote with v_i the uniquely determined vertex such that the arc (b_i, c_{v_i}) is in A. We claim that $K := \{v_i \mid 1 \leq i \leq k\}$ is a multi-colored clique in G. It is clearly multi-colored since the vertex v_i must have color i. Now, due to the claim, for each $1 \leq i < j \leq k$ there is exactly one edge $\{v'_i, v'_j\}$ in E such that $c(v'_i) = i$, $c(v'_j) = j$, and $\gamma'_{v'_i, v'_j} \in A$. Hence, if $v'_i \neq v_i$, then there is no path in A from b_i to f_{ij} since there is no arc from c_{v_i} to $d_{v'_i, v'_j}$ in $\mathcal{B}$. Thus, $v'_i = v_i$. By the same reasoning $v'_j = v_j$ and $\{v_i, v_j\}$ is an edge. This implies that K is a clique in G, completing the proof. □

8.4.2 Running-Time Improvements for Small Ratios

The DSN (0-DSN) algorithm developed by Feldman and Ruhl [FR06] uses an algorithm for SCSS (0-SCSS) as a subprocedure. Using the algorithms developed in the proofs of Theorems 8.4 and 8.5 in case of arc weight ratios 2 and 1, respectively, as the subprocedure, the running time of the overall algorithm of Feldman and Ruhl can be significantly improved by roughly halving the degree of its running time polynomial for the relevant case of these small arc weight ratios.

Corollary 8.8. Directed Steiner Network *with ratio* 2 *and* 0-Directed Steiner Network *with ratio* 1 *can be solved in time* $O((2^{2l} \cdot l^2 \cdot n^{2l})$ *and* $O(256^{l^2} n^{2l} + n^{2l+3})$*, respectively.*

Proof. The DSN (0-DSN)-algorithm of Feldman and Ruhl [FR06] is based on a simulation of a game in which one moves some tokens along the arcs of the

input digraph, the price of the move being the total weight of the arcs used. To this end, they construct a so-called "game graph", with vertices representing the possible token positions and arcs representing the legal moves. For DSN (0-DSN) with l terminal pairs a game graph for l tokens is constructed, and the solution is then determined by a single shortest path computation in this graph. One type of legal move is moving some $k \leq l$ tokens along some strongly connected subgraph to their new positions. The price of such a move can be determined by solving an instance of SCSS (0-SCSS) with up to $2l$ terminals corresponding to old and new positions of the tokens. For this computation our algorithms can be used. Since there are at most n^{2l} such moves and the time needed to perform the corresponding computations overshadows all other steps of the algorithm (see Section 6.2 of [FR06] for the details on the time complexity of the original algorithm), the result follows. □

8.5 Conclusion

In this chapter we contributed to parameterized complexity results for NP-hard problems on *directed* graphs, still a comparatively little developed field within the parameterized algorithmic graph theory (cf. [GY08]). We continued and extended previous work of Feldman and Ruhl [FR06]. In particular, we examined the impact of the ratio of the arc weights on the parameterized complexity of three Steiner problems with respect to the considered parameterizations. Table 8.1 in Section 8.1 summarizes known results and indicates open questions. More specifically, the parameterized complexity of SCSS, 0-SCSS, and DSN is unsettled for some small values of the arc weight ratio. We conjecture that the problems are hard in all these cases, but more clever arguments are needed to improve the reductions.

Given the vast literature on approximation algorithms, one may find many more questions to study concerning the parameterized complexity of network design problems in general, for instance, the connectivity augmentation problems with arc reversal and complement operations [AHS02]. Finally, it would also be interesting to investigate whether some restrictions on the structure of the underlying graph, such as planarity, could lead to fixed-parameter tractability results on Steiner-type problems (see Bateni et al. [BHM10] for some approximation results and Gassner [Gas10] for NP-hardness of steiner forest on treewidth 3 graphs).

Chapter 9

Generalized Domination

9.1 Introduction to Generalized Domination

9.1.1 Definition

Roughly speaking, in domination-type problems one is asked to place some objects to a subset of important places so that every place has an object nearby. This can mean placing a fire or police stations among villages of the country in such a fashion that a village without a station has one in a neighboring village. Among the first problems studied of such kind were also those asking for covering of the chessboard with various pieces.

The study of domination-type problems in graphs goes back to König [Kön50], Berge [Ber62] and Ore [Ore62]. During the development of the NP-hardness theory, domination problems were among the first to be studied and shown to be NP-complete [GJ79]. Around 1980 many papers appeared, either introducing a new variant of domination, showing its hardness or both. This resulted in that the bibliography on domination in graph [HL90] collected by Hedetniemi and Laskar soon grew over 500 entries. Hence an attempt to find a unifying generalization of many previously studied variants was made. It is based on the following notion.

Definition 9.1. Let σ, ρ be a pair of non-empty sets of non-negative integers. A set S of vertices of a graph G is called (σ, ρ)-*dominating* if for every vertex $v \in S$, we have $|S \cap N(v)| \in \sigma$, and for every $v \notin S$, we have $|S \cap N(v)| \in \rho$.

The concept of (σ, ρ)-domination was introduced by J.A. Telle [Tel94a, Tel94b] (and further elaborated on in [TP97, HT98]). Although it cannot capture all the different variants of domination studied (e.g., Connected Dominating Set, or Capacitated Dominating Set) it can also capture Independent Set and some of its variants. See Table 9.1 for some examples of how some variants of domination can be expressed by various σ and ρ.

It is well known that the optimization problems such as Maximum Independent Set, Minimum Dominating Set, etc. are NP-hard [GJ79]. Telle [Tel94a]

Table 9.1: Overview of some special cases of (σ, ρ)-domination and their parameterized complexity.

σ	ρ	Problem name	Parameterized Complexity
$\mathbb{N}_0$	$\mathbb{N}$	Dominating Set	Section 7.2
$\mathbb{N}$	$\mathbb{N}$	Total Dominating Set	W[2]-hard [BK94]
$\mathbb{N}_0$	$\{1\}$	Efficient Dominating Set	W[1]-hard [BK94]
$\{0\}$	$\mathbb{N}$	Independent Dominating Set	W[2]-complete [DF92b]
$\{0\}$	$\mathbb{N}_0$	Independent Set	Section 7.2
$\{0\}$	$\{1\}$	(1-)Perfect Code(Indep. Eff. D. S.)	W[1]-complete [DF95a],[Ces02]
$\{r\}$	$\mathbb{N}_0$	Induced r-Regular Subgraph	W[1]-hard [MT06]
$\{0\}$	$\{0,1\}$	Strong Stable Set	W[1]-hard
$\{1\}$	$\{1\}$	Total Perfect Dominating Set	W[1]-complete (Thm. 9.2)

has shown that whenever both σ and ρ are finite and non-empty, and $0 \notin \rho$ then already the existence of a (σ, ρ)-dominating set becomes NP-hard. On the other hand, when both σ and ρ are either set of all odd or of all even non-negative integers (ODD, EVEN), then the existence can be decided in polynomial time, while if we ask for a (σ, ρ)-dominating set of size at least k, exactly k, or at most k the question is again NP-hard [HKT00]. Finally if $0 \in \rho$ then the empty set is (σ, ρ)-dominating (regardless of σ), but the question for (σ, ρ)-dominating set of size at least k is again NP-hard for every finite σ and $\rho \neq \{0\}$ [GK07].

Although exact exponential time algorithms for (σ, ρ)-dominating set were designed [FGK$^+$07, FGK$^+$09b] (for the case that σ and ρ have together at most 3 elements or they can be obtained from so few elements by adding all multiples of some period m), the search for polynomial time algorithms focused the attention to special graph classes. In this direction Kratochvíl et al. [KMM95] showed polynomial-time decidability of the existence problem for any pair of finite σ, ρ on interval graphs, Golovach and Kratochvíl [GK07] established a P/NP-completeness dichotomy of this problem on chordal graphs and a way to classify σ, ρ in this dichotomy which was later generalized to degenerate graphs by the same authors in [GK08]. The generalized domination problems for graphs of bounded width-parameters (treewidth, branchwidth, cliquewidth, boolean-width) were considered in [ABXR$^+$10, BvLvRV10, vRBR09, Cha10, Tel94b, TP97].

9.1.2 Parameterized Complexity

Since the establishment of parameterized complexity domination-type problems have been among the first ones intensively studied in the framework of this theory. Actually Downey and Fellows in [DF98] state that the apparent parameterized in-

tractability of Dominating Set was their first motivation to develop the theory. As we have already said in Section 7.2, Independent Set is W[1]-complete and Dominating Set is W[2]-complete, if parameterized by the solution-size k. A number of domination-type problems is considered in [BK94], where it is shown (among other results) that Total Dominating Set is W[2]-hard and that Efficient Dominating Set is W[1]-hard. Independent Dominating Set is W[2]-complete [DF92b], while Efficient Independent Dominating Set (also called Perfect Code) is W[1]-complete ([DF95a] shows W[1]-hardness and [Ces02] shows W[1]-membership). The proof of W[1]-hardness of Strong Stable Set is not published (as we know) but easily follows from the W[1]-hardness of Independent Set. Again, all of the results are given with respect to the standard solution-size parameterization k. The complexity of finding an r-regular induced subgraph in a graph is studied by Moser and Thilikos in [MT06]. They proved that the problem is W[1]-hard when parameterized by the solution size but FPT for the dual parameterization.

Parity constraints have been considered in [DFVW99]. A subset of a color class of a bipartite graph is called *odd* (*even*) if every vertex from the other class has an odd (even, respectively) number of neighbors in the set. Downey et al. [DFVW99] show that deciding the existence of an odd set of size k, an odd set of size at most k, and an even set of size k are W[1]-hard problems; somewhat surprisingly, the complexity of Even Set of Size at Most k remains open.

All these individual results concern special (σ, ρ)-dominating sets, and thus call for a unifying approach. In this chapter, based on [GKS09], we initiate that by giving general results for large classes of pairs σ, ρ. Main results of this chapter are given in Table 9.2. For completeness, we also included in this table results which immediately follow from the fact proved by Telle [Tel94a] that it is NP-hard to test the existence of a (σ, ρ)-dominating set for any finite sets σ and ρ whenever $0 \notin \rho$. Following the approach of Telle [Tel94a] we focus mainly on finite σ and ρ, but we state the results in the most general form we were able to achieve. Our second goal in this chapter is to study (many of) the above problems from the dual parameterization point of view, both for the domination-type and for the parity-type problems.

Formally, we consider the following (σ, ρ)-domination problem

(σ, ρ)-Dominating Set of Size at most k
Input: A graph G and $k \in \mathbb{N}$.
Question: Is there a (σ, ρ)-dominating set in G of size at most k?
Parameter: solution-size k.

and its variants (σ, ρ)-Dominating Set of Size k, (σ, ρ)-Dominating Set of Size at least $n-k$, and (σ, ρ)-Dominating Set of Size $n-k$; the meaning should be clear. All these problems are parameterized by k, and in the latter two, n denotes the number of vertices of the input graph.

Table 9.2: Parameterized complexity of deciding the existence of a (σ, ρ)-dominating set of a given size.

size	parameterized complexity
$\leq k$	W[1]-complete for finite σ, ρ, $0 \notin \rho$ (Thm. 9.2)
$= k$	W[1]-complete for finite σ, ρ, $0 \notin \rho$ (Thm. 9.2)
$\geq k$	para-NP-complete for finite σ, ρ, $0 \notin \rho$ [Tel94a]
$\leq n - k$	para-NP-complete for finite σ, ρ, $0 \notin \rho$ [Tel94a]
$= n - k$	Open
$\geq n - k$	FPT for finite or co-finite σ, ρ (Thm. 9.12)

The first result of this chapter determines the parameterized complexity for finite sets σ and ρ. We prove in Section 9.2 that both (σ, ρ)-Dominating Set of Size k and (σ, ρ)-Dominating Set of Size at most k are W[1]-complete problems whenever $0 \notin \rho$. The W[1]-membership is proved in a stronger form when σ is only required to be recursive but not necessarily finite. Recall that a set of non-negative integers is called recursive, if there is a deterministic algorithm, that given a non-negative integer k decides in finite time, whether k is in the set or not.

We further study in Section 9.3 the dually parameterized problems and show in an even more general way (also for co-finite sets) that these problems become tractable. We prove that for non-empty sets of non-negative integers σ and ρ such that either σ or $\overline{\sigma}$ is finite (recall that $\overline{X} = \mathbb{N}_0 \setminus X$ for a set X of integers), and similarly either ρ or $\overline{\rho}$ is finite, the (σ, ρ)-Dominating Set of Size at least $n - k$ problem is in FPT.

In Section 9.4 we show that a similar result cannot be expected for arbitrary recursive sets σ and ρ. Even for the simplest case of sets that are neither finite nor cofinite, that is, for the parity case we prove W[1]-hardness if $\sigma, \rho \in \{\text{EVEN}, \text{ODD}\}$ (recall that $\text{EVEN} = \{0, 2, 4, 6, \ldots\}$ and $\text{ODD} = \{1, 3, 5, \ldots\}$).

As a tool for the previous result we consider the following parity problems on bipartite graphs. Suppose that G is a bipartite graph and R, B is a bipartition of its set of vertices (vertices of R are called *red* and vertices of B are *blue*). A non-empty set $S \subseteq R$ is called *even* if for every vertex $v \in B$, we have $|N(v) \cap S| \in \text{EVEN}$, and it is called *odd* if for every vertex $v \in B$, we have $|N(v) \cap S| \in \text{ODD}$. The following problem

Even Set of Size at least $r - k$
Input: A bipartite graph $G = (R, B, E)$ such that $|R| = r$ and $k \in \mathbb{N}$.
Question: Is there an even set in R of size at least $r - k$?
Parameter: dual parameterization k.

and its variants Even Set of Size $r - k$, Odd Set of Size at least $r - k$, and Odd Set of Size $r - k$ are the dually parameterized versions of Odd Set,

Exact Odd Set, Even Set and Exact Even Set studied in [DFVW99]. We prove in Section 9.4 that all four of them are W[1]-hard. We believe that these results are interesting by themselves. In particular it is unusual for parameterized problems to have the same complexity for dual parameterizations.

Sections 9.5 and 9.6 are devoted to FPT results for sparse graphs and open problems.

9.2 Complexity of (σ, ρ)-Dominating Set of Size at most k

Here we prove the following theorem.

Theorem 9.2. *Let σ and ρ be non-empty finite sets of non-negative integers, $0 \notin \rho$. Then both (σ, ρ)-Dominating Set of Size k and (σ, ρ)-Dominating Set of Size at most k are W[1]-complete problems.*

The remaining part of this section contains the proof of this theorem. First, we prove W[1]-hardness. To do it, we introduce and consider an auxiliary problem.

9.2.1 At most α-Satisfiability

To prove the hardness part of Theorem 9.2, we are going to reduce from a special variant of the Weighted Satisfiability problem. Recall that a weight of a truth assignment is the number of variables having value *true*.

At most α-Satisfiability
Input: A 2-normalized monotone Boolean formula φ and $k \in \mathbb{N}$.
Question: Does φ allow a satisfying truth assignment of weight at most k such that each clause of φ contains at most α variables which evaluate to *true*?
Parameter: weight of the assignment k.

We start with the proof of W[1]-hardness for this problem.

Lemma 9.3. *For any $\alpha \geq 1$,* At most α-Satisfiability *is W[1]-hard.*

Proof. We provide a reduction from the following problem:

Exact Satisfiability
Input: A 2-normalized Boolean formula φ and $k \in \mathbb{N}$.
Question: Does φ have a satisfying truth assignment of weight at most k such that each clause of φ contains exactly one literal which evaluates to *true*?
Parameter: weight of the assignment k

In [DF95b] an easy reduction from Perfect Code (which is shown there to be W[1]-hard) to a variant of Exact Satisfiability asking for an assignment

of weight exactly k is given. This reduction produces only monotone formulas. As it can be easily shown that all perfect codes in a given graph have the same cardinality [Kra91] it follows that also our variant of the problem is W[1]-hard, even for monotone formulas.

It suffices to prove the lemma for $\alpha \geq 2$ as At most 1-Satisfiability is the same problem as Exact Satisfiability for monotone formulas.

We reduce from Exact Satisfiability. Let $C_1, \ldots, C_m$ be the clauses of φ. We introduce α copies of φ with different sets of variables, and denote them by $\varphi_1, \ldots, \varphi_\alpha$. Let $C_{i,1}, \ldots, C_{i,m}$ be the clauses of φ_i. Define $\psi = \varphi_1 \wedge \cdots \wedge \varphi_\alpha \wedge [(C_{1,1} \vee \cdots \vee C_{\alpha,1}) \wedge \cdots \wedge (C_{1,m} \vee \cdots \vee C_{\alpha,m})]$, and let $k' = k\alpha$.

Suppose that φ has a satisfying truth assignment of weight at most k such that each clause of φ contains exactly one variable with value *true*. Using the same truth assignment for the sets of variables for each φ_i we get a satisfying truth assignment of weight at most k' for ψ such that each clause contains at most α variables with value *true*.

For the converse, assume that ψ has a satisfying truth assignment of weight at most k' such that each clause of ψ contains at most α variables with value *true*. Obviously each clause $C_{i,j}$ contains at least one variable with value *true*, but it cannot have two variables with value *true*, since otherwise the clause $C_{1,j} \vee \cdots \vee C_{\alpha,j}$ of ψ would contain more than α variables of value *true*. So, all formulas φ_i have a truth assignment such that each clause contains exactly one variable with value *true*. It remains to note that by a pigeonhole principle at least one formula φ_i has a truth assignment of weight at most $(k'/\alpha) = k$ as the formulas have different sets of variables. □

9.2.2 The Proof of W[1]-hardness

This subsection contains the proof of W[1]-hardness of (σ, ρ)-Dominating Set of Size k and (σ, ρ)-Dominating Set of Size at most k by a reduction from At most α-Satisfiability.

Suppose that σ and ρ are non-empty finite sets of non-negative integers and $0 \notin \rho$. Let us denote $p_{min} = \min \sigma$, $p_{max} = \max \sigma$, $q_{min} = \min \rho$ and $q_{max} = \max \rho$. Further we set $t = \max\{i \in \mathbb{N}_0 \colon i \notin \rho, i+1 \in \rho\}$ (t is correctly defined since $0 \notin \rho$), and $\alpha = q_{max} - t \geq 1$.

We first construct several auxiliary gadgets. These gadgets "enforce" on a given vertex the property of "not belonging to any (σ, ρ)-dominating set" and at the same time guarantee that this vertex has a given number of neighbors in any (σ, ρ)-dominating set in the gadget. To describe the properties formally, we consider rooted graphs and introduce the following notion. Let G be a rooted graph with a set of root vertices X. We call a set $S \subseteq V(G)$ a (σ, ρ)-*dominating set for* G if $|N(v) \cap S| \in \sigma$ for every $v \in S \setminus X$, and $|N(v) \cap S| \in \rho$ for every $v \notin S$, $v \notin X$ (i.e., the conditions from the definition of (σ, ρ)-domination are required for all vertices except for the roots).

The gadget $F(s)$. The first gadget we introduce is the gadget $F(s)$ (the number in brackets always refers to the number of roots of the gadget). The construction goes as follows (see Fig. 9.1): We take a complete graph $K_{p_{max}+1}$ with vertices $a_1,\ldots,a_{p_{max}+1}$. For each vertex a_i, we add $q_{max}+1$ vertices $b_{i,1},\ldots,b_{i,q_{max}+1}$ and join them to a_i by edges. For each vertex $b_{i,j}$, we add $q_{min}-1$ copies of the complete graph $K_{p_{min}+1}$ and make one vertex of each copy adjacent to $b_{i,j}$. Finally, s vertices $x_1,\ldots,x_s$ are added and joined to a_1. The resulting graph is denoted by $F(s)$ and $x_1,\ldots,x_s$ are its roots.

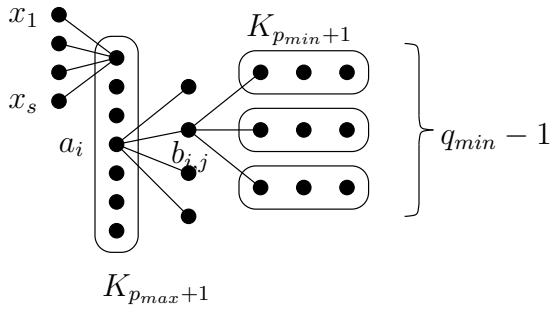


Figure 9.1: Construction of $F(s)$

Lemma 9.4. *The graph $F(s)$ has a unique (σ,ρ)-dominating set S for $F(s)$, and this set has the following properties:*

1. $x_1,\ldots,x_s \notin S$,
2. $a_1 \in S$ *(i.e., every root vertex has exactly one neighbor in S),*
3. S *contains* $f=(p_{max}+1)((q_{max}+1)(q_{min}-1)(p_{min}+1)+1)$ *vertices.*

Proof. Let S consist of all vertices a_i and all vertices of all $(p_{max}+1)(q_{max}+1)(q_{min}-1)$ copies of $K_{p_{min}+1}$. It is easy to check that S is a (σ,ρ)-dominating set for $F(s)$ and that S satisfies properties 1–3. We prove that S is unique. Let S be a (σ,ρ)-dominating set for $F(s)$. Suppose that some vertex a_i is not in S. If a neighbor $b_{i,j}$ is also not in S, then this neighbor can have at most $q_{min}-1$ neighbors in S, but this is impossible. So, all vertices $b_{i,j}$, $j=1,2,\ldots,q_{max}+1$ are in S, but then a_i has at least $q_{max}+1$ neighbors in S, and this is again a contradiction. Hence all vertices a_i, $i=1,2,\ldots,p_{max}+1$ are in S, and hence all their neighbors (i.e., $x_1,\ldots,x_s$ and $b_{1,1},\ldots,b_{p_{max}+1,q_{max}+1}$) are outside S. It follows that all neighbors of $b_{i,j}$ have to be included to S. This means that each copy of $K_{p_{min}+1}$ has at least one vertex in S and consequently all vertices of these copies are included in S. □

The gadget $F'(s)$. We take q_{max} copies of $F(1)$ and identify their roots into one vertex x. Then s root vertices $y_1, \ldots, y_s$ of degree one are added and each is made adjacent to x. The resulting graph is denoted by $F'(s)$.

Lemma 9.5. *The graph $F'(s)$ has a unique (σ, ρ)-dominating set S for $F'(s)$, and this set has the following properties:*

1. *$x, y_1, \ldots, y_s \notin S$, (i.e., roots have no neighbors in S)*
2. *S contains $f' = q_{max}(p_{max}+1)((q_{max}+1)(q_{min}-1)(p_{min}+1)+1)$ vertices.*

Proof. Clearly, the union of (σ, ρ)-dominating sets for the copies of $F(1)$ is a (σ, ρ)-dominating set for $F'(s)$, and this set has properties 1–2. Now note that any (σ, ρ)-dominating set S for $F'(s)$ must include (σ, ρ)-dominating sets for the copies of $F(1)$, and by Lemma 9.4, $x \notin S$. Since, by the same lemma, x has a neighbor from S in each copy of $F(1)$ and, thus, it has already q_{max} neighbors in S, none of $y_1, \ldots, y_s$ is in S. □

Using these gadgets we construct an auxiliary graph $H(l)$ for every positive integer l.

The $H(l)$ gadget. We start with a complete graph K_l with vertices $z_1, \ldots, z_l$, which will be the only roots of $H(l)$. We consider three cases:

$q_{max} = q_{min} = 1$: A copy of $F'(1)$ with the root y is added, and y is joined to $z_1, \ldots, z_l$ by edges.

$q_{max} > q_{min} = 1$: We take $q_{max} - 1$ copies of $F(1)$ with a common root x, and we add a copy of $F'(1)$ with the root y. Vertices x and y are made adjacent to $z_1, \ldots, z_l$.

$q_{max} \geq q_{min} > 1$: We introduce $q_{max} - 1$ copies of $F(1)$ with a common root x, and we add further $q_{min} - 1$ copies of $F(1)$ with a common root y. Vertices x and y are made adjacent to $z_1, \ldots, z_l$.

Lemma 9.6. *The graph $H(l)$ has the following properties:*

1. *For any (σ, ρ)-dominating set S for $H(l)$, each root vertex has no non-root neighbors in S.*
2. *Any (σ, ρ)-dominating set S for $H(l)$ contains exactly one root vertex.*
3. *For each root vertex z_i, there is a (σ, ρ)-dominating set S for $H(l)$ which contains z_i.*
4. *Any (σ, ρ)-dominating set S for $H(l)$ contains h vertices, where*

$$h = h(\sigma, \rho) = \begin{cases} f' + 1, & \text{if } q_{max} = q_{min} = 1, \\ (q_{max} - 1)f + f' + 1, & \text{if } q_{max} > q_{min} = 1, \\ (q_{max} + q_{min} - 2)f + 1, & \text{if } q_{max} \geq q_{min} > 1. \end{cases}$$

Proof. By Lemmas 9.4 and 9.5, the vertices x and y do not belong to any (σ,ρ)-dominating set for $H(l)$. Hence the first claim follows. The second claim follows from the observation that every root vertex has an adjacent non-root vertex y with $q_{min}-1$ neighbors in S (since $y \notin S$, at least one root vertex has to be in S) and a neighbor (x or y) with $q_{max}-1$ neighbors in S (and hence at most one root vertex may be in S). To prove the third and forth claims, observe that every (σ,ρ)-dominating set for $H(l)$ contains the union of the (σ,ρ)-dominating sets for all copies of $F(1)$ and of $F'(1)$, which were used in the construction, and this union plus one arbitrary root vertex z_i is indeed a (σ,ρ)-dominating set for $H(l)$. □

As a next step, we construct a selection gadget $R(l)$ for every $l \in \mathbb{N}$.

The selection gadget $R(l)$. Take $(p_{min}+1)q_{min}$ copies of $H(l)$ which are denoted $H_{i,j}(l)$ for $i \in \{1,\dots,p_{min}+1\}$ and $j \in \{1,\dots,q_{min}\}$. Let $z_1^{(i,j)},\dots,z_l^{(i,j)}$ be the roots of $H_{i,j}(l)$. For each $i \in \{1,\dots,p_{min}+1\}$, and each pair $j,j' \in \{1,\dots,q_{min}\}$, $j \neq j'$, the vertices of the root sets for $H_{i,j}(l)$ and $H_{i,j'}(l)$ are joined by the complements of perfect matchings, i.e., vertices $z_s^{(i,j)}$ and $z_{s'}^{(i,j')}$ are adjacent if and only if $s \neq s'$. Then for each $j \in \{1,\dots,q_{min}\}$ and any $s \in \{1,\dots,l\}$, the vertices $z_s^{(1,j)},\dots,z_s^{(p_{min}+1,j)}$ are made pairwise adjacent to form a clique. The roots of this constructed graph $R(l)$ are the vertices $z_1^{(1,1)},\dots,z_l^{(1,1)}$.

Lemma 9.7. *1. Any (σ,ρ)-dominating set S for $R(l)$ contains exactly one vertex from the set $\{z_1^{(1,1)},\dots,z_l^{(1,1)}\}$.*

2. For any vertex $z_s^{(1,1)}$, there is a (σ,ρ)-dominating set in $R(l)$ which contains this vertex.

3. Any (σ,ρ)-dominating set in $R(l)$ has $r = r(\sigma,\rho) = (p_{min}+1)q_{min}h$ vertices.

Proof. Observe that any (σ,ρ)-dominating set S for $R(l)$ induces (σ,ρ)-dominating sets for the graphs $H_{i,j}(l)$ for $i \in \{1,\dots,p_{min}+1\}$ and $j \in \{1,\dots,q_{min}\}$. Hence the first claim of the lemma follows from the second claim of Lemma 9.6. To prove the second claim, consider the union of (σ,ρ)-dominating sets for the rooted graphs $H_{i,j}$ which contain the vertices $z_s^{(i,j)}$. These sets exist because of the third claim of Lemma 9.6. Note explicitly that for this set, $z_s^{(i,j)}$ has exactly p_{min} neighbors in S and every other root vertex of $H_{i,j}(l)$ has exactly q_{min} neighbors in S by the first claim of Lemma 9.6 and the construction of $R(l)$. Therefore we have a (σ,ρ)-dominating set in $R(l)$ which contain $z_s^{(1,1)}$. The third claim follows immediately from the fourth claim of $z_s^{(1,1)}$. □

Now we are ready to describe the reduction. Let φ be a formula as an input of the At most α-Satisfiability problem. Let $x_1,\dots,x_n$ be its variables, and let $C_1,\dots,C_m$ be the clauses.

We take k copies of the graph $R(n+1)$ denoted by $R_1, \ldots, R_k$, with the roots of R_i being denoted by $x_{i,j}$. For each clause C_s a vertex C_s is added and joined by edges to all vertices $x_{i,j}$, $i = 1, \ldots, k$ such that the variable x_j occurs in the clause G_s. Observe that vertices $x_{i,n+1}$ are not joined with any C_s; they correspond to the case that less than k variables is set to true. Now we distinguish two cases (recall that $t = \max\{i \in \mathbb{N}_0 \colon i \notin \rho, i+1 \in \rho\}$):

$t = 0$: In this case a copy of $F'(m)$ is introduced, and the m roots of this gadget are identified with vertices $C_1, \ldots, C_m$. In this case we set $k' = kr + f'$.

$t > 0$: We construct t copies of $F(m)$, and the roots of each copy are identified with $C_1, \ldots, C_m$. In this case we set $k' = kr + tf$.

The resulting graph is called G. The proof of W[1]-hardness is then concluded by the following lemma.

Lemma 9.8. *The formula φ allows a satisfying truth assignment of weight at most k such that each clause of φ contains at most α variables with value* true *if and only if G has a (σ, ρ)-dominating set of size at most k'. Moreover, in such a case the size of any (σ, ρ)-dominating set is exactly k'.*

Proof. Suppose that the variables $x_1, \ldots, x_n$ have a satisfying truth assignment of weight at most k such that each clause of φ contains at least one variable, and at most α variables with value *true*. Without loss of generality we assume that $x_j = true$ for $j \in \{1, \ldots, l\}$, $x_j = false$ for $j \in \{l+1, \ldots, n\}$ and $l \leq k$. We construct a (σ, ρ)-dominating S set for G as follows. For each $j \in \{1, \ldots, l\}$, all vertices of the (σ, ρ)-dominating set of R_j which contains $x_{j,j}$ are included in S (see Lemma 9.7). Notice that the satisfying truth assignment can have weight strictly lower that k, i.e. $l < k$. In this case for each $j \in \{l+1, \ldots, k\}$, all vertices of the (σ, ρ)-dominating set of R_j which contains $x_{j,n+1}$ are included in S. For each copy of $F(m)$ (or $F'(m)$), all vertices of corresponding (σ, ρ)-dominating sets (see Lemmas 9.4 and 9.5) for these rooted graphs are added to S. By Lemmas 9.4, 9.5 and 9.7 we know that $|S| = k'$. By the same lemmas, for any vertex $v \neq C_1, \ldots, C_m$, the (σ, ρ)-conditions are satisfied, and we have to check them only for vertices C_s. Since $C_s \notin S$ (see Lemmas 9.4 and 9.5), it is necessary to prove that $|S \cap N(C_s)| \in \rho$. One more time using Lemmas 9.4 and 9.5 we note that C_s has t neighbors in S from gadgets $F(m)$ or $F'(m)$. Each clause C_j contains at least one variable and at most $\alpha = q_{max} - t$ variables with value *true*. By the construction of S and Lemma 9.7, each vertex C_s has at least one and at most $\alpha = q_{max} - t$ neighbors in S from gadgets $R_1, \ldots, R_k$. Therefore $t + 1 \leq |S \cap N(C_s)| \leq t + \alpha = q_{max}$, and $\{t+1, \ldots, q_{max}\} \subseteq \rho$.

Assume now that S is a (σ, ρ)-dominating set of size at most k' in G. By Lemmas 9.4, 9.5 and 9.7, S is the union of the (σ, ρ)-dominating sets of the graphs $R_1, \ldots, R_k$ and the (σ, ρ)-dominating sets for the gadgets $F(m)$ (or $F'(m)$) (note that it means that $|S| = k'$). It follows from Lemma 9.6 that for each $i \in \{1, \ldots, k\}$, S contains exactly one vertex from the set $\{x_{i,1}, \ldots, x_{i,n+1}\}$. For each

$j \in \{1,\ldots,n\}$, we set the variable $x_j = true$ if $x_{i,j} \in S$ for some $i \in \{1,\ldots,k\}$ and $x_j = false$ otherwise. Clearly we have a truth assignment of weight at most k. By Lemmas 9.4 and 9.5, vertices C_s are not in S, and each vertex C_s has t neighbors in S from gadgets $F(m)$ or $F'(m)$. Since S is a (σ,ρ)-dominating set, C_s has at least one and at most $q_{max} - t = \alpha$ neighbors in S from the graphs $R_1,\ldots,R_k$. Recall that C_s is not adjacent with vertices $x_{i,n+1}$. Hence the neighbors of C_s in S are vertices $x_{i,j}$ for $j \in \{1,\ldots,n\}$ which correspond to the variables that were set *true*. It follows immediately that each clause C_s contains at least one and at most α variables with value *true*. □

9.2.3 W[1]-membership

To complete the proof of Theorem 9.2, it remains to prove that our problems are included in W[1]. Here we prove a slightly stronger claim.

Theorem 9.9. *Let σ be recursive, and suppose that ρ is finite or $\rho = \mathbb{N}_0$. Then the (σ,ρ)-*Dominating Set of Size at most k *and (σ,ρ)-*Dominating Set of Size k *problems are in W[1].*

To show the membership of the problems in W[1], we use the characterization of W[1] by Nondeterministic Random Access Machines as presented in Section 7.3.

We introduce our program `SigmaRho` (Algorithm 1) for the case ρ is finite, that takes a graph G and a positive integer k as an input and there is an accepting computation of `SigmaRho` on G and k if and only if there is a (σ,ρ)-dominating set of size exactly k in G. We present it in a higher level language that can be easily translated to the NRAM instructions. We will then show that this program is tail-nondeterministic k-restricted (Definition 7.9).

Lemma 9.10. *Let G be a graph and $k \in \mathbb{N}$. There is an accepting computation of `SigmaRho` on G and k (see Algorithm 1) if and only if there is a (σ,ρ)-dominating set of size (exactly) k in G.*

Proof. We will show that the program `SigmaRho` accepts the input if and only if the set S guessed in step 2 is a (σ,ρ)-dominating set of size k for the input graph G. Clearly if the program accepts, then S contains k distinct vertices, otherwise it would have been rejected in step 3. It is easy to see that the members of the set S must satisfy the σ-condition due to step 4. Now observe that the number $D(r)$ computed in step 5 denotes the number of pairs (R,v) such that R is a subset of S of size r and v is a vertex not in S that has all vertices from R as neighbors (the first term counts all such vertices v in V and the second term subtracts such vertices v that are in S). Hence this $D(r)$ represents the number of vertices outside S which have at least r neighbors in S with multiplicities, in particular a vertex with t neighbors in S is counted $\binom{t}{r}$ times. Since, in the first

Program `SigmaRho`$(G = (V, E), k)$

1 **for** $r := 1$ **to** $q_{max} + 1$ **do forall** $R \in \binom{V}{r}$ **do**

$$B(R) := |\bigcap_{u \in R} N_G(u)| = |\{v | v \in V, \forall u \in R : uv \in E\}|;$$

2 **Guess** k **vertices** $v_1, \ldots, v_k$**, denote** $S = \{v_1, \ldots, v_k\}$**;**

3 **forall** $i, j, 1 \leq i < j \leq k$ **do if** $v_i = v_j$ **then REJECT;**

4 **for** $i := 1$ **to** k **do if** $|\{v_j | v_i v_j \in E\}| \notin \sigma$ **then REJECT;**

5 **for** $r := q_{max} + 1$ ***downto*** 1 **do**

$$\begin{aligned} D(r) &:= \sum_{R \in \binom{S}{r}} (B(R) - |\bigcap_{u \in R} N_G(u) \cap S|) = \\ &= \sum_{R \in \binom{S}{r}} |\{v | v \in V \setminus S, \forall u \in R : uv \in E\}|; \\ C(r) &:= D(r) - \sum_{t=r+1}^{q_{max}} \left(\binom{t}{r} \cdot C(t) \right); \end{aligned}$$

if $r \notin \rho$ *and* $C(r) \neq 0$ **then REJECT;**

6 **if** $0 \notin \rho$ *and* $\sum_{r \in \rho} C(r) \neq n - k$ **then REJECT; else ACCEPT;**

Algorithm 1: The algorithm from the proof of Theorem 9.9. Recall that $q_{max} = \max \rho$.

run of the cycle 5 with $r = q_{max} + 1$, we check that there is no vertex outside S with more than q_{max} neighbors in S, it follows that $C(r)$ represents the number of vertices outside S which have exactly r neighbors in S. It is now clear that if $r \notin \rho$ and there is a vertex outside S with r neighbors in S (i.e., $C(r) > 0$), then S cannot form a (σ, ρ)-dominating set. In the last step 6 we sum up the number of vertices outside S that satisfy the ρ-condition and thus S (which satisfies all the conditions checked by the previous steps) is (σ, ρ)-dominating if and only if this sum is equal to the total number of vertices outside S, i.e., $n - k$, or $0 \in \rho$. □

Lemma 9.11. *Program* `SigmaRho` *is tail-nondeterministic k-restricted.*

Proof. To prove the lemma, we prove the following claims.

Claim 1: There is a function $g(k)$ such that steps 2-6 can be performed in at most $g(k)$ steps.

Proof of Claim 1: Step 2 is a simple k times execution of the "GUESS" instruction. Hence it is carried out in $O(k)$ time. Step 3 takes $O(k^2)$ time. If $a(k)$ denotes the maximum time needed for $l \leq k$ to decide whether $l \in \sigma$ (such a function exists since σ is recursive), then step 4 can be carried out in $O(k^2 \cdot a(k))$. We can also suppose that $a(k)$ bounds any numbers involved in this computation and the number of registers used. The cycle in 5 is executed constantly $(q_{max} + 1)$

many times. The value $D(r)$ is computed according to the first expression, where there is at most $O(k^{q_{max}+1})$ different indices for the sum, the first term means just a table lookup and the second term can be determined in $O((q_{max}+1)\cdot k)$ steps for each fixed index. The expression for $C(r)$ contains constantly many (at most q_{max}) terms. With the last step taking also constant time, this means that steps 2-6 altogether can take at most $g(k) = O(k^{q_{max}+2}+k^2\cdot a(k))$ time. Since the first nondeterministic instruction is in step 2, nondeterministic steps are among the last $g(k)$ steps.

Claim 2: Step 1 can be carried out in $O(n^{q_{max}+2})$ time.

*Proof of Claim 2:*There are at most $O(n^{q_{max}+1})$ subsets of size at most $q_{max}+1$ in V, $|V|=n$ and for each subset R the computation of $B(R)$ can be performed in $O((q_{max}+1)\cdot n)$ time. Together with Claim 1 this shows the first condition of the definition.

Claim 3: During the computation the numbers involved and the number of registers used are bounded by $O(n^{q_{max}+2})$, except for step 4, where the bound is $a(k)$.

Proof of Claim 3: There is $n^{q_{max}+1}$ many $B(R)$'s, a constant number of $D(r)$'s and $C(r)$'s, k vertices of S, the input and an additional constant number of variables for the indices stored during the computation. The $B(R)$'s are bounded by n since they contain the number of vertices satisfying certain conditions. Each $D(r)$ is a sum of at most $n^{q_{max}+1}$ $B(R)$'s and hence bounded by $n^{q_{max}+2}$. The $C(r)$ and the sum in the last step are sums of constantly many $D(r)$'s and $C(r)$'s, respectively, hence bounded by $O(n^{q_{max}+2})$.

The bound for step 4 is proved in Claim 1. This shows the second and third conditions of the definition, completing the proof. □

Proof of Theorem 9.9. The W[1]-membership of (σ,ρ)-Dominating Set of Size k for the case ρ is finite is a direct consequence of Theorem 7.10 together with Lemmas 9.10 and 9.11. To show the membership of (σ,ρ)-Dominating Set of Size at most k it suffices to add one more nondeterministic step "Guess $l \le k$" and the assignment "$k := l$" before Step 2 of `SigmaRho`. To modify the program `SigmaRho` for the case $\rho = \mathbb{N}_0$ it is enough to omit Steps 1, 5, and 6. □

This completes the proof of Theorem 9.2.

9.3 Complexity of (σ,ρ)-Dominating Set of Size at least $n-k$

Now we consider our problems for the dual parameterization. Note that the studied class contains (except for others) also Vertex Cover (as a dual of Independent Set), probably the most studied problem in parameterized complexity

that is well known to be FPT (see Sections 5.1 and 6.1). The dual parameterization of r-Regular Induced Subgraph was shown to be FPT in [MT06]. We provide a common generalization for these results.

Theorem 9.12. *Let σ and ρ be sets of non-negative integers such that either σ or $\overline{\sigma}$ is finite, and similarly either ρ or $\overline{\rho}$ is finite. Then (σ,ρ)-Dominating Set of Size at least $n-k$ is in FPT.*

Proof. We present an algorithm (Algortihm 2) that is based on the bounded search trees technique, presented in Section 6.1. At the beginning the algorithm includes all vertices into the set S and then tries recursively excluding some of the vertices to make S (σ,ρ)-dominating. Once a vertex is excluded, it is never included in the set again (in the same branch of the algorithm). Obviously at most k vertices can be excluded from S to fulfill the size constraint.

We call a vertex v *satisfied* (with respect to the current set S) if it has the right number of neighbors in S (i.e., $v \in S$ and $|N(v) \cap S| \in \sigma$ or $v \notin S$ and $|N(v) \cap S| \in \rho$), otherwise we call it *unsatisfied*. Let $\widetilde{p}_{max}$ denote $\max \sigma$ if σ is finite and $\max \overline{\sigma}$ if $\overline{\sigma}$ is finite. Similarly let $\widetilde{q}_{max}$ denote $\max \rho$ or $\max \overline{\rho}$. (It is assumed here that $\max \emptyset = -\infty$.) Finally let b denote $\max\{\widetilde{p}_{max}, \widetilde{q}_{max}\}$. We call a vertex v *big* if $\deg(v) > b + k$ and *small* otherwise.

The main idea of the algorithm is that there is at most one way to make an unsatisfied big vertex satisfied — to exclude it from S — and if this does not work, there is no (σ,ρ)-dominating set at all. On the other hand to satisfy a small vertex, we must either exclude it or one of its first b neighbors that were in S.

```
Procedure Exclude(S)
    if there is no unsatisfied vertex then Return(S);Exit;
    if |S| = n − k then Halt;
    let v be an unsatisfied vertex;
    if v is big then
        if v ∈ S and ρ is infinite then Exclude(S \ v);
        else Halt;
    else
        if v ∈ S then Exclude(S \ v);
        let {u_1, ..., u_r} = S ∩ N(v) be the set of included neighbors of v;
        if r = 0 then Halt;
        for i := 1 to min{b + 1, r} do Exclude(S \ {u_i}).
```

Algorithm 2: The algorithm from the proof of Theorem 9.12.

The algorithm consists of a single call Exclude(V) and returns the set S returned by the procedure or NO if no set was returned.

Claim 1: The algorithm Exclude(V) runs in $O((b+2)^k \cdot n + m)$ time.

Proof of Claim 1: First observe that since each recursive call reduces the size of S by one, there can be at most k nested calls. With at most $b+2$ recursive calls made by one call this means altogether $O((b+2)^k)$ calls of `Exclude`. Note that we can decide which vertices are satisfied at the beginning in $O(m)$ time and then update this before each recursive call in $O(n)$ time. Since all the other operations in one call can be also carried out in $O(n)$ time, we get the claimed running time.

Claim 2: If there is a (σ, ρ)-dominating set O of size at least $n-k$, then the algorithm returns some (σ, ρ)-dominating set of size at least $n-k$.

Proof of Claim 2: We show that there is always a branch of the algorithm that keeps $O \subseteq S$, thus we cannot miss O. A big vertex has always at least b neighbors in any set of vertices of size at least $n-k$. Hence if the unsatisfied vertex v is big, this means that σ is finite, and since it is satisfied by O, this means $v \notin O$, which is the only branch tested by the algorithm (ρ must be infinite, because otherwise there would be no (σ, ρ)-dominating set). On the other hand, if v is small and in S, then either $v \notin O$ or $\{u_1, \ldots, u_{min\{b+1,r\}}\} \not\subseteq O$ because otherwise v would have either the same neighborhood or at least $b+1$ neighbors in O and it would remain unsatisfied, since σ is finite and thus $b+1 \notin \sigma$. Similarly for the other case.

Clearly if the algorithm returns a set S, then S is a (σ, ρ)-dominating set of size at least $n-k$, since all vertices are satisfied. This together with the two claims completes the proof. □

9.4 Complexity of Parity Constraints

We proved that our problems are in FPT for the dual parameterization if σ, ρ are finite or co-finite. Now we show that it cannot be expected that similar results could be established in more general cases. Particularly, these problems are W[1]-hard for $\sigma, \rho \in \{\text{EVEN}, \text{ODD}\}$. Note that the sets EVEN and ODD constitute the simplest examples of sets that are neither finite nor co-finite. This was also one of the reasons why similar problems were studied in [HKT00].

9.4.1 Complexity for the Bipartite Parity Problems

Recall that it was shown by Downey et al. [DFVW99] that for a bipartite graph $G = (R, B, E)$, deciding the existence of an odd set of red vertices (i.e. of a subset of R such that each blue vertex from B has an odd number of neighbors in this set) of size k, an odd set of size at most k, and an even set of size k are W[1]-hard problems. As a counterpart to these results, we first show that all four parity problems for Red/Blue bipartite graphs are hard under the dual parameterization.

Theorem 9.13. *The* Even Set of Size $r-k$, Even Set of Size at least $r-k$, Odd Set of Size $r-k$, *and* Odd Set of Size at least $r-k$ *problems are all W[1]-hard.*

Proof. We reduce from the following problem

Odd Set of Size at most k:
Input: A bipartite graph $G = (R, B, E)$ and $k \in \mathbb{N}$.
Question: Is there an odd set in R of size at most k?
Parameter: solution-size k.

It should be noted that W[1]-hardness was stated in [DFVW99] for the exact variant of the problem (i.e. for the question: Is there an odd set in R of size k?), but for our variant of the question, the proof of [DFVW99] works the same. We show that the problem remains W[1]-hard if all blue vertices have odd degrees and also if all of them have even degrees. Then we deduce the claims by considering the set $R \setminus S$ for a would-be odd set $S \subseteq R$.

Lemma 9.14. Odd Set of Size at most k *remains W[1]-hard even if*

1. *all blue vertices have odd degrees;*
2. *all blue vertices have even degrees.*

Proof. To prove the first claim, we reduce from Odd Set of Size at most k. Let $G = (R \cup B, E)$ and k be an instance of the problem and let $B' \subseteq B$ be the set of vertices of even degree. If $B' = \emptyset$, then we let $H = G$. Otherwise the graph H is constructed as follows. Red vertices $a, b, c_1, \ldots, c_k$ and blue vertices $f, d_1, \ldots, d_k$ are added to G. Then all vertices of B' are joined by edges with a, vertices a and b are joined with $d_1, \ldots, d_k$, vertex b is connected with f, and finally each vertex c_i is joined with d_i. The construction of H is shown in Fig. 9.2 a). Clearly, all blue vertices of H have odd degrees. Let $k' = k$ if $B' = \emptyset$ and $k' = k + 1$ otherwise. We prove that G has an odd set of size k if and only if H has an odd set of size k'.

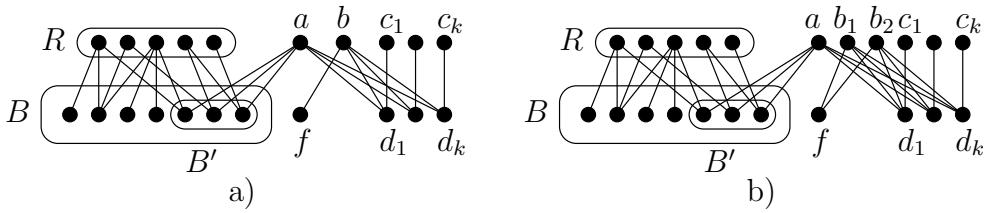


Figure 9.2: Construction of H in the proof of Lemma 9.14

If $B' = \emptyset$, then the claim is trivial. Suppose that B' contains at least one vertex. If $S \subseteq R$ is an odd set in G of size at most k, then $S \cup \{b\}$ is an odd set in H which contains at most $k + 1$ vertices. Assume now that H has an odd set

S of size at most k'. It is easy to see that $b \in S$, as it is the only neighbor of f. Suppose that $a \in S$. But in this case $c_1, \ldots, c_k \in S$, and S contains at least $k+2$ vertices. This contradiction proves that $a \notin S$, and therefore $c_1, \ldots, c_k \notin S$. It remains to note that $S \setminus \{b\}$ is an odd set in G of size at most k.

For the proof of the second claim, we again reduce from Odd Set of Size at most k, and the reduction uses same ideas. Let here $B' \subseteq B$ be the set of vertices of odd degree. We construct the graph H (see 9.2 b)) in a similar way as it was done above. The only difference is that for the case $B' \neq \emptyset$, vertex b is replaced by two vertices b_1 and b_2 with same neighborhoods. Parameter k' is defined as before. We prove that G has an odd set of size at most k if and only if H has an odd set of size at most k'.

If $B' = \emptyset$, then the claim is trivial. Suppose that B' contains at least one vertex. If $S \subseteq R$ is an odd set in G of size at most k, then $S \cup \{b_1\}$ is an odd set in H which contains at most $k+1$ vertices. Assume now that H has an odd set S of size at most k'. It is easy to see that either $b_1 \in S, b_2 \notin S$ or $b_1 \notin S, b_2 \in S$. Suppose that $a \in S$. But in this case $c_1, \ldots, c_k \in S$, and S contains at least $k+2$ vertices. This contradiction proves that $a \notin S$, and therefore $c_1, \ldots, c_k \notin S$. It remains to note that $S \setminus \{b_1, b_2\}$ is an odd set in G of size at most k. □

To complete the proof of W[1]-hardness of Even Set of Size at least $r-k$ it is enough to observe that if all blue vertices have odd degrees then S is an odd set of size at most k if and only if $R \setminus S$ is an even set of size at least $r-k$. The proof of W[1]-hardness for the Even Set of Size $r-k$ problem is similar, we reduce from the exact variant of the Odd Set of Size k problem.

For the proof of W[1]-hardness of Odd Set of Size at least $r-k$, it is sufficient to notice that if all blue vertices have even degrees, then S is an odd set of size at most k if and only if $R \setminus S$ is an odd set of size at least $r-k$. The proof of W[1]-hardness for the Odd Set of Size $r-k$ problem is the same, we only reduce from the exact variant of the Odd Set of Size at most k problem. □

9.4.2 Complexity of the (EVEN|ODD)-Domination Problems

The main result of this section is the W[1]-hardness of the (EVEN|ODD)-domination problems under the dual parameterization. Note that, in contrast to the previous subsection, these results are for general graphs.

Theorem 9.15. *Let $\sigma, \rho \in \{\text{EVEN}, \text{ODD}\}$. Then the (σ, ρ)-Dominating Set of Size $n-k$ and (σ, ρ)-Dominating Set of Size at least $n-k$ problems are W[1]-hard.*

Proof. We prove this theorem for the (σ, ρ)-Dominating Set of Size at least $n-k$ problem. The proof for the (σ, ρ)-Dominating Set of Size $n-k$ is done by similar arguments. We consider several cases.

1. $\sigma = \rho = \text{EVEN}$

We use the following lemma:

Lemma 9.16. Even Set of Size at least $r-k$ *remains W[1]-hard if all red vertices have even degrees.*

Proof. We reduce from the Even Set of Size at least $r-k$ problem by replacing each blue vertex by two vertices with the same neighborhoods. Trivially $S \subseteq R$ is an even set in the obtained graph if and only if it is an even set in the original graph. □

If all red vertices have even degrees then $S \subseteq R$ is an even set if and only if $S \cup B$ is an $(\text{EVEN}, \text{EVEN})$-dominating set. It follows immediately that G has an even set of size at least $r-k$ if and only if G has a (σ, ρ)-dominating set of size at least $n-k$ for $\sigma = \rho = \text{EVEN}$.

For the exact variant of the problem it is necessary to force all vertices of B (more precisely all vertices of $V \setminus R$) to be in any $(\text{EVEN}, \text{EVEN})$-dominating set of size $n-k$. This can be done by adding to each vertex $b \in B$ an adjacent clique with an even number of at least $k+1$ vertices. Any vertex of the clique not included in the dominating set would have one neighbor less than any included vertex of the clique, which is impossible. Since at least one vertex of the clique is included, whole the clique must be included in any $(\text{EVEN}, \text{EVEN})$-dominating set of size $n-k$ and hence b must be included as well, as each of the vertices of the clique has an odd number of neighbors inside it. Note also that this way b gains an even number of neighbors inside S.

2. $\sigma = \rho = \text{ODD}$

Lemma 9.17. Odd Set of Size at least $r-k$ *remains W[1]-hard if all red vertices have odd degrees.*

Proof. We reduce from the Odd Set of Size at least $r-k$ problem. Consider two copies of the instance of this problem. Denote by $u_1, \ldots, u_r$ the red vertices of the first graph, and by $v_1, \ldots, v_r$ the red vertices of the second graph. Assume that vertices $u_1, \ldots, u_s$ (vertices $v_1, \ldots, v_s$ correspondingly) have odd degrees, and the other red vertices have even degrees. We introduce one additional red vertex w. For each $i \in \{1, \ldots, s\}$, two blue vertices $x_{i,1}$ and $x_{i,2}$ are added and joined by edges with u_i, v_i and w. For each $i \in \{s+1, \ldots, r\}$, we add one blue vertex $x_{i,1}$ and join it with u_i, v_i and w. If $r-s$ is even, then one blue vertex y_1 is introduced and joined with w, and otherwise two blue vertices y_1, y_2 are added and joined with w. Denote the obtained graph by H (see Fig. 9.3). Clearly all red vertices of H have odd degrees. Let $k' = 2k$. Now we prove that the original graph G has an odd set of size at least $r-k$ if and only if H has an odd set of size at least $2r+1-k'$.

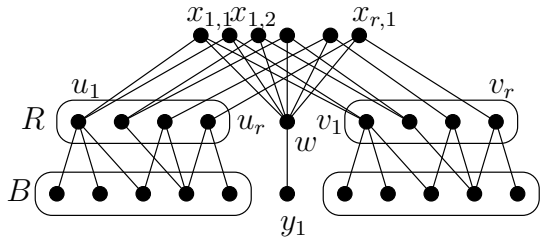


Figure 9.3: Construction of H in Lemma 9.17.

Suppose that G has an odd set S of size at least $r-k$. Denote by S_1 the copy of this set for the first copy of G, and by S_2 the odd set for the second copy. It can be easily checked that $S_1 \cup S_2 \cup \{w\}$ is an odd set in H of size at least $2r-2k+1$. Assume now that S is an odd set in H of size at least $2r+1-k'$. Note that $w \in S$ because of the presence of the vertex y_1. Let $S_1 = S \cap \{u_1, \ldots, u_r\}$ and $S_2 = S \cap \{v_1, \ldots, v_r\}$. We claim that $u_i \in S_1$ if and only if $v_i \in S_2$. Suppose that $u_i \in S_1$. Since $x_{i,1}$ has three red neighbors u_i, v_i, w and $w, u_i \in S$, $v_i \in S$ also. It remains to note that S_1 (or S_2) is an odd set of size at least $r-k$ in G. □

It is easy to see that if all red vertices in the Red/Blue bipartite graph G have odd degrees then $S \subseteq R$ is an odd set if and only if $S \cup B$ is an (ODD, ODD)-dominating set in G. Correspondingly G has an odd set of size at least $r-k$ if and only if G has a (σ, ρ)-dominating set of size at least $n-k$ for $\sigma = \rho = \text{ODD}$.

For the exact variant it is again necessary to force all vertices of B to be in any (ODD, ODD)-dominating set of size $n-k$. This is done in similar way as in the case 1 (the case $\sigma = \rho = \text{EVEN}$) by adding to each vertex $b \in B$ two cliques adjacent with b each having an odd number of at least $k+1$ vertices.

3. $\sigma = \text{ODD}$ and $\rho = \text{EVEN}$

We reduce from $(\text{EVEN}, \text{EVEN})$-Dominating Set of Size at least $n-k$. Consider two copies of the instance of this problem. Denote by $u_1, \ldots, u_n$ the vertices of the first copy of the graph G, and by $v_1, \ldots, v_n$ the vertices of the second copy. For each $i \in \{1, \ldots, n\}$, vertices u_i and v_i are connected by an edge, two copies of stars $K_{1,2k+1}$ denoted by $T_{i,1}$ and $T_{i,2}$ are introduced, and central vertices of these stars are joined with u_i and v_i. Denote the obtained graph by H (see Fig. 9.4). It is easy to see that H has $n' = 2n(2k+3)$ vertices. Let $k' = 2k$. We claim that G has an $(\text{EVEN}, \text{EVEN})$-dominating set of size at least $n-k$ if and only if H has an $(\text{ODD}, \text{EVEN})$-dominating set of size at least $n'-k'$.

Suppose that S is an $(\text{EVEN}, \text{EVEN})$-dominating set of size at least $n-k$ in G. Let S_1 be the set of vertices of S in the first copy of G, and correspondingly let S_2 be this set in the second copy. It can be straightforwardly checked that

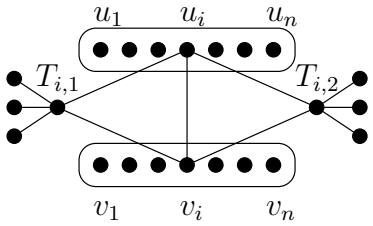


Figure 9.4: Construction of H for the case $\sigma = \text{ODD}$ and $\rho = \text{EVEN}$

$S' = S_1 \cup S_2 \cup \bigcup_{i=1}^{n}(V(T_{i,1}) \cup V(T_{i,2}))$ is an $(\text{ODD}, \text{EVEN})$-dominating set of size at least $n' - k'$ in H.

Assume now that S' is an $(\text{ODD}, \text{EVEN})$-dominating set of size at least $n'-k'$ in H. Consider some star $T_{i,j}$. Since this star has $2k+1$ leaves and $n'-|S'| \leq 2k$, at least one leaf of $T_{i,j}$ is included in S'. Thus the central vertex of the star is in S' as $\sigma = \text{ODD}$, and hence every leaf is in S' as $\rho = \text{EVEN}$. Therefore all vertices of the star are included in S'. It means that $\bigcup_{i=1}^{n}(V(T_{i,1}) \cup V(T_{i,2})) \subseteq S'$. Since the central vertex of each star has odd degree, it has an even number of neighbors in $\{u_i, v_i\}$, i.e. either $u_i, v_i \in S'$ or $u_i, v_i \notin S'$. Hence each vertex $u_i \in S'$ if and only if $v_i \in S'$. It remains to note that $S = \{u_1, \ldots, u_n\} \cap S'$ is an $(\text{EVEN}, \text{EVEN})$-dominating set in G of size at least $n - k$.

4. $\sigma = \text{EVEN}$ and $\rho = \text{ODD}$

We reduce from (ODD, ODD)-Dominating Set of Size at least $n - k$. Similarly as in the case 3, consider two copies of the instance of this problem. Denote by $u_1, \ldots, u_n$ the vertices of the first copy of the graph G, and by $v_1, \ldots, v_n$ the vertices of the second copy. For each $i \in \{1, \ldots, n\}$, vertices u_i and v_i are joined by an edge, and then $2k + 2$ vertices $x_{i,1}, \ldots, x_{i,2k+2}$ are introduced and joined with u_i and v_i. Denote the obtained graph by H (see Fig. 9.5). This graph has $n' = 2n(k + 2)$ vertices. Let $k' = 2k$. We prove that G has an (ODD, ODD)-dominating set of size at least $n - k$ if and only if H has an $(\text{EVEN}, \text{ODD})$-dominating set of size at least $n' - k'$.

Suppose that S is an (ODD, ODD)-dominating set of size at least $n - k$ in G. Let S_1 be the set of vertices of S in the first copy of G, and correspondingly let S_2 be this set in the second copy. It is easy to see that $S' = S_1 \cup S_2 \cup \bigcup_{i=1}^{n}\{x_{i,1}, \ldots, x_{i,2k+2}\}$ is an $(\text{EVEN}, \text{ODD})$-dominating set of size at least $n' - k'$ in H.

Assume now that S' is an $(\text{EVEN}, \text{ODD})$-dominating set of size at least $n'-k'$ in H. For each $i \in \{1, \ldots, n\}$, at least one vertex $x_{i,j}$ is in S', since otherwise there are at least $k' + 2$ vertices that are not included in S'. Therefore either

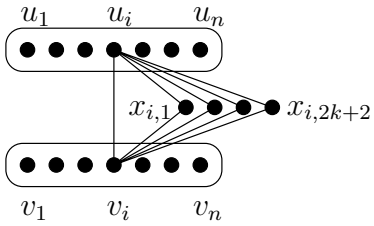


Figure 9.5: Construction of H for the case $\sigma = \text{EVEN}$ and $\rho = \text{ODD}$

$u_i, v_i \in S'$ or $u_i, v_i \notin S'$ since $\sigma = \text{EVEN}$. Hence all vertices $x_{i,1}, \ldots, x_{i,2k+2}$ are in S' since otherwise $x_{i,j} \notin S'$ would have an even number of neighbors in S' contrary to $\rho = \text{ODD}$. Also for each vertex u_i, $u_i \in S'$ if and only if $v_i \in S'$. Note now that $S = \{u_1, \ldots, u_n\} \cap S'$ is an (ODD, ODD)-dominating set in G of size at least $n - k$. □

9.5 Complexity of (σ, ρ)-Dominating Set of Size (at most) k for Sparse Graphs

We have already mentioned in Section 6.2 that many problems which are difficult for general graphs can be solved efficiently for sparse graphs. This is also the case of (σ, ρ)-Dominating Set.

If both σ and ρ are either finite or co-finite one can in linear time decide the existence of a (σ, ρ)-dominating set or even find the minimum and maximum cardinality of such a set on graphs of bounded treewidth [Tel94b, TP97, vRBR09], graphs of bounded branchwidth or clique-width [BvLvRV10] or even graphs of bounded boolean-width [ABXR$^+$10]. Actually, one can show that on graphs of bounded clique-width also by an argument somewhat similar to one bellow, using Theorem 6.4 (proved by Courcelle, Makowsky, and Rotics [CMR00]), but the mentioned specific results yield more practical running times. By way of contrast, when σ is a set with arbitrary large gaps between two consecutive elements and ρ is cofinite, then deciding the existence of (σ, ρ)-dominating set is already W[1]-hard [Cha10].

We prove that the existence of (σ, ρ)-dominating set of a particular size can be efficiently decided on much more general class of sparse graphs, namely on nowhere-dense classes of graphs and classes of graphs with bounded expansion. This result is a corollary of Theorems 6.5 and 6.6 (established in [DK09]).

Theorem 9.18. *Let σ and ρ be recursive sets of non-negative integers. Then (σ, ρ)-Dominating Set of Size (at most) k is FPT (with parameter k) on classes of graphs of bounded expansion and nowhere dense graph classes.*

Proof. Using Theorems 6.5 and 6.6, it suffices to provide for each k the FOL formula for "there is a (σ, ρ)-dominating set of size (at most) k". First note that a set of size (at most) k is (σ, ρ)-dominating if and only if it is (σ_k, ρ_k)-dominating, where $\sigma_k = \sigma \cap \{0, \ldots, k\}$ and $\rho_k = \rho \cap \{0, \ldots, k\}$, as no vertex can have more than k neighbors in a set of size (at most) k. Since both σ and ρ are recursive, sets σ_k and ρ_k can be computed in time $a(k)$ for some $a : \mathbb{N} \to \mathbb{N}$ solely depending on k.

Now we will gradually build a vocabulary, which will at the end allow us to formulate the desired formula, deciding, whether vertices $x_1, \ldots, x_k$ form a (σ_k, ρ_k)-dominating set of size (exactly) k. The "at most" k formula can be then easily obtained as a conjunction of formulas for sizes 0 to k. In what follows, $1 \leq r \leq k$, and formulas (except for those stated) also contain free variables $x_1, \ldots, x_k$.

$$
\begin{aligned}
\text{selected}(x) &= \bigvee_{i=1}^{k} (x = x_i) \\
\text{at\_least\_}r\text{\_sel\_neighs}(v) &= \exists y_1 \exists y_2 \ldots \exists y_r \Bigg(\bigwedge_{1 \leq i < j \leq r} \neg(y_i = y_j) \Bigg) \wedge \\
&\quad \wedge \Bigg(\bigwedge_{i=1}^{r} \text{selected}(y_i) \wedge \text{adj}(v, y_i) \Bigg) \\
\text{exact\_}r\text{\_sel\_neighs}(v) &= (\text{at\_least\_}r\text{\_sel\_neighs}(v)) \wedge \\
&\quad \wedge \neg(\text{at\_least\_}(r+1)\text{\_sel\_neighs}(v)) \\
\text{exact\_0\_sel\_neighs}(v) &= \neg(\text{at\_least\_1\_sel\_neighs}(v)) \\
\text{is\_satisfied}(v) &= \Big(\text{selected(v)} \wedge \bigvee_{r \in \sigma_k} (\text{exact\_}r\text{\_sel\_neighs}(v))\Big) \vee \\
&\quad \vee \Big(\neg(\text{selected}(v)) \wedge \bigvee_{r \in \rho_k} (\text{exact\_}r\text{\_sel\_neighs}(v))\Big)
\end{aligned}
$$

Now the desired formula can be expressed as

$$
\begin{aligned}
\exists(\sigma, \rho)\text{-dom\_set\_of\_size\_}k &= \exists x_1 \exists x_2 \ldots \exists x_k \Bigg(\bigwedge_{1 \leq i < j \leq k} \neg(x_i = x_j) \Bigg) \wedge \\
&\quad \wedge \forall v(\text{is\_satisfied}(v))
\end{aligned}
$$

It is easy to see, that the formula can be constructed in a time solely dependent on k and, hence, the result follows as a corollary of Theorems 6.5 and 6.6. □

9.6 Conclusion

In this chapter we studied the parameterized complexity of the (σ, ρ)-domination problems. Our results give more or less general picture for finite sets σ and

ρ. Still, it would be interesting to extend these results for (σ, ρ)-Dominating Set of Size k in the case $0 \in \rho$. We suppose that it would be a challenging task to investigate the case of (possibly) infinite sets. We presented some partial results in which σ or ρ can be co-finite, but we leave open the question about the parameterized complexity of (σ, ρ)-Dominating Set of Size (at most) k. Table 9.1 indicates that we can expect that these problems are W[1] or W[2]-hard for majority of sets.

Another direction of the research is to consider the parameterized complexity for different graphs classes. Particularly, we proved that the (σ, ρ)-domination problems are FPT for graphs of bounded expansion and nowhere dense graph classes when parameterized by the solution size. It is known that some domination problems are FPT for the more general class of *degenerate* graphs (see e.g. [AG07, KC00]). These results can be easily generalized for (σ, ρ)-domination problems for some special sets σ and ρ. It is an interesting open problem whether the results of Theorem 9.18 can be extended to degenerate graphs.

Finally, it is an interesting question, whether the FPT results of Sections 9.3 and 9.5 can be further strengthened to obtain polynomial kernels for such wide classes of σ and ρ. Such results are known for many problems that form special cases of the problems we studied in these sections. But it seems hard to develop a kernelization for the whole classes of σ and ρ at once, since, for example, the general framework developed by Fomin et al. in [FLST10] does not apply already to the Independent Dominating Set.

Chapter 10

Equitable Partitions

10.1 Introduction to Equitable Partitions

Splitting a set of objects to parts of comparable sizes is a natural problem of the everyday life. If the sizes of the partition classes differ by at most one, then such a partition is called equitable. Graph theory can be used to represent various relations among the objects to be respected, which could be done in various ways. In this chapter, based on [EFG+09], we focus on two probably most natural conditions. In the first studied problem, Equitable Connected Partition, each partition is required to induce connected subgraph of the input graph, whereas in Equitable Coloring the partitions should induce independent sets, i.e., it is a coloring. The sizes of the partitions should be as close as possible — the partition must be equitable.

Formally the problems studied are as follows:

Equitable Connected Partition (ECP)
Input: A graph $G = (V, E)$ and a positive integer $r \in \mathbb{N}$.
Question: Is there an equitable partition of V into r classes $V_1, V_2, \ldots, V_r$, such that each class of the partition induces a connected subgraph?

Equitable Coloring (EC)
Input: A graph $G = (V, E)$ and a positive integer $r \in \mathbb{N}$.
Question: Is there an equitable partition $V_1, V_2, \ldots, V_r$ of the vertex set V such that each partition induces an independent set?

In computer science these problems arise in various areas, particularly in the area of load balancing and scheduling. For example, the connectivity requirements can be used to model some problems with paging and overlaying in design of operating systems [DCB74]. The ECP problem also arises in computational social choice in the subject of *redistricting* [Alt07]. Often the task of diving graph into connected components of prescribed size was studied, sometimes with further restrictions [KH78, PS81]. The problems studied by Ito et al. in [IGZN07, IZN06]

in this way are the closest to our setting. As shown in [Gyö76, Lov77] it is always possible to equitably partition an r-connected graph into r connected components. On the other hand, ECP is known to be NP-complete, even for planar graphs or for fixed $r \geq 2$ [DF85, GJ79].

Equitable Coloring was introduced by Meyer [Mey73]. Hajnal and Szemerédi [HS70] proved (in a different setting) that a graph with maximum degree at most d has an equitable $(d+1)$-coloring. Kierstead et al. [KKMS10] presented an $O(dn^2)$-algorithm to find such a coloring. Furthermore, Kostochka and Nakprasit [KN03] showed that an n-vertex d-degenerate graph of maximum degree Δ has an equitable r-coloring for any $r \geq \max\{62d, 31dn/(n-\Delta+1)\}$.

As Graph Coloring can be trivially reduced to Equitable Coloring by adding sufficiently large independent set, Equitable Coloring is para-NP-hard with respect to r even for planar graphs. Bodlaender and Jansen [BJ95] showed that it is also NP-hard when restricted to cographs, bipartite graphs or interval graphs. It follows that it is NP-hard on graphs of bounded clique-width. On the other hand, polynomial time algorithms are known for trees [CL94] and split graphs [CKL95].

Since the above results disqualify the only natural parameter r from being successful for the problems from the parameterized perspective, we examine the problems with respect to several structural parameters, possibly combined with r. Namely, we consider parameterizations based on various combinations of:

- the treewidth of the input graph $tw(G)$,
- the pathwidth of the input graph $pw(G)$,
- the feedback vertex set number of the input graph $fvs(G)$,
- the vertex cover number of the input graph $vc(G)$,
- the max leaf number of the input graph $ml(G)$, and
- the number of partitions r.

It is widely believed, that almost every natural hard problem can be solved efficiently on graphs of bounded treewidth. For our problems this is only true up to some extend—both problems are in XP when parameterized by the treewidth. This was proved by Bodlaender and Fomin [BF05] for EC and for ECP this follows from the results of Ito et al. [IZN06].

On the other hand, it was shown in [FFL$^+$07] that Equitable Coloring is W[1]-hard when parameterized by $tw(G)$ and r combined. There it is first proved that a problem, in which we are given list of allowed colors for each vertex and each color should appear given number of times, is W[1]-hard on forests of depth at most 3 parameterized by the total number of colors in the instance. Then, by adding a lot of independent vertices each with single-color list, it is

observed there, that the problem remains intractable if the prescribed sizes of the color classes are the same. Finally, the problem is reduced to the standard Equitable Coloring by adding a clique, so that the vertices of the cligue one-to-one correspond to the colors, and enforcing the list by connecting each vertex to the vertices of the clique that correspond to the colors the vertex is not allowed to use. It is not hard to see that the instance constructed in this way in [FFL$^+$07] has not only low treewidth and low number of colors, but also low pathwidth and low feedback vertex set number as the removal of the clique leaves a forest of bounded depth. It follows that EC is also W[1]-hard with respect to the number of partitions, the pathwidth and the feedback vertex set number combined. More recently, it was shown in [FGK09a] that EC is FPT when parameterized by $vc(G)$.

We prove in Section 10.2 that ECP is also W[1]-hard when parameterized by $pw(G)$, $fvs(G)$, and r combined (the hardness with respect to the treewidth follows). We further show there that this result holds true even for planar graphs. On the positive side, we show that ECP becomes fixed-parameter tractable when parameterized by $vc(G)$, or by $ml(G)$.

Section 10.3 is devoted to the Equitable Coloring problem, and we complement there the known results by showing that the problem is FPT when parameterized by $ml(G)$.

In the rest of the chapter, we will denote by $l := (n \mod r)$ the number of partition classes whose size is larger by one than the size of the other classes, i.e., we have $r - l$ classes of size $s := \lfloor n/r \rfloor$ and l classes of size $s + 1$.

10.2 Equitable Connected Partition

We first look on the Equitable Connected Partition. In Subsection 10.2.1 it is shown to be W[1]-hard with respect to the combination of r, $tw(G)$, $fvs(G)$, and $pw(G)$. The result is strengthened in Subsection 10.2.2 also to planar graphs. Complementary, we show in Subsections 10.2.3 and 10.2.4 that the problem is fixed parameter tractable with respect to $vc(G)$ and with respect to $ml(G)$, respectively.

10.2.1 Hardness with Respect to the Treewidth

This subsection is devoted to proving the following theorem:

Theorem 10.1. Equitable Connected Partition *is W[1]-hard with respect to the pathwidth $pw(G)$, the minimum size of a feedback vertex set $fvs(G)$ and the number of partition classes r combined.*

Proof. We provide a parameterized reduction from Multicolored Clique (MCC) which is W[1]-complete (Theorem 7.11). Recall that in MCC we are given an undirected graph that is properly colored by k colors and the question

is whether there is a size-k clique in this graph consisting of exactly one vertex from each color class. The parameter is k.

A basic building block of our construction is an *anchor*. It is a vertex (the *root* of the anchor) with many neighbors of degree one (see Fig. 10.1). As the prescribed class size will be much greater than one, the pendant degree one vertices must belong to the same partition class as the root of the anchor. The number of degree one vertices of an anchor will be chosen so that two anchors cannot belong to the same class. We create exactly as many anchors as the specified number of classes r of the equitable partition; thus there must be exactly one anchor in each class. The situation can be viewed intuitively as if each class is "started" with one single anchor and then some more vertices are to be added later. The number of vertices that need to be added to the class started by a particular anchor will differ for different anchors and is forced by the number of pendant vertices that the anchor is "missing" relative to the prescribed class size. Anchors will be denoted by uppercase letters and by connecting something to an anchor we mean connecting it to the root of the anchor.

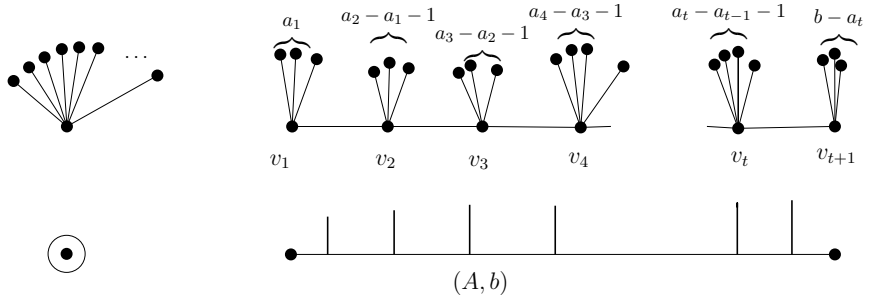


Figure 10.1: Basic building blocks of our construction: An anchor (left), an (A, b)-choice (right) and the way they are depicted in further figures (bellow).

We interconnect the anchors using a building block gadget called a *choice*. If $A = \{a_1, \ldots, a_t\}$, where $0 \leq a_1 < a_2 < a_3 < \ldots < a_t$ is a set of integers and $b \geq a_t$, then an (A, b)-*choice* is a path with $t+1$ vertices $v_1, \ldots, v_{t+1}$, where each vertex of the path can have some degree-one vertices pendant on it (see Fig. 10.1). In particular, vertex v_1 has a_1 degree one vertices pendant on it, vertex v_{t+1} has $b - a_t$ pendant vertices, and for every $i \in \{2, \ldots, t\}$, the vertex v_i has $a_i - a_{i-1} - 1$ pendant vertices. Observe again that the pendant vertices must always fall into the same class as their unique neighbor.

Now if an anchor X is connected to an anchor Y by an (A, b)-choice (which is done by simply identifying the vertices v_1 and v_{t+1} with the roots of the anchors X and Y, respectively), then there is an i, with $1 \leq i \leq t$, such that the vertices $v_1, \ldots, v_i$ (and their pendant vertices) fall into the partition class of X, while the vertices $v_{i+1}, \ldots, v_{t+1}$ fall into the class of Y. The vertices v_1 and v_{t+1} are identified with the respective anchors, and hence we do not count them. Thus, the number of vertices from this choice that fall into the class of X

is $a_1 + \sum_{j=2}^{i}((a_j - a_{j-1} - 1) + 1) = a_i \in A$, while the number of vertices falling into the class of Y is $b - a_t + \sum_{j=i+1}^{t}((a_j - a_{j-1} - 1) + 1) = b - a_i$. Note that the vertices pendant on v_1 and v_{t+1} are in fact pendant on the roots of the appropriate anchors, but we still consider them as a part of this choice.

The construction is based on sending "signals" between anchors. Let A and B be two anchors connected by a choice. The vertices of the choice have to fall into the two partition classes corresponding to A and B. The more vertices that fall into the class of A the less vertices that will fall into the one of B. However, since the sizes of the two classes differ by at most one, the class of B must include vertices from somewhere else in the construction. This generates the signal. The choices allow us to control the signal sent.

Now suppose that a graph $G = (V, E)$, an integer k, and a coloring $c : V \to \{1, \ldots, k\}$ form an instance of MCC. Also suppose that, for each i, there are n_i vertices of color i denoted by v_p^i, where $1 \leq p \leq n_i$. Furthermore, we give each edge an integer ID, i.e., there is a bijective labeling $l : E \to \{1, \ldots, |E|\}$. Our construction has a selection gadget for each vertex color, which ensures both the selection of a vertex of this color and the selection of edges from the selected vertex to vertices of the other colors. This corresponds to the edge representation strategy as mentioned in Section 7.4. The selection gadgets are interconnected in a way that an equitable partition is only possible if the IDs of the "selected" edges match.

The selection gadget for color i is formed by $2(k-1)$ anchors N_j^i and P_j^i, where $1 \leq j \leq k$ and $j \neq i$, connected into a cycle, each having a connection outside the gadget. The vertex selection is represented by a "big" signal that the outgoing connections are unable to handle, and hence is forced to run along the cycle without a change. The selection of an edge going from the selected vertex to the vertices of color j is then done between N_j^i and P_j^i via a "small" signal that equals the label of the selected edge. This signal is then sent to the anchors N_i^j, P_i^j of the selection gadget for color j, from anchor P_j^i to N_i^j and from N_j^i to P_i^j (in opposite directions).

Now we present the selection gadget more formally. First let $Z_0 := 2|E| + 10$. The "big" signal is formed by the order of the vertex selected times the number Z_0, i.e., the possible signal states are $A_0^i := \{p \cdot Z_0 \mid 1 \leq p \leq n_i\}$. As we mentioned, the small signal is formed by the edge IDs. Between the anchors N_j^i and P_j^i both the big and the small signal is sent, and a particular small signal can only be used with an appropriate big signal. Thus, between N_j^i and P_j^i, the possible signal states are $A_j^i := \{p \cdot Z_0 + l(\{u, v_p^i\}) \mid c(u) = j \text{ and } \{u, v_p^i\} \in E\}$. To catch the order of the anchors along the cycle, we introduce the notion of *successor*. For each $j, 1 \leq j \leq k$ set $succ(j) := j + 1$ for $j \neq k$ and $j \neq (i-1)$, set $succ(k) := 1$ and $succ(i-1) := succ(i)$. Now for each $j \in \{1, \ldots, k\} \setminus \{i\}$, the anchor P_j^i is connected to the anchor $N_{succ(j)}^i$ by an $(A_0^i, n_i \cdot Z_0)$-choice, the anchor N_j^i is connected to the anchor P_j^i by an $(A_j^i, n_i \cdot Z_0 + |E|)$-choice, and the anchor P_j^i to

the anchor N_i^j, and the anchor P_i^j to the anchor N_j^i by two $(\{1,\ldots,|E|\},|E|)$-choices (see Fig. 10.2).

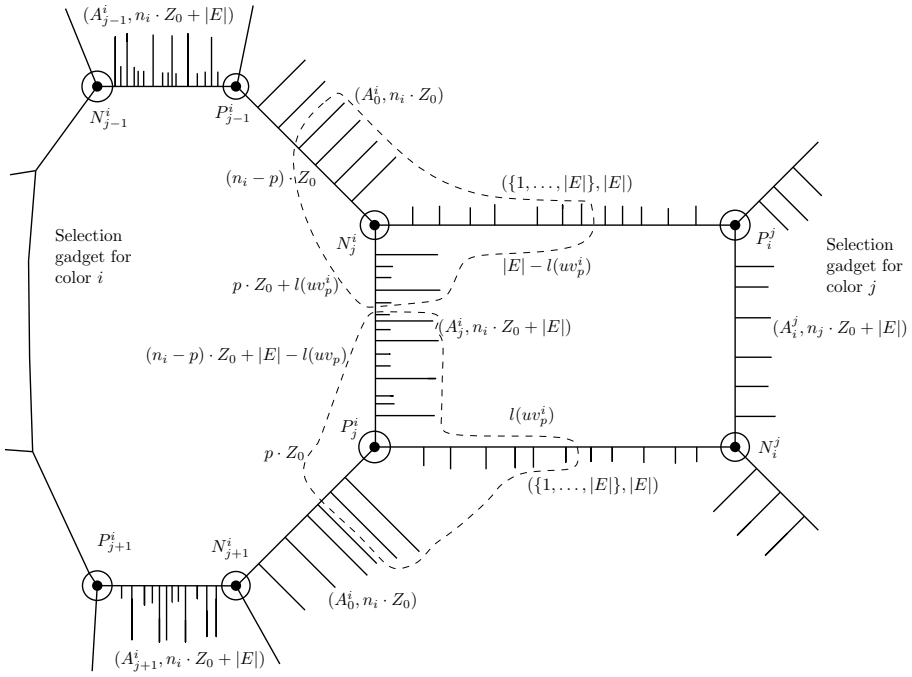


Figure 10.2: A part of the selection gadget with possible partition shown by dashed line.

Now the class sizes are set such that each of the anchors in this selection gadget needs to get $n_i \cdot Z_0 + |E|$ vertices from the choices that it is incident with. Hence, if the anchor P_j^i takes $p \cdot Z_0$ vertices from the choice connecting it to the anchor $N_{succ(j)}^i$, then the anchor $N_{succ(j)}^i$ gets $(n_i - p) \cdot Z_0$ vertices from this choice, and it must get $p \cdot Z_0 + |E|$ vertices from the two remaining choices that it is incident with. Since it can take at most $|E|$ vertices from the connection to the selection gadget for color $succ(j)$, it must take at least $p \cdot Z_0$ (but at most $p \cdot Z_0 + |E| < (p+1) \cdot Z_0$) vertices from the connection to $P_{succ(j)}^i$. Hence, it has to take $p \cdot Z_0 + l(\{u, v_p^i\})$ vertices for some $c(u) = succ(j)$ and $\{u, v_p^i\} \in E$ from this connection and $|E| - l(\{u, v_p^i\})$ vertices from the connection to the other selection gadget. Hence $P_{succ(j)}^i$ gets $(n_i - p) \cdot Z_0 + |E| - l(\{u, v_p^i\})$ vertices from the connection from $N_{succ(j)}^i$, and by a similar reasoning it is forced to take $p \cdot Z_0$ vertices from the connection to $N_{succ(succ(j))}^i$ and $l(\{u, v_p^i\})$ vertices from the connection to the other selection gadget. Thus the anchor $N_{succ(succ(j))}^i$ is again forced to select some edge incident with vertex v_p^i, etc.

The anchor N_j^i is connected to the anchor P_i^j by a $(\{1,\ldots,|E|\},|E|)$-choice, and the number of vertices it takes into its class out of this choice is $|E| - l(\{u, v_p^i\})$, where v_p^i is the vertex selected in the selection gadget for color i,

$c(u) = j$ and $\{u, v_p^i\} \in E$. The number of vertices that the anchor P_i^j takes out of this choice is $l(\{w, v_q^j\})$, where v_q^j is the vertex selected in the selection gadget for color j, $c(w) = i$ and $\{w, v_q^j\} \in E$. Since the $|E|$ vertices of the choice must be partitioned into the classes of its endpoints, it follows that $l(\{w, v_q^j\}) = l(\{u, v_p^i\})$ and $\{w, v_q^j\} = \{u, v_p^i\} = \{v_q^j, v_p^i\}$ is an edge of G. Hence a solution for the constructed graph is possible if and only if the selected vertices form a multi-colored clique in the graph G.

Now to determine the right size of the anchors, it is enough to ensure that each class is more than half full, once the starting anchor is added. The maximum demand (the number of vertices that should be added to its class except for itself and vertices pendant on it) of any anchor is less than $n \cdot Z_0 + |E|$, hence it is enough to set the desired class size to $s := (2n + 1) \cdot Z_0 = (2n + 1) \cdot (2|E| + 10)$.

The number of partition classes r is equal to the number of anchors. Since there are k selection gadgets, each containing $2(k-1)$ anchors, we have $2k(k-1)$ anchors in total. Now observe that if we delete all roots of the anchors, the resulting graph consists of paths with pendant vertices. Hence, the roots form a feedback vertex set in the graph, and the pathwidth of the graph is also bounded by the number of roots plus one. The construction can be clearly carried out in polynomial time; in particular, the graph has $r \cdot s = O(k^2 \cdot n^3)$ vertices. □

10.2.2 Hardness for Planar Graphs with Respect to the Treewidth

In the previous subsection we have shown, that ECP is hard with respect to $pw(G)$, $fvs(G)$ and r combined. In this subsection, we prove that this is true even in planar graphs:

Corollary 10.2. Equitable Connected Partition *is W[1]-hard for planar graphs with respect to the pathwidth $pw(G)$, the minimum size of a feedback vertex set $fvs(G)$ and the number of partition classes r combined.*

Proof. The graph H' constructed in Theorem 10.1 is in general not planar. But there is a drawing of this graph such that only the edges of the choices connecting two different selection gadgets cross. Moreover, we can assume that each pair of them crosses at most once, and only in the edges of their paths, not in the edges connecting the pendant vertices. We replace each such crossing one by one by a planar *crossing gadget*, such that the resulting planar graph H has a solution if and only if the graph H' does.

Suppose that in our drawing of H' the $(\{1, \ldots, |E|\}, |E|)$-choices between anchors A and B and between C and D cross. The crossing gadget is formed by four anchors R, S, X, Y such that the anchor R is connected to A, X to C, S to B, Y to D and X to Y by a $(\{1, \ldots, |E|\}, |E|)$-choice, respectively. The anchor X is connected to both R and S by $(\{z \cdot (Z_0+1) \mid 1 \leq z \leq |E|\}, |E| \cdot (Z_0+1))$-choices

and Y is connected to both R and S by $(\{z \cdot Z_0 \mid 1 \leq z \leq |E|\}, |E| \cdot Z_0)$-choices (see Fig. 10.3). The anchors R, S and Y need $|E| \cdot (Z_0 + 1)$ vertices to be added to their respective classes, while X needs $|E| \cdot (Z_0 + 2)$ vertices.

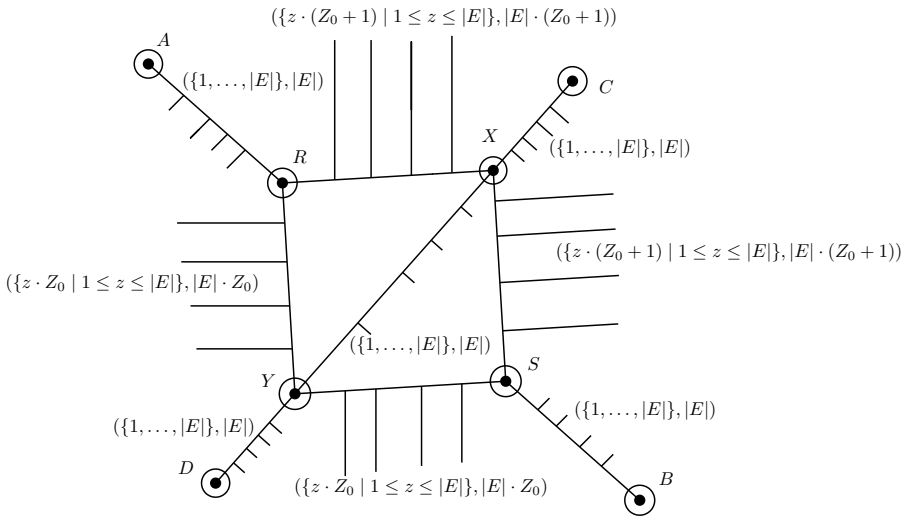


Figure 10.3: The crossing gadget.

If X gets $p \cdot (Z_0+1)$ vertices from the choice connecting it to R, then it can get between 2 and $2 \cdot |E|$ vertices from the connections to Y and C, and, hence, out of the choice connecting it to S it must take between $|E| \cdot (Z_0+2) - p \cdot (Z_0+1) - 2 < (|E| - p + 1) \cdot (Z_0+1)$ and $|E| \cdot (Z_0+2) - p \cdot (Z_0+1) - 2 \cdot |E| > (|E| - p - 1) \cdot (Z_0+1)$ vertices. Therefore it must take $(|E| - p) \cdot (Z_0 + 1)$ vertices. The anchor Y works in a similar way. Thus, if C takes q vertices from the connection to X, then X gets $|E| - q$ vertices, and takes q vertices from the connection to Y; the anchor Y does similarly, and thus D gets $|E| - q$ vertices, as if it were connected to C directly.

Counting the number of vertices inside the crossing gadget and the demands of the anchors, it follows that the number of vertices A get and the number of vertices B get must also sum up to $|E|$ as if they were connected directly.

It is easy to check, that the resulting graph H is planar, with $pw(G)$ a $fvs(H)$ bounded in terms of k. Moreover it can be constructed in polynomial time and has a solution if and only if the graph H' constructed in Theorem 10.1 has. □

Since the treewidth of a graph is never greater than its pathwidth, we immediately get (as mentioned in Section 4.4) also the following corollary:

Corollary 10.3. Equitable Connected Partition *is W[1]-hard for planar graphs with respect to the treewidth* $tw(G)$.

10.2.3 Tractability with Respect to the Vertex Cover Number

The vertex cover is more restrictive than both the pathwidth and the feedback vertex set number. Hence, it seems to be a suitable structural parameter for ECP. Indeed, we show, that ECP is FPT with respect to this parameter:

Theorem 10.4. Equitable Connected Partition *is in FPT with respect to the minimum size of a vertex cover* $vc(G)$.

Proof. Assume that we are given a vertex cover $C \subseteq V$ of size $c := vc(G)$. If not, we can compute it in time $O(1.2738^c + cn)$ by [CKX06]. Each class that contains at least 2 vertices must contain some vertex of C; otherwise, it would not be connected. Hence, if the minimum size of each class s is at least 2, then either $r \leq c$ or (G, r) is a no-instance. If $s = 1$, then we have l classes of size 2 and $r - l$ of size 1. Since a size-2 class contains two vertices connected by an edge, such a partition is in fact a matching of size l in G. The existence of such a matching can be decided in polynomial time. The case of $s = 0$ is trivial and yields a yes-instance. Hence, in what follows we can assume that $s \geq 2$ and $r \leq c$.

We search for an equitable partition such that the first l classes are the larger ones and the last $r - l$ are the smaller ones. We start by trying all possibilities of partitioning the vertices of C into r (not necessarily connected) non-empty classes $V_1^C, \ldots, V_r^C$. For each such partition, and each disconnected class V_i^C, we try the possibilities of adding at most $|V_i^C| - 1$ vertices of $V \setminus \bigcup_{i=1}^r V_i^C$ into V_i^C to make it connected. But we do not try all vertices. Instead, each vertex tried must have a different neighborhood. We try all such possibilities with different neighborhoods. It remains to distribute the remaining vertices among the classes so that the partition becomes equitable. We construct a network such that there is a flow of certain size in it if and only if the vertices can be distributed among the classes.

Let us denote by $D := V \setminus \bigcup_{i=1}^r V_i^C$ the set of vertices that are not used yet. The network consists of three intermediate layers, in addition to the source z and the target t. There are r vertices in the first layer, denoted $a_1, \ldots, a_r$. For every vertex a_i, there is an arc connecting the source z to the vertex a_i and its capacity equals the number of vertices that should still be added into the class i, i.e., $s - |V_i^C|$ if $i > l$ and $s + 1 - |V_i^C|$ if $i \leq l$. The second layer is formed by the vertices of C, and for each i, there are arcs of capacity ∞ from a_i to each vertex in $V_i^C \cap C$. The third layer is formed by vertices $b_J, J \subseteq C$, and there is an arc between $v \in C$ and b_J if and only if $v \in J$. Such arcs have also infinite capacity. Finally each b_J is connected to t by an arc with capacity equal to the number of vertices in D with neighborhood J.

The flow on arcs between the vertices of C and the vertices of type b_J directly shows how many vertices of the particular type should be put into the same class as the vertex of C. Hence it is easy to see that there is a flow of size $|D|$

in the constructed network if and only if the vertices can be distributed among the classes. Concerning the running time of the algorithm, there are at most r^c different colorings of C; for each of them we try adding at most $c-1$ vertices to the classes, each of at most 2^c types. Hence there are at most $O(2^{c^2})$ possibilities to do so. The network can be constructed in time linear in the number of edges of the original graph and the number of vertex types. The flow can be found in the time cubic in the number of vertices of the network, i.e., $O((2^c+2c+2)^3)$ [GT86]. Hence the overall running time of the algorithm is bounded by $O(2^{c^2+c\cdot\log c+4c} \cdot n^2)$. □

10.2.4 Tractability with Respect to the Max Leaf Number

In this subsection we prove that ECP is fixed-parameter tractable with respect to the maximum number of leaves in a spanning tree of the input graph G:

Theorem 10.5. Equitable Connected Partition *is in FPT with respect to the maximum number of leaves in a spanning tree* $ml(G)$.

Proof. We construct an instance of Integer Linear Programming(ILP) whose number of variables is a function of $ml(G)$ for ECP. In Section 4.4 we have mentioned that if $ml(G) = k$ then G is a subdivision of some graph H on (at most) $4k-2$ vertices [KW91] and that such a graph H can be easily found in linear time. We say that a partition class is *simple* if it contains no vertex of H. For an edge $\{u, v\}$ of H we use P_{uv} to denote the unique path in G having as endpoints u and v with internal vertices in $G \setminus H$; let $|P_{uv}|$ denote the number of its internal vertices.

We first branch on all possibilities of partitioning the vertices of H into at most $4k-2$ classes. Note that quite possibly $4k-2 < r$. We construct an ILP instance as follows. On the given branch, assume that the classes that are assigned the vertices of H, according to the partition of the branch, are $V_1, \ldots, V_p$. Recall that $s := \lfloor n/r \rfloor$. If a path P_{uv} has at least s internal vertices, then it cannot be fully contained in one class. Hence, it must be split into several parts. The first part is put into the same class as the vertex u, the last part into the same class as v and the rest is divided into several simple classes. Since the order of the simple classes on the path does not matter, we only have to know the number of classes having size s, and the number of classes having size $s+1$. Hence, for such an edge $\{u, v\}$ of H, we introduce four variables: $t_{u,uv}$ and $t_{v,uv}$ representing the number of internal vertices of the path to be placed in the same class as u and v, respectively, and a_{uv} and b_{uv} representing the number of simple classes of size s and $s+1$ on P_{uv}, respectively.

If a path P_{uv} contains less than s internal vertices, then there is no simple class on this path, and each vertex of the path is in the same class as one of the endpoints. In particular, if u and v are in the same class, then the whole path P_{uv} is in that class. If u and v are in different classes, then we introduce two variables $t_{u,uv}$ and $t_{v,uv}$ for that path, with the same meaning as in the previous

case. To simplify the equations, we denote by E_1 the set $\{\{u,v\} \in E(H) \mid |P_{uv}| < s$ and u and v lie in different classes$\}$, by E_2 the set $\{\{u,v\} \in E(H) \mid |P_{uv}| \geq s\}$, by E_3^i the set $\{\{u,v\} \in E(H) \mid |P_{uv}| < s$ and $u,v \in V_i\}$ and $E_3 := \bigcup_{i=1}^{p} E_3^i$. The class V_i is connected if and only if the graph (V_i, E_3^i) is connected. We check this before we call the procedure to solve the ILP. Finally, we introduce a $\{0,1\}$-variable c_i, for each $1 \leq i \leq p$, so that the size of V_i in the final partition will be $s + c_i$. The variables introduced are subject to the following constraints:

$$\begin{aligned} \forall \{u,v\} \in E_1 \cup E_2 &: 0 \leq t_{u,uv}, t_{v,uv}, \\ \forall \{u,v\} \in E_2 &: 0 \leq a_{uv}, b_{uv}, \\ \forall i, 1 \leq i \leq p &: 0 \leq c_i \leq 1, \end{aligned}$$

The above formalize obvious matters pertaining to our approach. The following constraints do the main work:

$$\forall \{u,v\} \in E_1 : t_{u,uv} + t_{v,uv} = |P_{uv}|, \tag{10.1}$$

$$\forall \{u,v\} \in E_2 : t_{u,uv} + t_{v,uv} + s \cdot a_{uv} + (s+1) \cdot b_{uv} = |P_{uv}|, \tag{10.2}$$

$$\forall i, 1 \leq i \leq p : \sum_{v \in V(H) \cap V_i} (1 + \sum_{\{u,v\} \in E_1 \cup E_2} t_{v,uv}) + \sum_{\{u,v\} \in E_3^i} |P_{uv}| = s + c_i \tag{10.3}$$

$$\sum_{\{u,v\} \in E_2} a_{uv} + \sum_{i=1}^{p} (1 - c_i) = r - l, \tag{10.4}$$

$$\sum_{\{u,v\} \in E_2} b_{uv} + \sum_{i=1}^{p} c_i = l. \tag{10.5}$$

Equation 10.1 ensures that the paths corresponding to the edges in E_1 are correctly divided. Equation 10.2 ensures the same for the edges in E_2. Equation 10.3 ensures that the classes containing some vertices of H have the right size, and the last two equations (10.4 and 10.5) ensure that there are the right numbers of the large and the small classes. It is easy to see that there is a solution to this ILP instance if and only if there is an equitable connected partition of the vertices of G with r classes, that extends the initial assignments made according to the branch (partition of $V(H)$) being explored. Since there are at most $4 \cdot \binom{4k}{2} + 4k \leq 32k^2$ variables, each of them used at most three times, the overall size of the instance is bounded by $O(k^2)$ and it can be solved in $O((32k^2)^{2.5 \cdot 32k^2 + o(k^2)} \cdot k^2)$ time due to the results of [FT87a] mentioned in Section 6.4. This yields a running time of $O(m \cdot 2^{160k^2 \log k + o(k^2 \log k)})$ for the whole algorithm, where $k = ml(G)$. □

10.3 Equitable Coloring

In this section we show that EQUITABLE COLORING is in FPT with respect to the maximum number $ml(G)$ of leaves in a spanning tree of G. Since a graph with bounded $ml(G)$ contains a lot of induced paths, we first examine the situation on the paths. There is a nice characterization lemma for that case:

Lemma 10.6. *Let $k \geq 2$ be an integer. Let P be a path with endpoints possibly colored by one of the colors $1, \ldots, k$. Let n be the number of uncolored vertices on the path, and for every $i \in \{1, \ldots, k\}$ let $t(i) \in \{0, 1, 2\}$ be the number of endpoints colored by color i. Then P can be properly colored by the colors $1, \ldots, k$ such that there are $n_i + t(i)$ vertices of color i if and only if $\sum_i n_i = n$ and forevery i we have $0 \leq n_i \leq \lceil \frac{1}{2}(n - t(i)) \rceil$.*

Proof. The "only if" part is easy; the proof of the "if" part is by induction on n. The cases $n = 1$ and $n = 2$ are trivial. For $n \geq 3$, let us assume that $\lceil \frac{1}{2}(n - t(1)) \rceil - n_1 \leq \lceil \frac{1}{2}(n - t(2)) \rceil - n_2 \leq \lceil \frac{1}{2}(n - t(3)) \rceil - n_3 \leq \ldots \leq \lceil \frac{1}{2}(n - t(k)) \rceil - n_k$. It is then easy to check that, for $n \geq 3$, we always have $\lceil \frac{1}{2}(n - t(3)) \rceil - n_3 > 0$. In what follows we distinguish several cases. In each of them we can color the first uncolored vertex by one of the colors $1, 2$, and color the rest using the induction hypothesis for the path starting with the newly colored vertex, $n' := n - 1$, with the sizes n'_i of the color classes and the numbers of the colored endpoints $t'(i)$ set appropriately. To do so, it is enough to show that $\lceil \frac{1}{2}(n' - t'(1)) \rceil - n'_1 \geq 0$ and $\lceil \frac{1}{2}(n' - t'(2)) \rceil - n'_2 \geq 0$, since for $i \geq 3$ we have $\lceil \frac{1}{2}(n' - t'(i)) \rceil - n'_i \geq \lceil \frac{1}{2}(n - t(i)) \rceil - 1 - n_i \geq 0$. Note also that if we use the color $i \in \{1, 2\}$ then $\lceil \frac{1}{2}(n' - t'(i)) \rceil - n'_i = \lceil \frac{1}{2}(n - 1 - (t(i) - 1)) \rceil - (n_i - 1) = \lceil \frac{1}{2}(n - t(i)) \rceil - (n_i) \geq 0$.

- If one of the endpoints has color 1, then we color its neighbor by 2. Since $n'_1 = n_1$ and $t'(1) = t(1) - 1$, the induction hypothesis applies in this case.
- If one of the endpoints has color 2, then we color its neighbor by 1. Since $n'_2 = n_2$ and $t'(2) = t(2) - 1$, the induction hypothesis applies in this case.
- In any other case we color the first uncolored vertex by color 1. Note that in this case $t(1) = t(2) = 0$ and thus if $\lceil \frac{1}{2}(n - t(2)) \rceil - n_2 = 0$ then necessarily $n_1 = n_2 = \lceil \frac{1}{2}n \rceil$ and thus n must be even. But then $\lceil \frac{1}{2}n' \rceil = \lceil \frac{1}{2}n \rceil$ and hence the induction hypothesis also applies.

□

Theorem 10.7. EQUITABLE COLORING *is in FPT with respect to the maximum number of leaves in a spanning tree $ml(G)$.*

Proof. First we show that if there are many classes, then we have a yes-instance; otherwise, we again construct an instance of ILP for EQUITABLE COLORING. It

is known that G is a subdivision of some graph H on (at most) $4k-2$ vertices, for $ml(G) = k$ [KW91]. Such a graph H can be easily found in linear time. It follows that no vertex in G has degree more than $4k-3$ and thus, by the mentioned result of Hajnal and Szemerédi [HS70], G has an equitable r-coloring for every $r \geq 4k-2$. For the rest of the proof we assume that $r \leq 4k-3$. For an edge $\{u,v\}$ of H we again use P_{uv} to denote the unique path in G having as endpoints u and v with internal vertices in $G \setminus H$. Let $|P_{uv}|$ denote the number of its internal vertices.

Now we try all the possibilities $c : V(H) \to \{1, \dots r\}$ to color the vertices of H. For each such possibility, we construct an instance of ILP, which will have a variable q^i_{uv} for each combination of color i and an edge $\{u,v\}$ of H. This variable expresses the number of the vertices of color i on the path P_{uv}. They are subject to the constraints given by Lemma 10.6 and the constraints that enforce the classes to have the right number of vertices. In the following formal description of the constraints, for a logical formula φ the expression $[\varphi]$ is 1 if φ is true and 0 otherwise. Note that these expressions as well as the ceilings only appear on the constant sides of the (in)equations.

$$\begin{aligned}
\forall \{u,v\} \in E(H), 1 \leq i \leq r \quad &: \quad 0 \leq q^i_{uv} \leq \left\lceil \frac{1}{2}(|P_{uv}| - [c(u) = i] - [c(v) = i]) \right\rceil, \\
\forall \{u,v\} \in E(H) \quad &: \quad \sum_{i=1}^{r} q^i_{uv} = |P_{uv}|, \\
\forall i, 1 \leq i \leq l \quad &: \quad \sum_{\{u,v\} \in E(H)} q^i_{uv} = s + 1 - \sum_{v \in V(H)} [c(v) = i], \\
\forall i, l+1 \leq i \leq r \quad &: \quad \sum_{\{u,v\} \in E(H)} q^i_{uv} = s - \sum_{v \in V(H)} [c(v) = i].
\end{aligned}$$

Clearly, there is a solution for Equitable Coloring if there is a solution to the ILP for one of the colorings c. Since an instance X of ILP with t variables can be solved in time $O(t^{2.5t+o(t)} \cdot |X|)$ (Theorem 6.10), the overall running time is at most $O((64k^3)^{2.5 \cdot 64k^3 + o(k^3)} poly(n))$, where the polynomial is independent of k. □

10.4 Conclusion

As you may observe, the graphs constructed in the proof of W[1]-hardness of ECP are not even 2-connected. The connectivity seems to play a crucial role when partitioning a graph into connected subgraphs. Namely, as we have mentioned, Győri [Gyö76] and independently Lovasz [Lov77] showed that an r-connected graph can be partitioned into r connected partitions of prescribed sizes, even if one vertex of each partition is given. Hence, we wonder whether ECP might

be in FPT for 3-connected planar graphs, or for 3-connected graphs of bounded treewidth.

It is also an interesting question, whether ECP is in XP or para-NP-complete with respect to the clique-width. Further it would be nice to explore, whether a planarity can be helpful in solving Equitable Coloring particularly with respect to some structural parameter such as the treewidth, as the results on d-degenerate graphs ([KN03]) can become a more powerful tool in this case.

List of Considered Problems

(σ, ρ)-Dominating Set of Size at most k
(σ, ρ)-Dominating Set of Size k
(σ, ρ)-Dominating Set of Size at least $n - k$
(σ, ρ)-Dominating Set of Size $n - k$
Input: A graph $G = (V, E)$ such that $|V| = n$ and $k \in \mathbb{N}$.
Question: Is there a set of vertices $S \subseteq V$ of size at most k (exactly k, at least $n - k$, exactly $n - k$) in which for every vertex $v \in S$, we have $|S \cap N(v)| \in \sigma$ and for every $v \notin S$, we have $|S \cap N(v)| \in \rho$?
Parameterizations Considered: solution-size or dual parameterization k
Considered on pages: 86–90, 95, 98, 101, 103–105, 107

d-Hitting Set
Input: A family $\mathcal{F}$ of sets, each with at most d elements, and $k \in \mathbb{N}$.
Question: Is there a set of at most k elements, that contains an element from (hits) every set in $\mathcal{F}$?
Parameterizations Considered: solution-size k
Considered on pages: 22, 26, 27, 30, 34

k-Leaf Out-Branching
Input: A directed graph and $k \in \mathbb{N}$.
Question: Does the directed graph contain a subgraph in which every vertex except for one has in-degree exactly 1 and k vertices has out-degree 0 (are leaves)?
Parameterizations Considered: number of leaves k
Considered on pages: 30, 31

k-Local Search for Traveling Salesperson
Input: A graph with positive weights on edges, a Hamiltonian cycle in it and $k \in \mathbb{N}$.
Question: Can we find a Hamiltonian cycle which uses at most k edges not used by the original cycle and its weight is smaller than the weight of the original one
Parameterizations Considered: locality k
Considered on pages: 19

k-Path
Input: A graph G and $k \in \mathbb{N}$.
Question: Does G contain a path of length k?
Parameterizations Considered: solution-size k
Considered on pages: 29, 40

0-Directed Steiner Network (0-DSN)
Input: A set of vertices V, a weight function $w : V \times V \to W$ ($W \subseteq N_0 \cup \{\infty\}$), l pairs of vertices $(s_1, t_1), (s_2, t_2), \ldots, (s_l, t_l)$, and a weight bound $p \in \mathbb{N}$.
Question: Is there a set of arcs $A \subseteq V \times V$ of weight $w(A) \leq p$ such that in the digraph $D := (V, A)$ for every $1 \leq i \leq l$ there is a directed path from s_i to t_i?
Parameterizations Considered: solution-size $p/(\min(W \setminus \{0\}))$, number of terminal pairs l, ratio $r = (\max W)/(\min(W \setminus \{0\}))$
Considered on pages: 64–67, 79–83

0-Directed Steiner Tree (0-DST)
Input: A set of vertices V, a weight function $w : V \times V \to W$ ($W \subseteq N_0 \cup \{\infty\}$), a set $T \subseteq V$ of terminals such that $|T| = l$, a root $s \in V$, and a weight bound $p \in \mathbb{N}$.
Question: Is there a set of arcs $A \subseteq V \times V$ of weight $w(A) \leq p$ such that in the digraph $D := (V, A)$ for every $t \in T$ there is a directed path from s to t?
Parameterizations Considered: solution-size $p/(\min(W \setminus \{0\}))$, number of terminals l, ratio $r = (\max W)/(\min(W \setminus \{0\})$
Considered on pages: 64–70

0-Strongly Connected Steiner Subgraph (0-SCSS)
Input: A set of vertices V, a weight function $w : V \times V \to W$ ($W \subseteq N_0 \cup \{\infty\}$), a set $S \subseteq V$ of terminals such that $|S| = l$, and a weight bound $p \in \mathbb{N}$.
Question: Is there a set of arcs $A \subseteq V \times V$ of weight $w(A) \leq p$ such that in the digraph $D := (V, A)$ for every $s, t \in S$ there is a directed path from s to t?
Parameterizations Considered: solution-size $p/(\min(W \setminus \{0\}))$, number of terminals l, ratio $r = (\max W)/(\min(W \setminus \{0\}))$
Considered on pages: 64, 66, 67, 69, 70, 74–76, 79, 80, 82, 83

At most α-Satisfiability
Input: A Boolean formula φ in conjunctive normal form, without negations and $k \in \mathbb{N}$.
Question: Does φ allow a satisfying truth assignment of weight at most k such that each clause of φ contains at most α variables which evaluate to *true*?
Parameterizations Considered: weight of the assignment k
Considered on pages: 89, 90, 93

Bounded-Degree Deletion

Input: A graph G, $d \in \mathbb{N}$ and $k \in \mathbb{N}$.
Question: Is there a set of at most k vertices such that its deletion turns G into a graph with maximum degree at most d?
Parameterizations Considered: solution-size k
Considered on pages: 30

Clique
Input: A graph and $k \in \mathbb{N}$.
Question: Does the graph have complete subgraph with at least k vertices?
Parameterizations Considered: solution-size k
Considered on pages: 52, 56, 58, 59, 63

Conservative Coloring
Input: A graph which have all vertices except for one properly colored by k colors and $c \in \mathbb{N}$.
Question: a proper coloring of that graph with k colors which differ from the original one on at most c places?
Parameterizations Considered: conservativeness c
Considered on pages: 19

Directed Steiner Network (DSN)
Input: A set of vertices V, a weight function $w : V \times V \to W$ ($W \subseteq N \cup \{\infty\}$), l pairs of vertices $(s_1, t_1), (s_2, t_2), \ldots, (s_l, t_l)$, and a weight bound $p \in \mathbb{N}$.
Question: Is there a set of arcs $A \subseteq V \times V$ of weight $w(A) \leq p$ such that in the digraph $D := (V, A)$ for every $1 \leq i \leq l$ there is a directed path from s_i to t_i?
Parameterizations Considered: solution-size $p/(\min W)$, number of terminal pairs l, ratio $r = (\max W)/(\min W)$
Considered on pages: 64, 67, 79, 82, 83

Directed Steiner Tree (DST)
Input: A set of vertices V, a weight function $w : V \times V \to W$ ($W \subseteq N \cup \{\infty\}$), a set $T \subseteq V$ of terminals such that $|S| = l$, a root $s \in V$, and a weight bound $p \in \mathbb{N}$.
Question: Is there a set of arcs $A \subseteq V \times V$ of weight $w(A) \leq p$ such that in the digraph $D := (V, A)$ for every $t \in T$ there is a directed path from s to t?
Parameterizations Considered: solution-size $p/(\min W)$, number of terminals l, ratio $r = (\max W)/(\min W)$
Considered on pages: 63, 64, 66–69

Dominating Set
Input: A graph G and $k \in \mathbb{N}$.
Question: Is there a set of at most k vertices, such that every vertex of G is either a part of it or has a neighbor in it?
Parameterizations Considered: solution-size k, treewidth $tw(G)$

Considered on pages: 31, 47, 48, 52, 58, 59, 85–87

Equitable Coloring (EC)
Input: A graph G and $r \in N$.
Question: Is there an r-coloring of the given graph with the sizes of any two color classes differing by at most one?
Parameterizations Considered: number of partitions r, treewidth $tw(G)$, pathwidth $pw(G)$, feedback vertex set number $fvs(G)$, vertex cover number $vc(G)$, max leaf number $ml(G)$
Considered on pages: 41, 42, 109–111, 120–122

Equitable Connected Partition (ECP)
Input: A graph $G = (V, E)$ and $r \in \mathbb{N}$.
Question: Is there an equitable partition of V into r classes $V_1, V_2, \ldots, V_r$, such that each class of the partition induces a connected subgraph?
Parameterizations Considered: number of partitions r, treewidth $tw(G)$, pathwidth $pw(G)$, feedback vertex set number $fvs(G)$, vertex cover number $vc(G)$, max leaf number $ml(G)$
Considered on pages: 109–111, 115–118, 121, 122

Even Set of Size at least $r - k$, Even Set of Size $r - k$
Input: A bipartite graph $G = (R, B, E)$ such that $|R| = r$ and $k \in \mathbb{N}$.
Question: Is there a set of vertices $S \subseteq R$ of size at least $r - k$ (exactly $r - k$) in which every vertex of B has an even number of neighbors?
Parameterizations Considered: dual parameterization k
Considered on pages: 88, 100–102

Even Set of Size at most k, Even Set of Size k
(also known as Even Set and Exact Even Set)
Input: A bipartite graph $G = (R, B, E)$ and $k \in \mathbb{N}$.
Question: Is there a set of vertices $S \subseteq R$ of size at most k (exactly k) in which every vertex of B has an even number of neighbors?
Parameterizations Considered: solution-size k
Considered on pages: 87, 89, 99

Exact Satisfiability
Input: A Boolean formula φ in conjunctive normal form and $k \in \mathbb{N}$.
Question: Does φ allow a satisfying truth assignment of weight at most k such that each clause of φ contains exactly one literal which evaluate to *true*?
Parameterizations Considered: weight of the assignment k
Considered on pages: 89, 90

Feedback Vertex Set

Input: A graph G and $k \in \mathbb{N}$.
Question: Is there a set of at most k vertices such that its deletion turns G into a forest?
Parameterizations Considered: solution-size k
Considered on pages: 30, 46

Graph Coloring
Input: A graph G on n vertices and $k \in \mathbb{N}$.
Question: Can G be properly colored by at most k colors?
Parameterizations Considered: number of colors available k, dual parameterization $n - k$
Considered on pages: 27, 51, 110

Independent Set
Input: A graph and $k \in \mathbb{N}$.
Question: Is there a subset of at least k vertices such that no two of the vertices are connected by an edge?
Parameterizations Considered: solution-size k, treewidth $tw(G)$
Considered on pages: 37, 39, 48, 52, 58, 85–87, 97

Integer Linear Programming (ILP)
Input: A $p \times n$-matrix A and an n-vector b.
Question: Is there a vector $x \in \mathbb{N}^p$ such that $Ax \leq b$ (coordinatewise)?
Parameterizations Considered: number of variables p
Considered on pages: 41, 118, 120, 121

List Coloring
Input: A graph G, a set of colors B and a mapping $L : V(G) \to \mathcal{P}(B)$ assigning to each vertex its list of available colors
Question: Is there a proper coloring $c : V(G) \to B$ respecting the lists? This means that for every $v \in V(G)$ we have $c(v) \in L(v)$
Parameterizations Considered: vertex cover number $vc(G)$
Considered on pages: 57

Max d-Sat
Input: A propositional formula φ in form of conjunction of clauses, each formed by exactly d literals and $k \in \mathbb{N}$.
Question: Is it possible to satisfy at least $m(1 - 2^{-d}) + k$ clauses?
Parameterizations Considered: param. above tight lower bounds k
Considered on pages: 18

Max Leaf
Input: A graph and $k \in \mathbb{N}$.

Question: Is there a spanning tree of the graph with at least k leaves?
Parameterizations Considered: solution-size k
Considered on pages: 32

Multicolored Clique (MCC)
Input: A graph $G = (V, E)$, positive integer $k \in \mathbb{N}$ and a proper k-coloring $c : V \to \{1, \ldots, k\}$ of G.
Question: Is there a multicolored clique in G, that is a clique taking exactly one vertex of each color?
Parameterizations Considered: number of colors k
Considered on pages: 56, 57, 70–72, 74, 80, 111, 113

Odd Cycle Transversal
Input: A graph G and $k \in \mathbb{N}$.
Question: Is there a set of at most k vertices such that its deletion turns G into a bipartite graph?
Parameterizations Considered: solution-size k
Considered on pages: 42

Odd Set of Size at least $r - k$, Odd Set of Size $r - k$
Input: A bipartite graph $G = (R, B, E)$ such that $|R| = r$ and $k \in \mathbb{N}$.
Question: Is there a set of vertices $S \subseteq R$ of size at least $r - k$ (exactly $r - k$) in which every vertex of B has an odd number of neighbors?
Parameterizations Considered: dual parameterization k
Considered on pages: 88, 100–102

Odd Set of Size at most k, Odd Set of Size k
(also known as Odd Set and Exact Odd Set)
Input: A bipartite graph $G = (R, B, E)$ and $k \in \mathbb{N}$.
Question: Is there a set of vertices $S \subseteq R$ of size at most k (exactly k) in which every vertex of B has an odd number of neighbors?
Parameterizations Considered: solution-size k,
Considered on pages: 87–89, 99–101

Packing 3-Sets
Input: A system $\mathcal{C}$ of three-element subsets of a finite set S and an integer $k \in \mathbb{N}$.
Question: Is there a subsystem $\mathcal{C}'$ of at least k mutually disjoint sets?
Parameterizations Considered: solution-size k
Considered on pages: 44

Planar Deletion
Input: A graph G and $k \in \mathbb{N}$.
Question: Is there a set of at most k vertices such that its deletion turns G into

a planar graph?
Parameterizations Considered: solution-size k
Considered on pages: 30, 46

Short Multi-Tape Nondeterministic Turing Machine Computation
Input: A nondeterministic Turing machine $M = (\Sigma, t, Q, s, A, \Delta)$; a positive integer k.
Question: Is there a computation of M on empty input that reaches accepting state in at most k steps?
Parameterizations Considered: allowed length of a computation k, number of tapes
Considered on pages: 54

Short Nondeterministic Turing Machine Computation
Input: A single-tape nondeterministic Turing machine $M = (\Sigma, 1, Q, s, A, \Delta)$; a positive integer k.
Question: Is there a computation of M (on empty tape) that reaches the accepting state in at most k steps?
Parameterizations Considered: allowed length of a computation k, size of the alphabet $|\Sigma|$, number of states $|Q|$, amount of non-determinism, their combinations
Considered on pages: 54, 58

Steiner Tree
Input: A graph $G = (V, E)$ with integral weights on edges $w : E \to \mathbb{N}$, a set of terminals $T \subseteq V$ and an integer $p \in \mathbb{N}$.
Question: Is there a tree containing all the terminals of cost at most p?
Parameterizations Considered: maximum number of non-terminals in a solution (solution-size parameterization above tight lower bound) $p - |T| + 1$, number of terminals $|T|$
Considered on pages: 19, 35, 63, 64, 66–68

Strongly Connected Steiner Subgraph (SCSS)
Input: A set of vertices V, a weight function $w : V \times V \to W$ ($W \subseteq N \cup \{\infty\}$), a set $S \subseteq V$ of terminals such that $|S| = l$, and a weight bound $p \in \mathbb{N}$.
Question: Is there a set of arcs $A \subseteq V \times V$ of weight $w(A) \leq p$ such that in the digraph $D := (V, A)$ for every $s, t \in S$ there is a directed path from s to t?
Parameterizations Considered: solution-size $p/(\min W)$, number of terminals l, ratio $r = (\max W)/(\min W)$)
Considered on pages: 63, 64, 66, 67, 69–72, 74, 75, 79, 82, 83

Vertex Cover
Input: A graph G and $k \in \mathbb{N}$.

Question: Is there a set of at most k vertices, that contains at least one endpoint of each edge?
Parameterizations Considered: solution-size k
Considered on pages: 13, 15, 25, 26, 28, 34, 45, 46, 58, 97

Weighted t-Normalized Satisfiability
Input: A t-normalized Boolean formula φ and $k \in N$.
Question: Is there a satisfying assignment for φ of weight exactly k?
Parameterizations Considered: weight of the assignment k
Considered on pages: 50–52, 55

Weighted Antimonotone t-Normalized Satisfiability
Input: A t-normalized antimonotone Boolean formula φ and $k \in N$.
Question: Is there a satisfying assignment for φ of weight exactly k?
Parameterizations Considered: weight of the assignment k
Considered on pages: 50–52

Weighted Circuit Satisfiability
Input: A Boolean decision circuit C and $k \in N$.
Question: Is there a satisfying assignment for C of weight exactly k?
Parameterizations Considered: weight of the assignment k
Considered on pages: 50, 51

Weighted Monotone t-Normalized Satisfiability
Input: A t-normalized monotone Boolean formula φ and $k \in N$.
Question: Is there a satisfying assignment for φ of weight exactly k?
Parameterizations Considered: weight of the assignment k
Considered on pages: 50, 51, 66, 68

Weighted Satisfiability
Input: A Boolean formula φ and $k \in N$.
Question: Is there a satisfying assignment for φ of weight exactly k?
Parameterizations Considered: weight of the assignment k
Considered on pages: 50, 89

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. Cited on page 6.

[ABXR+10] Isolde Adler, Binh-Minh Bui-Xuan, Yuri Rabinovich, Gabriel Renault, Jan Arne Telle, and Martin Vatshelle. On the boolean-width of a graph: structure and applications. In Thilikos, editor, *Graph-Theoretic Concepts in Computer Science, WG 2010*, volume 6410 of *LNCS*. Springer, 2010. Cited on pages 86 and 105.

[ACP87] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2):277–284, 1987. Cited on page 20.

[ADF95] Karl R. Abrahamson, Rodney G. Downey, and Michael R. Fellows. Fixed-parameter tractability and completeness IV: On completeness for W[P] and PSPACE analogues. *Ann. Pure Appl. Logic*, 73(3):235–276, 1995. Cited on page 58.

[AEFM89] Karl R. Abrahamson, John A. Ellis, Michael R. Fellows, and Manuel E. Mata. On the complexity of fixed parameter problems (extended abstract). In *FOCS*, pages 210–215. IEEE, 1989. Cited on page 1.

[AG07] Noga Alon and Shai Gutner. Linear time algorithms for finding a dominating set of fixed size in degenerated graphs. In Lin, editor, *COCOON 2007*, LNCS vol. 4598, pages 394–405. Springer, 2007. Cited on page 107.

[AG08] Noga Alon and Shai Gutner. Kernels for the dominating set problem on graphs with an excluded minor. *Electronic Colloquium on Computational Complexity (ECCC)*, 15(066), 2008. Cited on page 22.

[AGK+10] Noga Alon, Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. Solving MAX-r-SAT above a tight lower bound. In

Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*, pages 511–517, 2010. Cited on page 19.

[AHS02] Esther M. Arkin, Rafael Hassin, and Shimon Shahar. Increasing digraph arc-connectivity by arc addition, reversal and complement. *Discrete Applied Mathematics*, 122(1-3):13–22, 2002. Cited on page 83.

[Alt07] Micah Altman. Is automation the answer? The computational complexity of automated redistricting. *Rutgers Computer and Technology Law Journal*, 23:81–142, 2007. Cited on page 109.

[AWW02] Karen Aardal, Robert Weismantel, and Laurence A. Wolsey. Non-standard approaches to integer programming. *Discrete Appl. Math.*, 123(1-3):5–74, 2002. Cited on page 41.

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. Cited on page 40.

[BDFH09] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009. Cited on pages 29 and 68.

[Ber62] Claude Berge. *Theory of Graphs and its Applications*. Methuen, London, 1962. Cited on page 85.

[BF05] Hans L. Bodlaender and Fedor V. Fomin. Equitable colorings of bounded treewidth graphs. *Theor. Comput. Sci.*, 349(1):22–30, 2005. Cited on page 110.

[BGN08] Nadja Betzler, Jiong Guo, and Rolf Niedermeier. Parameterized computational complexity of Dodgson and Young elections. In Gudmundsson, editor, *SWAT*, volume 5124 of *LNCS*, pages 402–413. Springer, 2008. Cited on page 19.

[BHKK07] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In *Proc. 39th STOC*, pages 67–74. ACM, 2007. Cited on page 66.

[BHM10] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Daniel Marx. Approximation schemes for Steiner forest on planar graphs and graphs of bounded treewidth. In *Proc. 42nd STOC*. ACM, 2010. Cited on page 83.

[BJ95] Hans L. Bodlaender and Klaus Jansen. Restrictions of graph partition problems. part I. *Theor. Comput. Sci.*, 148(1):93–109, 1995. Cited on page 110.

[BK94] Hans L. Bodlaender and Dieter Kratsch. A note on fixed parameter intractability of some domination-related problems. Private communication, 1994. Cited on pages 86 and 87.

[BK10] Hans L. Bodlaender and Arie M.C.A. Koster. Treewidth computations I. Upper bounds. *Information and Computation*, 208(3):259 – 275, 2010. Cited on page 20.

[Bod93] Hans L. Bodlaender. On linear time minor tests with depth-first search. *J. Algorithms*, 14(1):1–23, 1993. Cited on page 32.

[Bod96] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996. Cited on pages 20, 39, and 47.

[Bod09] Hans L. Bodlaender. Kernelization: New upper and lower bound techniques. In *Proc. 4th IWPEC*, volume 5917 of *LNCS*, pages 17–37. Springer, 2009. Cited on page 28.

[BTY08] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Analysis of data reduction: Transformations give evidence for non-existence of polynomial kernels. Technical report, Department of Information and Computing Sciences Utrecht University, Utrecht, The Netherlands, 2008. Cited on page 30.

[BvLvRV10] Hans L. Bodlaender, Erik Jan van Leeuwen, Johan M. M. van Rooij, and Martin Vatshelle. Faster algorithms on branch and clique decompositions. In Hliněný and Kučera, editors, *MFCS*, volume 6281 of *LNCS*, pages 174–185. Springer, 2010. Cited on pages 86 and 105.

[BXTV09] Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. Boolean-width of graphs. In Chen and Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009*, volume 5917 of *LNCS*, pages 61–74. Springer, 2009. Cited on page 21.

[CCC+99] Moses Charikar, Chandra Chekuri, To-yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed Steiner problems. *Journal of Algorithms*, 33(1):73–91, 1999. Cited on page 65.

[CEGS08] Chandra Chekuri, Guy Even, Anupam Gupta, and Danny Segev. Set connectivity problems in undirected graphs and the directed Steiner network problem. In *Proc. 19th SODA*, pages 532–541. SIAM, 2008. Cited on page 65.

[Ces02] Marco Cesati. Perfect code is $W[1]$-complete. *Inf. Process. Lett.*, 81(3):163–168, 2002. Cited on pages 86 and 87.

[Ces03] Marco Cesati. The Turing way to parameterized complexity. *J. Comput. Syst. Sci.*, 67(4):654–685, 2003. Cited on page 54.

[CF03] Yijia Chen and Jörg Flum. Machine characterization of the classes of the W-hierarchy. In Baaz and Makowsky, editors, *CSL*, volume 2803 of *LNCS*, pages 114–127. Springer, 2003. Cited on pages 53, 54, and 55.

[CFG03] Yijia Chen, Jörg Flum, and Martin Grohe. Bounded nondeterminism and alternation in parameterized complexity theory. In *IEEE Conference on Computational Complexity*, pages 13–29. IEEE Computer Society, 2003. Cited on pages 54 and 55.

[CFJ04] Benny Chor, Mike Fellows, and David W. Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In Hromkovič, Nagl, and Westfechtel, editors, *Graph-Theoretic Concepts in Computer Science, 30th International Workshop,WG 2004*, volume 3353 of *LNCS*, pages 257–269, 2004. Cited on page 27.

[Cha10] Mathieu Chapelle. Parameterized complexity of generalized domination problems on bounded tree-width graphs. *CoRR*, abs/1004.2642, 2010. Cited on pages 86 and 105.

[CHKX04] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Linear fpt reductions and computational lower bounds. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 212–221, New York, NY, USA, 2004. ACM. Cited on page 58.

[CI97] Marco Cesati and Miriam Di Ianni. Computation models for parameterized complexity. *Math. Log. Q.*, 43:179–202, 1997. Cited on page 54.

[CJ03] Liming Cai and David W. Juedes. On the existence of subexponential parameterized algorithms. *J. Comput. Syst. Sci.*, 67(4):789–807, 2003. Cited on page 58.

[CKL95] Bor-Liang Chen, Ming-Tat Ko, and Ko-Wei Lih. Equitable and m-bounded coloring of split graphs. In Michel Deza, Reinhardt Euler, and Yannis Manoussakis, editors, *Combinatorics and Computer Science*, volume 1120 of *LNCS*, pages 1–5. Springer, 1995. Cited on page 110.

[CKX06] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved parameterized upper bounds for vertex cover. In Kralovic and Urzyczyn, editors, *MFCS*, volume 4162 of *LNCS*, pages 238–249. Springer, 2006. Cited on page 117.

[CL94] Bor-Liang Chen and Ko-Wei Lih. Equitable coloring of trees. *J. Comb. Theory, Ser. B*, 61(1):83–87, 1994. Cited on page 110.

[CMR00] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000. Cited on pages 39 and 105.

[CO00] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000. Cited on page 9.

[Cou92] Bruno Courcelle. The monadic second-order logic of graphs III: Tree-decompositions, minor and complexity issues. *ITA*, 26:257–286, 1992. Cited on page 38.

[CR72] Stephen A. Cook and Robert A. Reckhow. Time-bounded random access machines. In *STOC '72: Proceedings of the fourth annual ACM symposium on Theory of computing*, pages 73–80, New York, NY, USA, 1972. ACM. Cited on page 6.

[DCB74] Philip A. Tsichritzis Dionysios C. Bernstein. *Operating Systems*. Academic Press, New York, 1974. Cited on page 109.

[DF85] Martin E. Dyer and Alan M. Frieze. On the complexity of partitioning graphs into connected subgraphs. *Discrete Applied Mathematics*, 10(2):139 – 153, 1985. Cited on page 110.

[DF92a] Rodney G. Downey and Michael R. Fellows. Fixed-parameter intractability. In *Structure in Complexity Theory Conference*, pages 36–49, 1992. Cited on page 1.

[DF92b] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness. *Congressus Numerantium*, 87:161–178, 1992. Cited on pages 1, 86, and 87.

[DF95a] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995. Cited on pages 51, 68, 86, and 87.

[DF95b] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theor. Comput. Sci.*, 141(1&2):109–131, 1995. Cited on pages 51, 52, 68, and 89.

[DF98] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1998. Cited on pages 1, 13, 15, 25, 31, 51, 66, 67, 68, and 86.

[DFHT05] Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52(6):866–893, 2005. Cited on pages 46, 47, and 48.

[DFVW99] Rodney G. Downey, Michael R. Fellows, Alexander Vardy, and Geoff Whittle. The parametrized complexity of some fundamental problems in coding theory. *SIAM J. Comput.*, 29(2):545–570 (electronic), 1999. Cited on pages 87, 89, 99, and 100.

[Dij59] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. 10.1007/BF01386390. Cited on page 36.

[DK99] Yevgeniy Dodis and Sanjeev Khanna. Designing networks with bounded pairwise distance. In *Proc. 31th STOC*, pages 750–759. ACM, 1999. Cited on page 65.

[DK09] Anuj Dawar and Stephan Kreutzer. Parameterized complexity of first-order logic. *Electronic Colloquium on Computational Complexity*, TR09-131, 2009. Cited on pages 39 and 105.

[DKT09] Zdeněk Dvořák, Daniel Král', and Robin Thomas. Deciding first-order properties for sparse graphs. accepted to FOCS'10, prelim. version available as vol. 484, KAM-DIMATIA Series, 2009. Cited on page 39.

[DLS09] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Incompressibility through colors and IDs. In *ICALP '09: Proceedings of the 36th International Colloquium on Automata, Languages and Programming*, pages 378–389, Berlin, Heidelberg, 2009. Springer-Verlag. Cited on pages 29 and 30.

[DvM10] Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In Schulman [Sch10], pages 251–260. Cited on pages 30 and 31.

[DW72] Stuart E. Dreyfus and Robert A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972. Cited on pages 35, 63, and 66.

[DYW+07] Bolin Ding, Jeffrey Xu Yu, Shan Wang, Lu Qin, Xiao Zhang, and Xuemin Lin. Finding top-k min-cost connected trees in databases. In *Proc. 23rd ICDE*, pages 836–845. IEEE, 2007. Cited on pages 35, 66, and 67.

[EFG+09] Rosa Enciso, Michael R. Fellows, Jiong Guo, Iyad A. Kanj, Frances A. Rosamond, and Ondřej Suchý. What makes equitable connected partition easy. In Chen and Fomin, editors, *IWPEC*, volume 5917 of *LNCS*, pages 122–133. Springer, 2009. Cited on page 109.

[ET76] Kapali P. Eswaran and Robert E. Tarjan. Augmentation problems. *SIAM Journal on Computing*, 5(4):653–665, 1976. Cited on page 65.

[Eve07] Guy Even. Recursive greedy methods. In Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*, chapter 5. CRC, 2007. Cited on page 65.

[Fei98] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998. Cited on page 65.

[Fel09] Michael R. Fellows. Towards fully multivariate algorithmics: Some new results and directions in parameter ecology. In *Proc. 20th IWOCA*, volume 5874 of *LNCS*, pages 2–10. Springer, 2009. Cited on pages 2 and 23.

[FF56] Lester R. Ford and Delbert R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. Cited on page 44.

[FFL+07] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, Stefan Szeider, and Carsten Thomassen. On the complexity of some colorful problems parameterized by treewidth. In Dress, Xu, and Zhu, editors, *COCOA*, volume 4616 of *LNCS*, pages 366–377. Springer, 2007. Cited on pages 57, 110, and 111.

[FFL+09] Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Daniel Raible, Saket Saurabh, and Yngve Villanger. Kernel(s) for problems with no kernel: On out-trees with many leaves. In Albers and

Marion, editors, *STACS*, volume 3 of *LIPIcs*, pages 421–432. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009. Cited on pages 30 and 31.

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc., 2006. Cited on pages 2, 13, 14, 15, and 27.

[FGK+07] Fedor V. Fomin, Petr A. Golovach, Jan Kratochvíl, Dieter Kratsch, and Mathieu Liedloff. Branch and recharge: Exact algorithms for generalized domination. In Dehne, Sack, and Zeh, editors, *WADS*, volume 4619 of *LNCS*, pages 507–518. Springer, 2007. Cited on page 86.

[FGK09a] Jiří Fiala, Petr A. Golovach, and Jan Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. In Chen and Cooper, editors, *Theory and Applications of Models of Computation, 6th Annual Conference, TAMC 2009*, volume 5532 of *LNCS*, pages 221–230. Springer, 2009. Cited on pages 41 and 111.

[FGK+09b] Fedor V. Fomin, Petr A. Golovach, Jan Kratochvíl, Dieter Kratsch, and Mathieu Liedloff. Sort and search: Exact algorithms for generalized domination. *Inf. Process. Lett.*, 109(14):795–798, 2009. Cited on page 86.

[FHRV09] Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009. Cited on pages 56 and 57.

[FKM+07] Bernhard Fuchs, Walter Kern, Daniel Mölle, Stefan Richter, Peter Rossmanith, and Xinhui Wang. Dynamic programming for minimum Steiner trees. *Theory of Computing Systems*, 41(3):493–500, 2007. Cited on page 66.

[FLST10] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. In Charikar, editor, *SODA*, pages 503–510. SIAM, 2010. Cited on pages 48 and 107.

[FR06] Jon Feldman and Matthias Ruhl. The directed Steiner network problem is tractable for a constant number of terminals. *SIAM Journal on Computing*, 36(2):543–561, 2006. Cited on pages 66, 79, 82, and 83.

[Fra92] András Frank. Augmenting graphs to meet edge-connectivity requirements. *SIAM Journal on Discrete Mathematics*, 5(1):25–53, 1992. Cited on pages 63, 65, 66, 67, 68, and 70.

[FS08] Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 133–142. ACM, 2008. Cited on page 29.

[FT87a] András Frank and Éva Tardos. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987. Cited on page 119.

[FT87b] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987. Cited on page 37.

[Gas10] Elisabeth Gassner. The steiner forest problem revisited. *Journal of Discrete Algorithms*, 8(2):154 – 163, 2010. Selected papers from the 3rd Algorithms and Complexity in Durham Workshop ACiD 2007. Cited on page 83.

[GHK+09] Robert Ganian, Petr Hliněný, Joachim Kneis, Alexander Langer, Jan Obdržálek, and Peter Rossmanith. On digraph width measures in parameterized algorithmics. In *Proc. 4th IWPEC*, volume 5917 of *LNCS*, pages 185–197. Springer, 2009. Cited on page 65.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. Cited on pages 51, 65, 85, and 110.

[GK07] Petr A. Golovach and Jan Kratochvíl. Computational complexity of generalized domination: A complete dichotomy for chordal graphs. In Brandstädt, Kratsch, and Müller, editors, *WG 2007*, LNCS vol. 4769, pages 1–11. Springer, 2007. Cited on page 86.

[GK08] Petr A. Golovach and Jan Kratochvíl. Generalized domination in degenerate graphs: A complete dichotomy of computational complexity. In Agrawal, Du, Duan, and Li, editors, *TAMC 2008*, LNCS vol. 4978, pages 182–191. Springer, 2008. Cited on page 86.

[GKS09] Petr A. Golovach, Jan Kratochvíl, and Ondřej Suchý. Parameterized complexity of generalized domination problems. In Paul and Habib, editors, *WG*, volume 5911 of *LNCS*, pages 133–142, 2009. journal version to appear in Disc. App. Math. Cited on page 87.

[GMN09] Jiong Guo, Hannes Moser, and Rolf Niedermeier. Iterative compression for exactly solving NP-hard minimization problems. In Lerner, Wagner, and Zweig, editors, *Algorithmics of Large and Complex Networks*, volume 5515 of *LNCS*, pages 65–80. Springer, 2009. Cited on page 44.

[GN07] Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007. Cited on page 28.

[GNS09] Jiong Guo, Rolf Niedermeier, and Ondřej Suchý. Parameterized complexity of arc-weighted directed Steiner problems. In Dong, Du, and Ibarra, editors, *ISAAC*, volume 5878 of *LNCS*, pages 544–553. Springer, 2009. journal version to appear in SIAM J. on Discrete Math. Cited on page 66.

[GT86] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum flow problem. In *STOC '86: Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 136–146, New York, NY, USA, 1986. ACM. Cited on page 118.

[GT05] Qian-Ping Gu and Hisao Tamaki. Optimal branch-decomposition of planar graphs in $O(n^3)$ time. In Caires, Italiano, Monteiro, Palamidessi, and Yung, editors, *ICALP*, volume 3580 of *LNCS*, pages 373–384. Springer, 2005. Cited on page 48.

[GY08] Gregory Gutin and Anders Yeo. Some parameterized problems on digraphs. *The Computer Journal*, 51(3):363–371, 2008. Cited on page 83.

[Gyö76] Ervin Györi. On division of graphs to connected subgraphs. In *Proceedings of the Fifth Hungarian Combinatorial Colloquium*, 1976. Cited on pages 110 and 121.

[Hak71] S. Luis Hakimi. Steiner's problem in graphs and its implications. *Networks*, 1:113–133, 1971. Cited on page 63.

[HK62] Michael Held and Richard M. Karp. A dynamic programming approach to sequencing problems. *Journal of SIAM*, 10:196–210, 1962. Cited on page 75.

[HKT00] Magnús M. Halldórsson, Jan Kratochvíl, and Jan Arne Telle. Mod-2 independence and domination in graphs. *Int. J. Found. Comput. Sci.*, 11(3):355–363, 2000. Cited on pages 86 and 99.

[HL90] Stephen T. Hedetniemi and Renu C. Laskar. Bibliography on domination in graphs and some basic definitions of domination parameters. *Discrete Mathematics*, 86(1-3):257 – 277, 1990. Cited on page 85.

[HN10] Sepp Hartung and Rolf Niedermeier. Incremental list coloring of graphs, parameterized by conservation. In Kratochvíl, Li, Fiala, and Kolman, editors, *Theory and Applications of Models of Computation, 7th Annual Conference, TAMC 2010*, volume 6108 of *LNCS*, pages 258–270. Springer, 2010. Cited on page 19.

[HOSG08] Petr Hliněný, Sang-il Oum, Detlef Seese, and Georg Gottlob. Width parameters beyond tree-width and their applications. *Comput. J.*, 51(3):326–362, 2008. Cited on pages 9 and 21.

[HS70] András Hajnal and Endre Szemerédi. Proof of a conjecture of p. erdös. *Combinatorial Theory and its Application*, pages 601–623, 1970. Cited on pages 110 and 121.

[HT98] Pinar Heggernes and Jan Arne Telle. Partitioning graphs into generalized dominating sets. *Nordic J. Comput.*, 5(2):128–142, 1998. Cited on page 85.

[IGZN07] Takehiro Ito, Kazuya Goto, Xiao Zhou, and Takao Nishizeki. Partitioning a multi-weighted graph to connected subgraphs of almost uniform size. *IEICE Transactions*, 90-D(2):449–456, 2007. Cited on page 109.

[IZN06] Takehiro Ito, Xiao Zhou, and Takao Nishizeki. Partitioning a graph of bounded tree-width to connected subgraphs of almost uniform size. *J. Discrete Algorithms*, 4(1):142–154, 2006. Cited on pages 109 and 110.

[JK34] Vojtěch Jarník and Miloš Kössler. O minimálních grafech obsahujících n daných bodů. *Časopis Pěst. Mat. Fys.*, 63:223–235, 1934. Cited on page 63.

[JZC04] Weijia Jia, Chuanlin Zhang, and Jianer Chen. An efficient parameterized algorithm for m-set packing. *J. Algorithms*, 50(1):106–117, 2004. Cited on page 44.

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987. Cited on page 41.

[KC00] Ton Kloks and Leizhen Cai. Parameterized tractability of some (efficient) Y - domination variants for planar graphs and t-degenerate graphs. In *Proceedings of the International Computer Symposium (ICS 2000), Taiwan*, 2000. Cited on page 107.

[KH78] David G. Kirkpatrick and Pavol Hell. On the completeness of a generalized matching problem. In *STOC '78: Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 240–245, New York, NY, USA, 1978. ACM. Cited on page 109.

[Khu95] Samir Khuller. Approximation algorithms for finding highly connected subgraphs. In Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*. PWS, 1995. Cited on page 65.

[KKMS10] Henry Kierstead, Alexandr Kostochka, Marcelo Mydlarz, and Endre Szemerédi. A fast algorithm for equitable coloring. *Combinatorica*, 30:217–224, 2010. 10.1007/s00493-010-2483-5. Cited on page 110.

[KMM95] Jan Kratochvíl, Paul D. Manuel, and Mirka Miller. Generalized domination in chordal graphs. *Nordic J. Comput.*, 2(1):41–50, 1995. Cited on page 86.

[KN03] Alexandr V. Kostochka and Kittikorn Nakprasit. Equitable colourings of d-degenerate graphs. *Combinatorics, Probability & Computing*, 12(1), 2003. Cited on pages 110 and 122.

[KN07] Guy Kortsarz and Zeev Nutov. Approximating minimum cost connectivity problems. In Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*, chapter 58. CRC, 2007. Cited on page 65.

[Kna10] Christian Knauer. The complexity of geometric problems in high dimension. In Kratochvíl, Li, Fiala, and Kolman, editors, *Theory and Applications of Models of Computation, 7th Annual Conference, TAMC 2010*, volume 6108 of *LNCS*, pages 40–49. Springer, 2010. Cited on page 19.

[Kön50] Denes König. *Theorie der Endlichen und Unendlichen Graphen*. Chelsea, New York, 1950. Cited on page 85.

[Kra91] Jan Kratochvíl. Perfect codes in general graphs. *Rozpravy Československé Akad. Věd Řada Mat. Přírod. Věd*, 7, 1991. Cited on page 90.

[KT05] Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005. Cited on page 34.

[KW91] Daniel J. Kleitman and Douglas B. West. Spanning trees with many leaves. *SIAM J. Discret. Math.*, 4(1):99–106, 1991. Cited on pages 21, 118, and 121.

[KW10] Ken-ichi Kawarabayashi and Paul Wollan. A shorter proof of the graph minor algorithm: the unique linkage theorem. In Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010*, pages 687–694. ACM, 2010. Cited on page 45.

[Len83] Hendrik W. Lenstra, jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983. Cited on page 41.

[Lev71] Anatolii Y. Levin. Algorithm for the shortest connection of a group of graph vertices. *Sov. Math. Dokl.*, 12:1477–1481, 1971. Cited on page 63.

[Lov77] László Lovasz. A homology theory for spanning tress of a graph. *Acta Mathematica Hungarica*, 30:241–251, 1977. 10.1007/BF01896190. Cited on pages 110 and 121.

[LSS09] Daniel Lokshtanov, Saket Saurabh, and Somnath Sikdar. Simpler parameterized algorithm for OCT. In Fiala, Kratochvíl, and Miller, editors, *IWOCA*, volume 5874 of *LNCS*, pages 380–384. Springer, 2009. Cited on page 42.

[Mar08] Dániel Marx. Searching the k-change neighborhood for TSP is W[1]-hard. *Oper. Res. Lett.*, 36(1):31–36, 2008. Cited on page 19.

[Mey73] Walter Meyer. Equitable coloring. *American Mathematical Monthly*, 80:920–922, 1973. Cited on page 110.

[MN09] Jiří Matoušek and Jaroslav Nešetřil. *Invitation to Discrete Mathematics, Second Edition.* Oxford University Press, 2009. Cited on page 6.

[Moh99] Bojan Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM J. Discrete Math.*, 12(1):6–26, 1999. Cited on page 20.

[MT06] Hannes Moser and Dimitrios M. Thilikos. Parameterized complexity of finding regular induced subgraphs. In Broersma, Dantchev, Johnson, and Szeider, editors, *ACiD 2006*, volume 7 of *Texts in Algorithmics*, pages 107–118. King's College, London, 2006. Cited on pages 86, 87, and 98.

[Nie06] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006. Cited on pages 1 and 13.

[Nie10] Rolf Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In *Proc. 27th STACS*, volume 5 of *LIPIcs*, pages 17–32. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. Cited on pages 2 and 23.

[NOdM05] Jaroslav Nešetřil and Patrice Ossona de Mendez. The grad of a graph and classes with bounded expansion. *Electronic Notes in Discrete Mathematics*, 22:101–106, 2005. Cited on page 10.

[NOdM08a] Jaroslav Nešetřil and Patrice Ossona de Mendez. First order properties on nowhere dense structures. *J. Symbolic Logic*, 2008. submitted. Cited on page 10.

[NOdM08b] Jaroslav Nešetřil and Patrice Ossona de Mendez. Grad and classes with bounded expansion I. Decompositions. *Eur. J. Comb.*, 29(3):760–776, 2008. Cited on page 10.

[NOdM08c] Jaroslav Nešetřil and Patrice Ossona de Mendez. Grad and classes with bounded expansion II. Algorithmic aspects. *Eur. J. Comb.*, 29(3):777–791, 2008. Cited on page 10.

[NOdM08d] Jaroslav Nešetřil and Patrice Ossona de Mendez. Grad and classes with bounded expansion III. Restricted graph homomorphism dualities. *Eur. J. Comb.*, 29(4):1012–1024, 2008. Cited on page 10.

[NOdM08e] Jaroslav Nešetřil and Patrice Ossona de Mendez. On nowhere dense graphs. *Eur. J. Comb.*, 2008. submitted. Cited on page 10.

[NP85] Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985. Cited on page 59.

[NR03] Rolf Niedermeier and Peter Rossmanith. An efficient fixed-parameter algorithm for 3-Hitting Set. *J. Discrete Algorithms*, 1(1):89–102, 2003. Cited on page 35.

[Ore62] Oystein Ore. *Theory of Graphs*, volume 38 of *Amer. Math. Soc. Colloq. Publ.* Amer. Math. Soc., Providence, RI, 1962. Cited on page 85.

[Pie03] Krzysztof Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *Journal of Computer and System Sciences*, 67(4):757 – 771, 2003. Parameterized Computation and Complexity 2003. Cited on page 56.

[PS81] Yehoshua Perl and Stephen R. Schach. Max-min tree partitioning. *J. ACM*, 28(1):5–15, 1981. Cited on page 109.

[PW10] Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*, pages 1065–1075, 2010. Cited on page 59.

[Rao08] Michaël Rao. Clique-width of graphs defined by one-vertex extensions. *Discrete Mathematics*, 308(24):6157 – 6165, 2008. Cited on page 21.

[Ree92] Bruce A. Reed. Finding approximate separators and computing tree width quickly. In *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 221–228, New York, NY, USA, 1992. ACM. Cited on page 20.

[RS83] Neil Robertson and Paul D. Seymour. Graph minors. I. Excluding a forest. *J. Comb. Theory, Ser. B*, 35(1):39–61, 1983. Cited on page 45.

[RS84] Neil Robertson and Paul D. Seymour. Graph minors. III. Planar tree-width. *J. Comb. Theory, Ser. B*, 36(1):49–64, 1984. Cited on pages 9 and 20.

[RS95] Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995. Cited on page 45.

[RS04] Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *J. Comb. Theory, Ser. B*, 92(2):325–357, 2004. Cited on page 45.

[RST94] Neil Robertson, Paul D. Seymour, and Robin Thomas. Quickly excluding a planar graph. *J. Comb. Theory, Ser. B*, 62(2):323–348, 1994. Cited on page 46.

[RSV04] Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. Cited on page 42.

[Sch10] Schulman, editor. *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*. ACM, 2010. Cited on page 137.

[ST94] Paul D. Seymour and Robin Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994. Cited on page 48.

[Tel94a] Jan Arne Telle. Complexity of domination-type problems in graphs. *Nordic J. Comput.*, 1(1):157–171, 1994. Cited on pages 85, 87, and 88.

[Tel94b] Jan Arne Telle. *Vertex partitioning problems: characterization, complexity and algorithms on partial k-trees, PhD thesis.* Department of Computer Science, University of Oregon, Eugene, 1994. Cited on pages 85, 86, and 105.

[TP97] Jan Arne Telle and Andrzej Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM J. Discrete Math.*, 10(4):529–550, 1997. Cited on pages 85, 86, and 105.

[vRBR09] Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In Amos Fiat and Peter Sanders, editors, *ESA*, volume 5757 of *LNCS*, pages 566–577. Springer, 2009. Cited on pages 86 and 105.

[Wes96] Douglas B. West. *Introduction to graph theory.* Prentice Hall Inc., Upper Saddle River, NJ, 1996. Cited on page 6.