Algorithms in
Moderately Exponential Time
Algorithms in Moderately Exponential Time

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de rector magnificus, prof.dr.ir. C.J. van Duijn, voor een commissie aangewezen door het College voor Promoties in het openbaar te verdedigen op woensdag 22 september 2010 om 16.00 uur

door

Matthias Mnich

geboren te Parchim, Duitsland
Abstract

Algorithms in Moderately Exponential Time

This thesis studies exponential time algorithms that give optimum solutions to optimization problems which are highly unlikely to be solvable to optimality in polynomial time. Such algorithms are designed and upper bounds on their running times are analyzed, for problems from graph theory, constraint satisfaction, and computational biology.

It is studied how restrictions on one computationally hard parameter affect the running time of algorithms for computing another parameter. Algorithms are obtained to compute minimum dominating sets, minimum bandwidth and topological bandwidth layouts, and other hard parameters, in time exponential only in the maximum number of leaves in a spanning tree of a graph. For sparse graphs, algorithms find minimum connected dominating sets in time exponential only in the optimum value. These algorithms are based on kernels of small size, meaning that arbitrary instances are polynomial-time compressible to equivalent instances whose size depends on the parameter alone.

Lower and upper bounds on the number and size of certain combinatorial objects are proved. For minimal feedback vertex sets in tournaments, improved bounds on their number and a polynomial-space polynomial-delay algorithm for their enumeration are given. For subcubic planar graphs, fast algorithms finding large induced matchings are given.

Fixed-parameter algorithms are presented for maximization problems whose optimum solution value grows as an unbounded function with the instance size. Polynomial kernels are obtained for all permutation constraint satisfaction problems with ternary constraints, and independent set problems in restricted graph classes, when these problems are parameterized above tight lower bounds on their solution value.

Polynomial space algorithms with subexponential time requirement are obtained for the minimum triplet inconsistency problem in phylogenetics, and basically all bidimensional problems in sparse graphs such as graphs excluding a fixed minor.
Samenvatting

Algoritmen met bescheiden exponentiële rekentijd

In dit proefschrift bestuderen we algoritmen met exponentiële rekentijden die optimale oplossingen geven voor problemen waarvan het onwaarschijnlijk is dat die in polynomialie tijd optimaal kunnen worden opgelost. We ontwerpen algoritmen en analyseren bovengrenzen op hun rekentijden voor problemen uit de grafentheorie, constraint satisfaction en de computacionele biologie.

We bestuderen hoe restricties op de ene computationeel lastige structurele parameter de rekentijden beïnvloeden van algoritmen voor het bepalen van een andere structurele parameter. We ontwerpen algoritmen om optimale dominating sets te bepalen, optimale bandbreedte en topologische bandbreedte verdelingen, en andere lastige parameters, in een rekentijd die weliswaar exponentieel is, maar slechts in het maximaal aantal bladeren in een opspannende boom van de graaf. Voor sparce grafen vinden de algoritmen exponentieel in de waarde van het antwoord. Deze algoritmen zijn gebaseerd op kernels van kleine omvang, wat betekent dat willekeurige instanties in polynomialie tijd gereduceerd kunnen worden naar equivalentale instanties waarvan de grootte alleen afhangt van de parameter.

We leiden onder- en bovengrenzen af op het aantal en de grootte van zekere combinatorische objecten. Voor minimale feedback vertex sets in toernooigrafen geven we verbeterde grenzen op hun aantal, met een algoritme voor hun enumeratie die polynomialie ruimte gebruikt en een polynomialie rekentijd per gevonden oplossing kost. Voor sub-kubische planaire grafen geven we snelle algoritmen voor het vinden van grote geïnduceerde koppelingen.

We presenteren vaste parameter algoritmen voor maximaliseringsproblemen waarvan de optimale oplossingswaarde groeit als een onbegrensde functie van de instantiegrootheid. Polynomialie kernels worden gevonden voor alle permutation constraint satisfaction problemen met ternaire beperkingen, en voor independent set problemen in een beperkte klasse van grafen, wanneer deze problemen parameters hebben hoger dan krappe ondergrenzen op hun oplossingswaarde.

Algoritmen met polynomialia ruimtegebruik en subexponentiële rekentijden worden afgeleid voor het minimum-triplet-inconsistentieprobleem in de phylogenetica, en voor zo goed als alle tweedimensionale problemen in sparce grafen die een vaste minor uitsluiten.
Acknowledgements

This thesis presents the results of four years work as a PhD student at the Technische Universiteit Eindhoven. Many people have contributed to their development, and here I want to thank some of them.

First and foremost, I want to thank my supervisor Gerhard Woeginger. His thoughtful advice guided my ideas in the right directions, and his trust in my abilities made my time as a PhD-student a particularly enjoyable one. Our conversations on algorithms and complexity, academic writing, and German literature were all insightful and pleasant—danke!

As a PhD student, I often approached borders of tractability, and borders of countries. The latter was made possible by the hospitality and time to exchange ideas of Hans Bodlaender, Jean Daligault, Holger Dell, Michael Dom, Mike Fellows, Serge Gaspers, Gregory Gutin, Thore Husfeldt, Ross Kang, Petteri Kaski, Eun Jung Kim, Mikko Koivisto, Daniel Lokshtanov, Hannes Moser, Rolf Niedermeier, Anthony Perez, Christophe Paul, Tobias Müller, Johan van Rooij, Fran Rosamond, Saket Saurabh, Ildikó Schlotter, Siamak Tazari, Alexander Wolff, and Anders Yeo, whom I want to thank for that.

I was fortunate to have an inspiring set of collaborators: Mike Fellows, Serge Gaspers, Sylvain Guillemot, Gregory Gutin, Leo van Iersel, Ross Kang, Steven Kelk, EunJung Kim, Daniel Lokshtanov, Neeldhara Misra, Tobias Müller, Johan van Rooij, Fran Rosamond, Saket Saurabh, and Anders Yeo.

Thanks to Adriana Gabor for her guidance in the initial phase of my PhD; and thanks to Alexander Wolff for his enthusiasm and guidance. Thanks to Mike Fellows and Fran Rosamond for sharing their passion for Parameterized Complexity, hiking, and mathematical theater with me; and thanks to Saket Saurabh for his clear and critical views of my work.

Special thanks to the people in Eindhoven that made our shared time a pleasant one: Peter Korteweg, Peter van Liesdonk, Maciej Modelsik, Christiane Peters, Daniela Sudfeld, José Villegas and Stefan van Zwam.

Dank aan mijn collega’s in de groep Combinatorische Optimalisering, in bijzonderder aan Cor Hurkens, Judith Keijser en Rudi Pendavingh voor hun goede ideeën.

Thanks to the Royal Mathematical Society in the Netherlands for awarding me their Philips Prize in 2010.

Thanks in advance to the members of my promotion committee: Mark de Berg, Hans Bodlaender, Andries Brouwer, Dieter Kratsch, Rolf Niedermeier, Marc Uetz and Gerhard Woeginger.

Meiner Familie danke ich für ihre uneingeschränkte Unterstützung, und ihr stetiges Interesse an meiner Arbeit.

Rita, köszönöm neked a sok erőt és szeretetet.

Eindhoven, 31 May 2010
Contents

1 Introduction 13
  1.1 Techniques for Moderately Exponential-Time Algorithms . . . . . . . 15
  1.2 Results and Outline of this Thesis . . . . . . . . . . . . . . . . . . . . 26
  1.3 Notions and Notations . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
  1.4 List of Problems . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36

2 The Complexity Ecology of Parameters 41
  2.1 A Complexity Matrix of Parameters . . . . . . . . . . . . . . . . . . . . 42
  2.2 Attacking a Row: Kernelization . . . . . . . . . . . . . . . . . . . . . . 44
  2.3 Attacking a Row: Win/Wins . . . . . . . . . . . . . . . . . . . . . . . . . 53
  2.4 Attacking a Row: Well-Quasi Ordering . . . . . . . . . . . . . . . . . . 54
  2.5 Attacking an Entry: Bandwidth Parameterized by Max-Leaf Number 56
  2.6 Attacking an Entry: Fixed-Parameter Approximation . . . . . . . . . . 62
  2.7 Concluding Remarks . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64

3 Kernelization 65
  3.1 Connected Dominating Set in Planar Graphs . . . . . . . . . . . . . . . 65
  3.2 Connected Dominating Set in $K_{3,3}$-topological minor free Graphs . . 75
  3.3 Kernelization Lower Bounds . . . . . . . . . . . . . . . . . . . . . . . . . 81
  3.4 Concluding Remarks . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 82

4 Algorithms through Extremal Combinatorics 85
  4.1 Minimal Feedback Vertex Sets in Tournaments . . . . . . . . . . . . . . . 89
  4.2 Minimum Feedback Vertex Set in Bipartite Directed Graphs . . . . . . . 105
  4.3 Induced Matchings in Planar Graphs . . . . . . . . . . . . . . . . . . . . 108
  4.4 Concluding Remarks . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132

5 Parameterizations Above Guarantee 135
  5.1 Independent Set in Graphs of Maximum Degree Three . . . . . . . . . . 136
  5.2 Ordinal Embeddings . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 137
  5.3 Ternary Permutation-CSPs . . . . . . . . . . . . . . . . . . . . . . . . . . 146
  5.4 Concluding Remarks . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 162
6 Subexponential Parameterized Algorithms
   6.1 Minimum Triplet Inconsistency ....................................... 164
   6.2 Bidimensionality in Polynomial Space .............................. 176
   6.3 Concluding Remarks .................................................... 188

7 Perspectives ....................................................................... 191

A Program for Second Moment Evaluation ............................... 193

Curriculum Vitæ ................................................................... 195

Bibliography ........................................................................ 197
Many fundamental problems in theoretical computer science have been classified as “intractable”, meaning that no “efficient” algorithm to solve them is known or likely to exist. As this is no excuse for not solving these problems, algorithms have been designed that produce solutions of different qualities, compared to an optimal solution. Popular kinds of algorithms for intractable problems are heuristics, that aim for good-quality solutions for instances occurring in practice but for which no such quality can be guaranteed, and approximation algorithms, that produce solutions for which the ratio between the quality of a produced solution and an optimal solution can be bounded, for all possible instances. Both kinds of algorithms are “efficient”, that is, the time required by them to produce a solution is bounded by a polynomial in the size of the input instance.

Another kind of algorithms for intractable problems is the topic of this thesis: exact algorithms, that always return an optimal solution for any instance. Our objective is to design exact algorithms running in time that is “moderate” for instances of relatively small size. That means, their time requirement is provably (and desirably significantly) less than that of exhaustively searching through all feasible solutions for an optimal one.

Let us clarify what we mean by intractable, efficient and moderate. As customary, for input instances that are graphs, formulas, collections of trees or sets, we denote by $n$ the number of vertices, variables, leaf labels or elements of the input. For all other types of input instances we denote by $n$ the length of the input, unless specified otherwise. We say that a problem can be solved efficiently if some algorithm solves it in time that is bounded by a polynomial in $n$. This class of problems will be denoted by $P$. Decision problems for which, given an input instance $x$ of it together with a certificate $y(x)$, there is a polynomial-time algorithm deciding whether $x$ is a “yes”-instance of the problem, form the class $NP$. The inclusion $P \subseteq NP$ follows by definition, but whether equality holds is a long-standing open problem in theoretical computer science. It is however by now a widely believed conjecture that equality does not hold, that is, the inclusion is strict. A problem $\Pi$ is NP-hard if all problems in NP are polynomial-time reducible to $\Pi$, and $\Pi$ is NP-complete if it is NP-hard and belongs to NP. We call NP-hard problems intractable.
In this thesis we design algorithms for problems that are intractable. Definitions of all problems mentioned in this thesis are given in Section 1.4. Under the assumption that $P \neq NP$, solving an intractable problem optimally requires time that is superpolynomial in $n$. Yet every problem in $NP$ can be solved optimally by exhaustively searching through the set of feasible solutions, or "brute force". As such takes a prohibitively large amount of time even for small $n$, we design algorithms whose running time is provably (and often significantly) faster than exhaustive search. We refer to such algorithms as moderately exponential-time algorithms, and they are the topic of this thesis.

Measuring the computational complexity of a problem in a more refined way than only by $n$ can be very enlightening. For such a refined analysis of intractable problems we use the multivariate framework of Parameterized Complexity Theory. Here, every input instance $x$ of a problem $\Pi$ is accompanied with an integer parameter $k$ and $\Pi$ is said to be fixed-parameter tractable if $(x, k) \in \Pi$ is decided by an algorithm running in time $f(k) \cdot n^{O(1)}$, where $f$ is a computable function. A central problem in parameterized complexity is to obtain algorithms with running time $f(k) \cdot n^{O(1)}$ such that $f$ is a function growing as slowly as possible. Fixed-parameter tractability or parameterized intractability crucially depends on the choice of the parameter, which can be anything computationally relevant: the value of an optimal solution, lower or upper bounds on the size of some structure of the instance, or some edit distance from general instances to efficiently solvable instances.

For some intractable problems moderately exponential-time algorithms have long been known to exist, and some such algorithms require only pseudo-polynomial time. The first moderately exponential-time algorithms date back to the early 1960s, and some early examples include

- an $O(1.4422^n)$-time algorithm for 3-COLORING by Lawler [181];
- an $O(1.2599^n)$-time algorithm for INDEPENDENT SET by Tarjan and Trojanowski [235];
- an $O(1.4142^n)$-time algorithm for SUBSET-SUM problems with $n$ integers by Horowitz and Sahni [159].

For each of these problems, a brute-force algorithm requires time $2^n \cdot n^{O(1)}$. A trigger to the systematic study of moderately exponential-time algorithms were surveys by Woeginger [248, 250], Iwama [162], Schöning [226] and Fomin et al. [107]. The field flourished immensely in the past ten years, with several PhD theses devoted to it [48, 122, 182, 187, 224, 228, 242, 247]. This list could be extended considerably by adding theses focusing on parameterized algorithms. As of today, for several intractable problems it remains open whether they admit moderately exponential-time algorithms. And certainly, once an algorithm is found, algorithms with faster running times are aimed for.

Motivation for designing moderately exponential-time algorithms comes from a variety of sources. Major interest stems from applications where exact solutions are the only plausible result, such as determining the satisfiability of a formula or the feasibility of equation systems. Besides, reducing the base of exponential running times, say from $O(2^n)$ to $O(1.99^n)$, increases the size of instances solvable within a
given amount of time by a multiplicative factor. In contrast, expediting computers often only allows solving additively larger instances within the same amount of time. Further stimuli are captured by Alan Perlis’ preference for exponential-time algorithms over efficient algorithms. His statement brings to the heart that for many computational problems one can trade running time of algorithms with quality of solutions produced by them; for instance as follows.

First, it can be that for an intractable problem $\Pi$ the quality of solutions returned by polynomial-time algorithms is too poor to be of any value. In particular, inapproximability results modulo conjectures such as $P \neq NP$ provide bounds on the solution quality returned by any efficient algorithm. In such cases, moderately exponential-time algorithms might be the only feasible approach to attack instances of $\Pi$. For example, INDEPENDENT SET has no polynomial-time approximation algorithm providing an $O(n^{1-\varepsilon})$-guarantee on $n$-vertex graphs for any $\varepsilon > 0$, assuming that $P \neq NP$ [151].

Second, it can be that the quality of solutions increases “continuously” with the available running time. For example, for VERTEX COVER there are several polynomial-time 2-approximations available and the factor 2 is conjectured to be best possible. On the other hand, better approximation factors of 3/2 and 5/3 can be obtained in time $O(1.0883^n)$ and $O(1.0381^n)$, respectively [100].

Third, even for problems in $P$ it can be advantageous to employ moderately exponential-time algorithms for their solution. For instance, an algorithm running in time $2^{n/10}$ is faster than an $n^5$-time algorithm for $n \leq 439$, whereas for $n > 439$ the estimated computation time is around two days (assuming a performance of $2^{30}$ operations per second).

That said, there is an immense potential for moderately exponential-time algorithms from the practical point of view, and with this thesis we hope to give some contributions.

1.1 Techniques for Moderately Exponential-Time Algorithms

We sketch some fundamental methods for the design of moderately exponential-time algorithms. Many more techniques are available, and the fast pace with which the field develops steadily yields new approaches. Further, it is sometimes a question of perspective whether a certain technique is a concept of its own or a variant of another, and techniques often come from or find applications in areas other than moderately exponential-time algorithms. Definitions and notations are explained in Section 1.3.

1.1.1 Search Trees

Search trees, or branching algorithms, are among the oldest tools for moderately exponential-time algorithms; their first usage dates back to the 1960s.

We explain the idea by means of the 3-SAT problem, which for a given Boolean formula $F$ with at most three literals per clause seeks a truth assignment to the variables $x_1, \ldots, x_n$ in $F$ satisfying all the clauses $C_1, \ldots, C_m$. A simple algorithm is
to iteratively build a search tree, by (1) creating a root node, (2) creating branches "$x_1 \rightarrow \text{true}$" and "$x_1 \rightarrow \text{false}$" at the root, and (3) for each leaf of the tree constructed so far and $i = 2, \ldots, n$ creating branches "$x_i \rightarrow \text{true}$" and "$x_i \rightarrow \text{false}$". The search tree constructed in this way has $O(2^n)$ leaves, each leaf corresponding to a truth assignment of $F$, and the algorithm building the tree is hence asymptotically not faster than exhaustive search.

A simple change of perspective improves the running time to $O(1.8393^n)$: we branch on clauses instead of variables. For a clause $C$ with three literals $\ell_1, \ell_2, \ell_3$, every satisfying truth assignment for $F$

(1) either satisfies literal $\ell_1$,

(2) or does not satisfy literal $\ell_1$ but satisfies literal $\ell_2$,

(3) or does not satisfy literals $\ell_1, \ell_2$ but satisfies literal $\ell_3$.

We fix the values of the corresponding one, two, or three variables appropriately, and branch into three subtrees according to cases (1), (2), and (3) with $n - 1$, $n - 2$, and $n - 3$ unfixed variables, respectively. By doing this, we cut away the subtree where all three literals $\ell_1, \ell_2, \ell_3$ are false. The formulas in the three subtrees are handled recursively. The stopping criterion is when we reach a formula in which each clause contains at most two variables, for which the existence of a satisfying assignment can be determined in polynomial time. The worst-case running time of this algorithm is within a polynomial factor of the number $T(n)$ of leaves of the search tree, where

$$T(n) \leq T(n - 1) + T(n - 2) + T(n - 3).$$

Therefore $T(n) \leq \alpha^n$, where $\alpha < 1.8393$ is the largest real root of the polynomial $\alpha^3 - \alpha^2 - \alpha - 1$. The currently fastest search-tree based algorithm for 3-SAT runs in time $O(1.4963^n)$ [179].

In parameterized complexity, search trees of bounded size can provide simple fixed-parameter algorithms (see also Section 1.1.4). A bounded search tree algorithm for ($k$)-Vertex Cover, due to Mehlhorn [193], works as follows. First observe that a graph $G$ has a vertex cover of size 0 if and only if $G$ is edgeless. Consider now an instance $(G, k)$ of ($k$)-Vertex Cover with $k > 0$ and an edge $\{u, v\} \in E(G)$. Any vertex cover $S$ of $G$ contains either $u$ or $v$. A set $S$ that contains a vertex $x$ is a vertex cover of $G$ if and only if $S \setminus \{x\}$ is a vertex cover of $G - x$, since vertex $x$ covers all edges incident to it. Hence $G$ has a vertex cover of size at most $k$ if and only if $G - u$ or $G - v$ has a vertex cover of size at most $k - 1$. Thus there is a polynomial-time reduction from the instance $(G, k)$ to two instances $(G - u, k - 1), (G - v, k - 1)$. As the depth of the search tree is bounded by $k$, this algorithm runs in time $O(2^k \cdot |V(G)| + |E(G)|)$.

Once a simple and intuitive search tree has been designed, reductions on its size are obtained by preprocessing the input, and more involved branching rules exploiting the special structure of the preprocessed input.
1.1.2 Measure and Conquer

Measure and Conquer provides a more detailed analysis of search tree algorithms, as to narrow the gap between worst-case upper and lower bounds on their running times. It can sometimes also support the design of improved search tree algorithms. Its idea is to let the input structure reflect on the “size” of an instance, by choosing a non-standard measure that is analyzed as a potential function.

An example is the algorithm by Fomin et al. [109] that determines the size of a maximum independent set in a graph $G$ on $n$ vertices. For $G$ connected in which there are no vertex pairs $v, w$ with $N[w] \subseteq N[v]$, the algorithm first folds 2-vertices $v$ with non-adjacent neighbors $u, w$, by creating a new vertex $v'$ and edges from $v'$ to all vertices of $N(u) \cup N(w)$, and finally removing $N[v]$. This operation decreases the size of a maximum independent set by exactly one. If no 2-vertices are foldable then the algorithm branches on a vertex $v$ of maximum degree with minimum number of edges in its open neighborhood, returning the size $\text{mis}(G)$ of a maximum independent set in $G$ by

$$\text{mis}(G) = \max\{\text{mis}(G - \{v\} - M(v)), 1 + \text{mis}(G - N[v])\},$$

where $M(v) = \{u \in N^2(v) \mid N(v) \setminus N(u) \text{ is a clique}\}$. This completes the algorithm.

Its running time is analyzed by letting the size of an instance $G$ to be the measure $\mu(G) = \sum_{d \geq 0} w_d n_d$, where $n_d$ is the number of $d$-vertices in $G$ and $w_d \in [0, 1]$ is their associated weight. This measure allows to assess structural changes of the graph caused by the branching (here the decrease of the degrees of the vertices) more accurately. The actual weights $w_d$ are obtained by optimizing a quasi-convex function of the weights over a set of nearly 5 million recurrences, that are imposed by initial assumptions on the weights and a detailed analysis of the decrease in measure caused by the branching. For instance, by assuming that $w_d \leq w_{d+1}$, the measure intuitively considers graphs with small average degree as easier. Another assumption is that for all $d_1, d_2 \in \{2, 3, \ldots, 8\}$,

$$w_2 + w_{d_1} + w_{d_2} - w_{d_1 + d_2 - 2} = w_{d_1} + w_{d_2} - w_{d_1 + d_2 - 2} \geq 0,$$

to avoid an increase in problem size when folding 2-vertices. With initial assumption that

$$w_0 = w_1 = w_2 = 0, \quad w_i = 1 \text{ for all } i \geq 7,$$

Fomin et al. find as optimum the weights

$$(w_3, w_4, w_5, w_6) = (0.620196, 0.853632, 0.954402, 0.993280),$$

leading to an overall running time of the algorithm of $O(1.22688^n)$.

Measure and Conquer analysis is particularly useful to analyze the progress of branching algorithms when including (excluding) one object into (from) the solution does not have immediate effects on other objects being included into (excluded from) a solution. An example is the Feedback Vertex Set problem, and up to now only non-standard measures of the input size proved the existence of $o(2^n)$-time algorithms on directed [214] and undirected [106, 215] $n$-vertex graphs. Recently,
the first application of measure and conquer to a parameterized problem was given, for the \((k)\)-SET SPLITTING \([186]\) problem.

1.1.3 Dynamic Programming

Dynamic programming across subsets is used whenever an optimal solution to an instance can be built by combining optimal solutions to subinstances no matter how the solutions to the subinstances were obtained. The idea of this technique is to store, for each subset of a ground set on \(n\) elements, a partial solution (and often auxiliary information) to the problem in a huge table so that the partial solutions can be looked up quickly.

For example, the dynamic programming algorithm by Held and Karp \([155]\) for the TRAVELING SALESMAN PROBLEM expedites, for instances with \(n\) cities, the exhaustive search over \(n!\) permutations to time \(O(2^n n^2)\), and up to today no algorithm with better time complexity is known for this problem. It works as follows. For every non-empty subset \(S \subseteq \{2, \ldots, n\}\) and every city \(i \in S\), let \(\text{opt}[S; i]\) denote the length of a shortest path that starts in city 1, then visits all cities in \(S \setminus \{i\}\) in arbitrary order, and finally stops in city \(i\). Then \(\text{opt}[[i]; i] = d(1, i)\) and \(\text{opt}[S; i] = \min \{\text{opt}[S \setminus \{i\}; j] + d(j, i) \mid j \in S \setminus \{i\}\}\), where \(d(i, j)\) denotes the distance between cities \(i\) and \(j\). By working through the subsets \(S\) in order of increasing cardinality, we can compute the value \(\text{opt}[S; i]\) in time proportional to \(|S|\). The optimal travel length is given as the minimum value of \(\text{opt}[[2, \ldots, n]; j] + d(j, 1)\) over all \(j \in \{2, \ldots, n\}\).

A disadvantage of many dynamic programming algorithms working in this “bottom-up” manner is that they need to store exponentially many solutions to subproblems.

Also for the design of fixed-parameter algorithms (see next subsection), dynamic programming finds abundant applications. For example, a great deal of NP-hard problems on graphs of bounded treewidth are solved by dynamic programming over a tree-decomposition \([175, 236]\).

1.1.4 Fixed-Parameter Algorithms

Parameterized complexity is a special case of what one might call a “multivariate” approach to complexity analysis and algorithm design \([94, 206]\). Here, in addition to the overall input size \(n\), a secondary measurement \(k\) (the parameter) is considered, where one expects the parameter \(k\) to be significantly smaller than \(n\) and to capture information about the structure of typical inputs or other aspects of the problem situation that affect its computational complexity. In the familiar “classical” one-dimensional approach, the central concept is polynomial time \((P)\), “the good class”. In the parameterized complexity framework the central notion is fixed-parameter tractability (FPT), defined to be solvability in time \(f(k)n^{O(1)}\), where \(f\) is a computable function dependant only on \(k\). In the classical framework, an algorithm with running time in \(P\) is the desirable outcome, as contrasted with the possibility that only running times of the form \(2^{cO(1)}\), the “bad class”, might be achievable. Classical complexity analysis unfolds in the contrast between these two univariate function classes.
1.1. TECHNIQUES FOR MODERATELY EXPONENTIAL-TIME ALGORITHMS

Parameterized complexity analysis unfolds analogously in the contrast between the “good class” of bivariate functions FPT, and the “bad class” of running times of the form \( O(n^{f(k)}) \) (Solvability in such time defines the parameterized complexity class XP.). To emphasize the contrast, one could also consider defining \( \text{FPT} \) additively as solvability in time \( f(k) + n^{O(1)} \). It turns out that this makes no difference qualitatively: a parameterized problem is additively \( \text{FPT} \) if and only if it is \( \text{FPT} \) by the usual definition [82, 87]. (However, quantitatively, in classifying a parameterized problem as additively \( \text{FPT} \), it might be necessary to use a “larger” function \( f(k) \).

The basic contrast in parameterized complexity is thus concerned with whether any exponential costs of the problem can be confined to the parameter.

In the classical framework, evidence that a problem is unlikely to have an algorithm with a running time in the good class is given by determining that it is \( \text{NP} \)-hard, \( \text{PSPACE} \)-hard, \( \text{EXP} \)-hard, etc. In parameterized complexity analysis there are analogous means to show likely parameterized intractability. The current tower of the main parameterized complexity classes is

\[
\text{FPT} \subseteq \text{M}[1] \subseteq \text{W}[1] \subseteq \text{M}[2] \subseteq \text{W}[2] \subseteq \ldots \subseteq \text{W}[P] \subseteq \text{XP};
\]

see p. 31 for their definitions.

The \((k)\)-independent set problem is complete for \( \text{W}[1] \) [83], and \((k)\)-dominating set is complete for \( \text{W}[2] \) [84]. The best known algorithms for \((k)\)-independent set and \((k)\)-dominating set are slight improvements on the brute-force approach of trying all \( k \)-subsets, and run in time \( n^{O(k)} \) [202]. The \((k)\)-bandwidth problem is hard for \( \text{W}[t] \) for all \( t \geq 1 \) [35]. The parameterized class \( \text{W}[1] \) is strongly analogous to \( \text{NP} \), because the \((k)\)-step halting problem is complete for \( \text{W}[1] \) [50, 85]. \( \text{FPT} \) is equal to \( \text{M}[1] \) if and only if the Exponential Time Hypothesis fails [81, 160]. There is an algorithm for the \((k)\)-independent set problem that runs in time \( n^{o(k)} \) if and only if \( \text{FPT} = \text{M}[1] \), and there is an algorithm for the \((k)\)-dominating set problem that runs in time \( n^{o(k)} \) if and only if \( \text{FPT} = \text{M}[2] \) [53].

There are numerous useful recent surveys about parameterized complexity and algorithm design [1, 86, 87, 97, 137, 204, 212]; one can also turn to the books and monographs by Downey and Fellows [82], Flum and Grohe [104], or Niedermeier [205] for further background.

Further motivation for the subject of parameterized complexity and algorithms has come from the parameterized graph minor problem, which asks for given graphs \( G \) and \( H \) whether \( H \) is a minor of \( G \), for parameter \( H \). To show that this fundamental problem is \( \text{FPT} \) requires, according to present knowledge, the entire panoply of the graph minors structure theory by Robertson and Seymour [222]. This structure theory is of high practical relevance, since, for one example, many naturally occurring databases have bounded treewidth (or bounded “hyper-tree-width”, a related notion). This provides significant inroads for hard database problems [103, 128, 130], but bounded treewidth seems to be an almost universally relevant parameter.

Fixed-parameter algorithms and moderately exponential time algorithms are closely related. On the one hand, many \( \text{FPT} \)-algorithms run in moderately exponential time, such as when small upper bounds on the maximum value of the parameter in terms of the input size are available [213, Theorem 24]. Further, \( \text{FPT} \)-
algorithms can serve as a tool for the design of moderately exponential time algorithms, for instance by employing FPT-algorithms for small parameter values and brute force subset enumeration for large parameter values [213, Theorem 16]. On the other hand, fast moderately exponential time algorithms can lead to fast FPT-algorithms for fixed-parameter tractable problems, when applied to kernels of small size—see the following subsection.

1.1.5 Kernelization

The idea of kernelization is to efficiently compress arbitrary instances of a parameterized problem $\Pi$ to equivalent instances of $\Pi$ whose size is bounded by a function of the parameter $k$ alone. Such compressed instance we call a kernel for $\Pi$. Kernelization, and more general polynomial-time preprocessing, is key in solving large instances of intractable problems in practice [137, 141]. If the parameter is small then the kernel can practically be solved by moderately exponential-time algorithms, and sometimes even by exhaustive search. By the size of a kernel we can measure the quality of the kernelization, in general the smaller the kernel the better.

The following well-known proposition codifies how every fixed-parameter tractable problem has a canonically associated structure theory project, via the quest for efficient FPT kernel size bounds.

**Proposition 1.1** ([82, 104]). A parameterized problem $\Pi$ is solvable in time $f(k) \cdot n^{O(1)}$ if and only if $\Pi$ is decidable and has a kernel of size $g(k)$, for computable functions $f$ and $g$.

**Proof.** If $\Pi$ is decidable and has a kernel of size $g(k)$ then instances of $\Pi$ can be decided in $f(k) \cdot n^{O(1)}$ time, by first producing the kernel in polynomial time and thereafter deciding the kernel, whose size depends only on $k$. Conversely, suppose that some algorithm $A$ decides instances $x$ of $\Pi \subseteq \Sigma^*$ with size $n$ in time $f(k) \cdot n^c$ time, for some constant $c \geq 1$ independent of $k$. Apply $A$ to $x$ for at most $(n^c)^2$ steps. If $A$ accepts (rejects) $x$ then accept (reject) $x$ by outputting a “yes”-instance (“no”-instance) $K(x)$ of constant size (at most $f(k)$). Otherwise, output $K(x) := x$ and note that in this case, $n \leq n^c \leq f(k)$. Now $K(x)$ is a kernel for $\Pi$, as it has been obtained in polynomial time, its size is bounded by $f(k)$, and $x \in \Pi$ if and only if $K(x) \in \Pi$. □

The proof yields a kernel bound of only $g(k) = f(k)$, and therefore often the kernels obtained by this general result have impractically large size. However, the shift of perspective that the proposition codifies is useful and important.

It is by now a research field of its own to on the one hand obtain smaller and smaller kernels for fixed-parameter tractable problems, and on the other hand to look for larger and larger classes of instances for which kernels of small size, like polynomial or linear in $k$, can be obtained. A kernel is usually obtained in a two-step process. First, reduction rules are designed which in polynomial time remove local structures from the input $(x, k)$ to obtain an equivalent instance $(x', k')$ where $k'$ is bounded by a function of $k$. Second, by analyzing reduced instances $(x', k')$ in which none of the local structures is present, a bound on the size of $x'$ in terms of $k$ is derived. For example, two reduction rules for $(k)$-Vertex Cover (attributed to Buss [82]) are to
(1) remove isolated vertices from the graph and set \( k' = k \),

(2) remove vertices of degree 1, or degree strictly larger than \( k \), and their neighbors from the graph and set \( k' = k - 1 \).

Now observe graphs of minimum degree 2, maximum degree \( k \), and a vertex cover of size at most \( k \), can have at most \( k^2 + k \) vertices and at most \( k^2 \) edges. Thus, \((k)\)-VERTEX COVER has a kernel of size \( O(k^2) \), which can be improved by more elaborate methods [2, 97, 201].

By devising clever reduction rules, we can often achieve strikingly non-exponential bounds on the kernel sizes, and the polynomial-time preprocessing routines that produce small kernels have proven practical value [2, 204, 205, 244]. The initial bounds on the kernel sizes, and the polynomial-time preprocessing routines that do not have a kernel with less than 1.36\( n \) vertices, as such would yield a 1.36-approximation of VERTEX COVER implying that \( P = \text{NP} \) [76].

### 1.1.6 Chromatic Coding

Consider the \((k)\)-PATH problem of finding a path of length at least \( k \) in a graph \( G \) with \( n \) vertices. It was a question of Papadimitriou and Yannakakis [208] whether for \( k = \log n \), this problem can be solved in polynomial time. The question was affirmatively answered by Alon et al. [20], by introducing a randomized algorithmic technique termed color coding. Its principle is to randomly color the vertices of \( G \) and then to search for a path no two vertices of which receive the same color. The coloring provides sufficient additional input structure to perform this search in polynomial time by dynamic programming. After sufficiently many random colorings the answer to the original question is correct with high probability.

A variant of color coding is chromatic coding [18], where compared with the original version, the number of used colors is much larger. There, the algorithm can be derandomized via “universal coloring families”, a class of hash functions. For integers \( m, k \) and \( r \), a family \( \mathcal{F} \) of functions from \( \{1, \ldots, m\} \) to \( \{1, \ldots, r\} \) is called a universal coloring family if for any graph \( G \) with \( V(G) = \{1, \ldots, m\} \) and \( |E(G)| \leq k \) there exists an \( f \in \mathcal{F} \) which is a proper vertex-coloring of \( G \). There are explicit constructions for \((10k^2, k, O(\sqrt{k}))\)-coloring families of size \( |\mathcal{F}| \leq 2^{O(\sqrt{k})} \) and universal \((n, k, O(\sqrt{k}))\)-coloring families of size \( |\mathcal{F}| \leq 2^{O(\sqrt{k}) \log n} \), where \( O(\sqrt{k}) = O(\sqrt{k} \log k)^{O(1)} \). The chromatic coding technique was used to obtain subexpo-
ential fixed-parameter algorithms for the problems $(k)$-Feedback Arc Set in tournaments [18], Betweenness parameterized by the number of constraints to be removed on instances with exactly one constraint per 3-set of variables [225], and $(k)$-Quartet Inconsistency [225].

1.1.7 Well-Quasi Ordering

Well-quasi ordering is a powerful tool to classify parameterized problems as fixed-parameter tractable.

For a set $S$, a quasi-order $\preceq$ on $S$ is a reflexive transitive subset of $S \times S$. A quasi-order is well-founded if it contains no infinite strictly descending sequences, that is, infinite sequences $s = (s_i)_{i \in \mathbb{N}}$ of elements from $S$ with $s_i > s_{i+1}$ for all $s_i \in s$. A good sequence of $\preceq$ is an infinite sequence $(s_i)_{i \in \mathbb{N}}$ of elements from $S$ which contains a good pair, that is, a pair $(s_i, s_j)$ with $s_i \preceq s_j$ and $i < j$. A bad sequence is an infinite sequence that is not good. A well-quasi order on $S$ is a well-founded quasi-order without infinite “anti-chains”. An anti-chain under $\preceq$ is a sequence $(s_i)_{i \in \mathbb{N}}$ of elements from $S$ such that for every $i \neq j$ it holds $s_i \neq s_j$ and neither $s_i \preceq s_j$ nor $s_j \preceq s_i$.

Well quasi orders $\preceq$ on sets $S$ are interesting because they imply that all “closed” subsets of $S$ have finite “forbidden characterizations”. A subset $S' \subseteq S$ is closed under $\preceq$ if for all $s_1, s_2 \in S'$ with $s_1 \preceq s_2$ also $s_2 \in S'$. A forbidden characterization of a subset $S' \subseteq S$ under $\preceq$ is a set $\text{Forb}(S') \subseteq S$ such that for any $s \in S$ it holds $s \in S'$ if and only if $t \not\preceq s$ for all $t \in \text{Forb}(S')$. Observe that if $S'$ is closed under $\preceq$ then $S \setminus S'$ is a forbidden characterization of $S'$ under $\preceq$, where the set of all $\preceq$-minimal elements of $S \setminus S'$ is the unique minimal forbidden characterization. Now if $\preceq$ is a well-quasi order on $S$ then this set must be finite as it constitutes an anti-chain under $\preceq$.

The Graph Minor Theorem [223] states that the set of finite graphs is well-quasi-ordered under the minor relation. Moreover, the $H$-minor problem that checks if $H$ is a minor of some input graph is non-uniformly fixed-parameter tractable for parameter $k = |V(H)|$. As a consequence, any parameterized problem $(\Pi, k)$ whose “yes”-instances are closed under minors for each parameter value $k$ is non-uniformly fixed-parameter tractable. A uniformly fixed-parameter algorithm can be obtained if, additionally, an explicit list of the graphs in the minimal forbidden characterization $\text{Forb}(S')$ of $S'$ is known, as well as a uniform polynomial-time algorithm testing for each graph $H \in \text{Forb}(S')$ whether $H$ is a minor of a given graph $G$.

For example, if $(G, k)$ is a “yes”-instance for $(k)$-Feedback Vertex Set and some graph $H$ is a minor of $G$, then $(H, k)$ is a “yes”-instance for $(k)$-Feedback Vertex Set. If $M_k$ denotes the set of minimal forbidden minors for the class of graphs with feedback vertex sets of size at most $k$ then $M_k$ is an anti-chain under the minor order and so it is finite. Now $(G, k)$ is a “yes”-instance of $(k)$-Feedback Vertex Set if and only if there is no graph $H \in M_k$ that is a minor of $G$. As the size of $M_k$ only depends on $k$ but not on $G$, we can decide $(G, k)$ by checking for $|M_k|$ graphs whether they are a minor of $G$. Therefore, $(k)$-Feedback Vertex Set is non-uniformly fixed-parameter tractable. In this case, the set $M_k$ is computable by the
method of Adler et al. [3], because the set of minimal forbidden minors for the class
of trees is explicitly known. Since their method also yields a computable function f
that for a given graph $G$ and an integer $k$ decides in time $f(k) \cdot n^3$ whether $G$ has
a set $X$ of at most $k$ vertices such that $G - X$ is a forest, the problem $(k)$-Feedback
Vertex Set is strongly uniformly fixed-parameter tractable.

1.1.8 Enumeration
An elegant way of obtaining moderately exponential-time algorithms for an opti-
mization problem $\Pi$ is to prove a non-trivial upper bound on the number of “local
optima” in any instance of $\Pi$, and then show that these local optima can be listed in
time faster than exhaustive search. The choice of what constitutes a local optimum
has to be made carefully: if it is defined too loose then only weak bounds on the
time to list all local optima can be proven, whereas if it is defined too restrictive
then finding one local optimum might itself be an intractable problem.

As we are dealing with intractable problems, usually the set $S$ of local optima of
$\Pi$ will be of size that is superpolynomial in the input size. We measure the time to
list the elements in $S$, for inputs $x$ of size $n$, as follows [165].

**Polynomial total time.** An algorithm runs in polynomial total time if it lists all ele-
ments of $S$ in time that is bounded by a polynomial in $n$ and $|S|$. 

**Incremental polynomial time.** An algorithm requires incremental polynomial time if,
given $x$ and a subset $S' \subseteq S$, outputs an element of $S \setminus S'$ (or determines that
no such element exists) in time that is polynomial in $n$ and $|S'|$. Observe that
any algorithm with this property also runs in polynomial total time.

**Polynomial delay.** An algorithm $A$ runs with polynomial delay if the time until it
first outputs an element of $S$, the time between outputting any two elements
of $S$ and the time it requires after having listed all elements of $S$, are all poly-
nomial in $n$. There is no requirement on the order in which the elements of $S$
are listed; and observe that if $A$ runs with polynomial delay then it requires
incremental polynomial time.

An orthogonal requirement to the time needed by an algorithm to list all elements
of $S$ is its space requirement; ideally we would like the space demand to be poly-
nomial in the input size $n$.

An example application is the Independent Set problem: for an instance a
graph $G$, we let the set of local optima be the collection of maximal independent sets
of $G$. Any graph on $n$ vertices has at most $3^{n/3}$ maximal independent sets [198], far
fewer than the $2^n$ vertex subsets. Further, the maximal independent sets of a graph
can be listed with polynomial delay and polynomial space [238]. We conclude
that a maximum independent set of $G$ can be found in time $O^*(3^{n/3})$ and space
polynomial in $n$, where $n = |V(G)|$.

1.1.9 Approximation Algorithms
Approximation algorithms can help to design moderately exponential-time algo-
rithms in several ways.
First, for problems parameterized by the value of a solution, an approximation algorithm can efficiently narrow the range of the parameter to be considered. For example, to solve the \((k)\)-Feedback Vertex Set problem on planar graphs \(G\) one can start by computing a tree-decomposition of \(G\) of width \(t\) using a factor-\(\alpha\) approximation algorithm [70]. Now if \(t > a\alpha\sqrt{k}\), where \(\alpha > 0\) is a fixed constant, then \(G\) has a \((4\sqrt{\alpha} + 1) \times (4\sqrt{\alpha} + 1)\)-grid minor. As every vertex in this grid minor breaks at most four cycles, this means that \((G,k)\) is a “no”-instance. Otherwise, \(t \leq a\alpha\sqrt{k}\), and the instance can be decided by dynamic programming over the tree decomposition.

Second, exact algorithms might use the structure of a solution returned by an approximation algorithm. For instance, Vertex Cover has a 2-approximation algorithm that for a given graph \(G\) returns a tripartition \(\{V_0, V_{1/2}, V_1\}\) of \(V(G)\) such that

- the union of \(V_1\) and a minimum vertex cover of the graph induced by \(V_{1/2}\) is a minimum vertex cover of \(G\), and
- minimum vertex covers of the graph induced by \(V_{1/2}\) have size at least \(|V_{1/2}|/2\).

This structural insight is due to Nemhauser and Trotter [201]. The tripartition \(\{V_0, V_{1/2}, V_1\}\) corresponds to a tripartition of the variables, with respect to their assigned values, in an optimal half-integral solution of a linear programming relaxation of an integer programming formulation of Vertex Cover. Now a fixed-parameter algorithm for \((k)\)-Vertex Cover can, given \((G,k)\) together with \(\{V_0, V_{1/2}, V_1\}\), identify \((G,k)\) as a “no”-instance if \(|V_{1/2}| > 2(k - |V_1|)\), and otherwise decide the instance by examining the at most \(2(k - |V_1|)\) \(2\alpha\delta\) vertices of \(V_{1/2}\).

Third, by weakening the requirement that approximation algorithms must be efficient, we obtain exponential-time approximation algorithms. Such algorithms, for which efficiency cannot be guaranteed, can be advantageous provided that the expense in running time compared to polynomial-time approximation algorithms is made up by a better approximation factor. For Bandwidth, the currently fastest algorithm that returns an optimal bandwidth layout takes time and space \(O(4.473^n)\) on graphs with \(n\) vertices [65]. To approximate the optimal bandwidth within a factor of 2, one may use an algorithm by Fürer et al. [119] that runs in \(O(1.9797^n)\) time and polynomial space. No polynomial-time constant-factor approximation algorithm for the optimal bandwidth exists even for caterpillars [239], unless \(P = NP\).

Exponential-time approximation algorithms, with time complexity measured in the input size, have a natural bidimensional analogue with fixed-parameter approximation algorithms. For a parameterized problem \(\Pi\), consider the following algorithmic problem:

\[ \Pi \, g(k)\text{-Approximation} \]

Input: An instance \((x,k)\) of \(\Pi\).

Parameter: \(k\).

Output: Either that \((x,k)\) is a “no”-instance for \(\Pi\), or a certificate that \((x,g(k))\) is a “yes”-instance for \(\Pi\).

A fixed-parameter approximation algorithm for \(\Pi\) is an algorithm that for every instance \((x,k)\) of \(\Pi \, g(k)\text{-Approximation}\) returns one of the desired outputs in time \(f(k) \cdot |x|^{O(1)}\), for some computable function \(f\). The existence of such an
algorithm strongly depends on the choice of the function $g$. For certain functions $g$, fixed-parameter approximation algorithms have been developed for $(k)$-Vertex Cover [100], $(k)$-Vertex Disjoint Cycles [129], $(k)$-Edge Multicut [192] and $(k)$-Topological Bandwidth [94].

For Vertex Cover, the best approximation factors achievable in polynomial time are 1.36 (unless $P = NP$ [76]) and 2 (unless the Unique Games Conjecture fails [172]). The currently fastest fixed-parameter algorithm for $(k)$-Vertex Cover runs in time $O(1.2738^k)$ [56]. The fixed-parameter approximation algorithm by Fernau et al. [100] trades time with approximation guarantees: for $g(k) = \frac{2}{3}k$ it solves $(k)$-Vertex Cover $g(k)$-APPROXIMATION in time $1.0883^k \cdot n^{O(1)}$, whereas for $g(k) = \frac{5}{3}k$ the algorithm runs in time $1.0381^k \cdot n^{O(1)}$.

The $(k)$-Vertex Disjoint Cycles problem is $W[1]$-hard [229], and hence unlikely to be fixed-parameter tractable. A polynomial-time algorithm by Grohe and Grüber [129] returns a set of $k/\rho(k)$ vertex-disjoint cycles, if the input graph $G$ has $k$ vertex-disjoint cycles. Here, $\rho$ is a computable function such that $k/\rho(k)$ is non-decreasing and unbounded. If $G$ does not contain $k$ vertex-disjoint cycles then the output of their algorithm is arbitrary.

Some $W[1]$-hard parameterized decision problems also do not admit fixed-parameter approximation algorithms with good approximation factors, modulo an unexpected collapse of complexity classes. For instance, Downey et al. [86] showed that $(k)$-Dominating Set $g(k)$-APPROXIMATION is $W[2]$-hard if $g(k) = k + c$, for any integer $c \geq 0$.

For a more extensive treatise of the connections between approximation algorithms and parameterized algorithms we refer to the survey of Marx [191].

### 1.1.10 Lower Bounds on Running Times

There are lower bounds on time complexities to solve intractable problems, and there are lower bounds on running times of algorithms for intractable problems.

Lower bounds on time complexities to solve intractable problems are based on assumptions about the distinctness of complexity classes, and reductions controlling the size of instances or parameters. Unless $P = NP$, no NP-complete problem can be solved in polynomial time. Unless $\text{FPT} = W[t]$, no $W[t]$-complete parameterized problem can be solved by a fixed-parameter algorithm. Thus, strong evidence that a problem $\Pi$ cannot be solved in polynomial time is a polynomial-time reduction to $\Pi$ from an NP-hard problem. Similarly, evidence that a parameterized problem $\Pi$ is not fixed-parameter tractable is a parameterized reduction from a $\text{W}[t]$-hard problem, for some $t \geq 1$.

The Exponential-Time Hypothesis says that there is no algorithm solving 3-SAT in time $2^{o(n)}$. Evidence that a problem $\Pi$ cannot be solved in time $2^{o(n)}$ is a polynomial-time reduction from 3-SAT translating formulas with $n'$ variables into instances of $\Pi$ of size $n = O(n')$.

It was further shown by Chen et al. [53] that unless all problems in $\text{SNP}$ can be solved in subexponential time, there is no computable function $f(k)$ such that $(k)$-CLIQUE or $(k)$-INDEPENDENT SET can be solved in time $f(k)n^{o(k)}$, and unless $\text{FPT} = \text{W}[1]$, there is no computable function $f(k)$ such that $(k)$-DOMINATING SET...
1. INTRODUCTION

Algorithm-specific lower bounds on running times help to understand how far the predicted worst-case running time of an algorithm deviates from its actual running time. They potentially also guide the quest for faster algorithms, by indicating instances that are hard to solve for the algorithm. To show lower bounds on running times of a search-tree based algorithm, one constructs instances such that the algorithm branches on the same structure on every node of a path in a search tree from the root to a leaf. For algorithms based on enumerating local optima, lower bounds on their running time can possibly be obtained by constructing infinite families of instances with many local optima. Brute-force and dynamic-programming based algorithms usually take the same amount of time for all instances of the same size, regardless of the actual input structure.

1.2 Results and Outline of this Thesis

In this thesis we present results on moderately exponential-time algorithms from three different perspectives.

Faster Running Times. Faster running times of moderately exponential-time algorithms usually far outweigh a reduction of computation times by applying more advanced computer technology. Doubling computing power, measured as the number of simple operations performable per second, often only allows to solve additively larger instances of intractable problems within the same amount of time. Whereas improving an algorithm with running time $\Omega(c^n)$ to one running in time $O(d^n)$ for a constant $d$ only slightly smaller than $c$ means that multiplicatively larger instances can be solved within the same amount of time. For some intractable problems, even exponential speed-ups compared to exhaustive search are conceivable, by reducing running times of $\Omega(c^n)$ to $c^{o(n)}$. In this thesis we describe algorithms whose running times outperform previously smallest time complexities by multiplicative and sometimes exponential factors.

- We give the fastest known algorithms for finding a minimum feedback vertex set in tournaments, in time $O(1.6740^n)$ and polynomial space. In fact, the algorithm lists the minimal feedback vertex sets in time $O(1.6740^n)$ and polynomial space.

- We give an algorithm for finding minimum feedback vertex sets in bipartite directed graphs in time $O(1.8621^n)$.

- We give a subexponential fixed-parameter algorithm for triplet inconsistency parameterized by solution size, running in time $2^{O(k^{1/3} \log k)} \cdot n^{O(1)}$.

Smaller Space Demands. Reduced time complexities are sometimes not sufficient to make a theoretically fast algorithm practically applicable, if the algorithm in question uses an exorbitant (read: exponential) amount of space. Physical limits on memory size possibly mean that algorithms with exponential space demands are “absolutely useless for real life applications” [249]. It is then sensible to trade
time for space, and design algorithms whose space demand is very moderately exponential, or polynomial, in the input size. Our contributions are algorithms for intractable problems whose time complexity is asymptotically no worse than that of previously fastest algorithms, but whose space demand is exponentially smaller.

- We present polynomial-space fixed-parameter algorithms with subexponential parameter dependence for almost all bidimensional problems on graphs excluding a fixed minor. Most notably, we give such an algorithm for the fundamental problem of finding a path of length $k$.

- We present a polynomial-space polynomial-delay algorithm for enumerating the minimal feedback vertex sets in tournaments.

Deeper Structure Analysis. In this thesis, we contribute new fixed-parameter algorithms as well as parameterized intractability results, for various problems from graph theory, constraint satisfaction and phylogenetics, and multiple parameter choices.

- We establish linear vertex-kernels for maximum leaf spanning tree in general graphs, and its parameterized dual, connected dominating set, in planar graphs and $K_{3,3}$-minor free graphs. For $H$-minor free graphs, we show that no polynomial kernel exists for connected dominating set parameterized by solution size and $|H|$, unless $PH = \Sigma^P_3$.

- We establish fixed-parameter tractability of bandwidth, and topological bandwidth, parameterized by the max leaf number of the input graph.

- We establish fixed-parameter tractability of all ternary permutation constraint satisfaction problems parameterized above tight lower bound on their solution value.

- We establish fixed-parameter tractability of independent set in planar graphs with maximum degree three parameterized above tight lower bound on the solution value.

The results are obtained by applying and extending techniques from graph theory, parameterized complexity, probability theory and extremal combinatorics. Most of them are accompanied by lower bounds on running times or kernel sizes, modulo standard complexity-theoretic assumptions.

Parts of the results are based on and variations of the following publications:


The remainder of this thesis is organized as follows. We first evoke the basic concepts from algorithms and computational complexity, in Section 1.3. In Chapter 2, we study the parameterized complexity of various problems on graphs with bounded max-leaf number. Chapter 3 presents polynomial kernels for the connected dominating set problem in sparse graph classes. Then in Chapter 4 we give combinatorial bounds and enumeration algorithms for feedback vertex sets in classes of directed graphs. Chapter 5 contains fixed-parameter algorithms for graph problems and constraint satisfaction problems parameterized above tight lower bounds on their solution value. In Chapter 6 we give subexponential fixed-parameter algorithms for a large class of problems on sparse graphs, and the minimum triplet inconsistency problem on dense instances. An outlook on the future evolvement of the field is given in Chapter 7.

1.3 Notions and Notations

This section collects the notions, and their notations, that repeatedly appear throughout the thesis.

Sets, Functions and Logic

The sets of integers, positive integers, reals and positive reals will be denoted by \( \mathbb{Z}, \mathbb{N}, \mathbb{R} \) and \( \mathbb{R}_+ \), respectively. By \( \mathbb{F}_2 \) we denote the smallest Galois field.

Let \( S \) be a finite set. A linear order of \( S \) is a binary relation on \( S \) that is transitive, antisymmetric and total. For a linear order \( \prec \) of \( S \) and any integer \( q \in \{1, \ldots, |S|\} \), let \( \prec^{-1}(q) \) be the element \( s \in S \) for which there are exactly \( q-1 \) elements \( s' \in S \).
with $s' \prec s$. Two linear orders $\prec, \prec'$ of $S$ are cyclically equivalent if there exists some $q \in \{1, \ldots, |S|\}$ such that $\mu - 1 \equiv (\nu - 1 + q) \mod |S|$ implies $\prec^{-1}(\nu) = \prec'^{-1}(\mu)$. A cyclic order of $S$ is an equivalence class of linear orders of $S$ modulo cyclic equivalence. For a family $\mathcal{F}$ of subsets of $S$, a set cover is a subfamily $\mathcal{F}'$ such that $\bigcup_{F \in \mathcal{F}'} F = S$.

We consider Monadic Second Order (MSO) logic on graphs in terms of their incidence structure whose universe contains vertices and edges. The atomic formulas are $E(x)$ ("$x$ is an edge"), $V(x)$ ("$x$ is a vertex"), $I(x, y)$ ("vertex $x$ is incident with edge $y$"), $E(x, y)$ ("vertices $x$ and $y$ are adjacent"), $x = y$ (equality) and $X(y)$ ("vertex $y$ is element of set $X$"). MSO formulas are built up from atomic formulas using the usual Boolean connectives ($\neg, \lor, \land, \rightarrow, \leftrightarrow$), quantification over variables ($\forall x, \exists x$) and quantification over sets ($\forall X, \exists X$).

### Problems

Let $\Sigma$ be a finite alphabet. A decision problem is a set $\Pi \subseteq \Sigma^*$ of strings over $\Sigma$. A string $s$ is a "yes"-instance for $\Pi$ if $s \in \Pi$ and a "no"-instance otherwise. A parameterization of a decision problem is a polynomial-time computable function $k : \Sigma^* \rightarrow \mathbb{N}$. A parameterized decision problem is a pair $(\Pi, k)$, where $\Pi \subseteq \Sigma^*$ is a decision problem and $k$ is a parameterization of $\Pi$. Intuitively, we can view a parameterized decision problem as a decision problem where each input instance $x \in \Sigma^*$ has a positive integer $k$ associated with it, and $k$ is referred to as the parameter of the instance $(x, k)$. For a parameterized problem $\Pi$ with input $(x, k)$ the unparameterized version is defined as $\tilde{\Pi} = \{x\#1^k \mid (x, k) \in \Pi\}$, where $\# \notin \Sigma$ is the blank letter and $1 \in \Sigma$ is arbitrary. By $\tilde{\Pi}$ we denote the set of all "no"-instances of $\Pi$.

An optimization problem is a tuple $(I, \mathcal{F}, c, \text{goal})$, where $I$ is a set of instances, $\mathcal{F}$ is collection of sets $\mathcal{F}(I)$ of feasible solutions for every instance $I \in I$, $c$ is a cost function assigning a cost, or value, $c(F)$ to every feasible solution $F \in \mathcal{F}(I)$, and $\text{goal} \in \{\text{min}, \text{max}\}$. When goal = min (goal = max) then we speak of a minimization (maximization) problem, where for each instance $I \in I$ we call a feasible solution $F \in \mathcal{F}(I)$ optimal if $c(F) = \min_{F' \in \mathcal{F}(I)} c(F')$ (if $c(F) = \max_{F' \in \mathcal{F}(I)} c(F')$). The cost of an optimal solution is the optimal value of instance $I$. For a minimization (maximization) problem $\Pi$ let $(k)$-$\Pi$ denote the parameterized decision problem that takes as input pairs $(I, k) \in I \times \mathbb{N}$ and asks whether the optimal value of $I$ is at most $k$ (at least $k$).

### Complexity Classes and Reductions

We use the familiar $O(\cdot)$ notation, and modifications thereof which suppress polynomially and polylogarithmically bounded terms:

$$O^*(f(n)) := \bigcup_{d \geq 1} O(f(n) \cdot n^d),$$

$$\tilde{O}(f(n)) := \bigcup_{d \geq 1} O(f(n) \cdot (\log n)^d).$$

Let $P$ denote the class of problems solvable in polynomial time by a deterministic
Turing machine, and let NP denote the class of problems solvable in polynomial time by a nondeterministic Turing machine. Let coNP denote the class of problems whose complement is in NP. Let NP/poly denote the class of problems Π for which there exists a problem Π′ ∈ NP, a polynomial p(·), and for each n ∈ N a string s_n of length p(n), such that any x ∈ Σ^∗ belongs to Π if and only if (x, s_n |x|) belongs to Π′.

Let EXP denote the class of decision problems solvable by a deterministic Turing machine in 2^{O(n)} time. Let SNP denote the class of decision problems reducible to a graph-theoretic predicate with only universal quantifiers over vertices, no existential quantifiers. Let PSPACE denote the class of decision problems solvable by a Turing machine using a polynomial amount of space.

A problem Π is NP-hard if there is a polynomial-time reduction from every problem in NP to Π, and Π is NP-complete if Π is NP-hard and belongs to NP. Similarly defined are coNP-hard, EXP-hard, PSPACE-hard and coNP-complete, EXP-complete, PSPACE-complete. Let PH = ⋃_{i≥1} Σ^P_i, where Σ^P_i is the class of decision problems solvable in polynomial time by an alternating Turing machine that starts with an existential configuration and is i-alternating [104], for all i ≥ 1.

For parameterized problems with instances (x,k) we assume that the parameter k is given in unary and hence k ≤ |x|^{O(1)}. Let XP denote the class of parameterized problems Π such that for every pair (x,k) ∈ Σ^* × N it can be decided in time n^{g(k)} whether (x,k) ∈ Π, for some computable function g. Let FPT denote the class of parameterized problems Π such that for every pair (x,k) ∈ Σ^* × N it can be decided in time f(k) · |x|^c whether (x,k) ∈ Π, for some computable function f and some constant c independent of k; such problems are called (strongly uniformly) fixed-parameter tractable. Originally, the class of fixed-parameter tractable problems as introduced by Downey and Fellows [82] does not require the function f to be computable; we refer to such problems as uniformly fixed-parameter tractable. Besides, we can relax the condition that there is one algorithm for all parameter values k, and consider the class of non-uniformly fixed-parameter tractable problems Π, for which there is a constant c and an algorithm family \{A_k\} such that for every pair (x,k) ∈ Σ^* × N, algorithm A_k decides membership of (x,k) in Π in time O(|x|^c). Throughout this thesis, “fixed-parameter tractable” always means “strongly uniformly fixed-parameter tractable”, unless stated otherwise.

Let Π, Π′ be parameterized problems. A parameterized reduction from Π to Π′ is an algorithm that transforms a pair (x,k) ∈ Σ^* × N into a pair (x’,g(k)) ∈ Σ^* × N in time f(k)|x|^c, for arbitrary functions f,g and a constant c independent of k, such that (x,k) ∈ Π if and only if (x’,g(k)) ∈ Π′; then Π is said to be reducible to Π′.

Let C be a Boolean decision circuit, with small gates “not”, “and”, “or” of fan-in at most two and large gates “and”, “or” of unbounded fan-in. The depth (width) of C is the maximum number of gates (large gates) on an input-output path in C. The weight of a Boolean vector is the number of 1’s in it. For a family \mathcal{F} of decision circuits define the parameterized problems

\[ Π_\mathcal{F} \]

Input: A circuit C ∈ \mathcal{F} and an integer k ≥ 0.
Parameter: k.
Question: Does C accept an input vector of weight k?

and
1.3. NOTIONS AND NOTATIONS

\[ \log^{-1}\text{-SAT}(F) \]

**Input:** A circuit \( \gamma \in F \) of size \( m \) with \( n \) input variables.

**Parameter:** \( [n / \log m] \).

**Question:** Is \( \gamma \) satisfiable?

For \( t \geq 1 \) and \( d \geq 1 \), let \( C(t,d) \) denote the family of decision circuits of weight at most \( t \) and depth at most \( d \), and let \( C \) denote the family of all circuits. For \( t \geq 1 \), let \( W[t] \) denote the class of parameterized problems reducible to \( \Pi_{C(t,d)} \), and let \( M[t] \) denote the class of parameterized problems reducible to \( \log^{-1}\text{-SAT}(C(t,d)) \). Let \( W[P] \) denote the class of parameterized problems reducible to \( \Pi_C \). A parameterized problem \( \Pi \) is called \( W[t] \)-hard if there is a parameterized reduction from \( \Pi_{C(t,d)} \) to \( \Pi \), and \( \Pi \) is \( W[t] \)-complete if \( \Pi \) is \( W[t] \)-hard and belongs to \( W[t] \).

For parameterized problems \( \Pi \) and \( \Pi' \), a polynomial parameter transformation is a polynomial-time computable function \( f : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N} \), that for all pairs \( (x,k) \in \Sigma^* \times \mathbb{N} \) satisfies that \( (x,k) \in \Pi \) if and only if \( (x',k') = f(x,k) \in \Pi' \), and \( k' = k^{O(1)} \).

For further notions and background on parameterized complexity we refer to Flum and Grohe [104].

**Algorithms**

An algorithm for a problem \( \Pi \) is an algorithm that correctly determines for every string \( s \) whether \( s \in \Pi \). The running time of the algorithm is measured in the number of steps the algorithm performs. Throughout, we assume a single processor, random-access machine as the underlying machine model, as it is for instance described by Mehlhorn and Sanders [194, Section 2.2]. In the random-access machine any simple operation (arithmetic, if-statements, memory-access etc.) takes unit length of time, and the word size is sufficiently large to hold numbers that are singly exponential in the size of the input.

A \( c \)-approximation algorithm for a minimization (maximization) problem is an algorithm that for a given instance finds a feasible solution \( F \) such that the value of the of the cost function \( c(F) \) is at most \( c \) times (at least \( 1/c \) times) the optimal value. An optimization problem that has a \( c \)-approximation algorithm is called \( c \)-approximable. A polynomial-time approximation scheme (PTAS) for a minimization problem is an algorithm that for any fixed \( \varepsilon > 0 \) computes a polynomial-time \((1+\varepsilon)\)-approximation algorithm for the problem.

Let \( \Pi \) be a fixed-parameter tractable problem. A fixed-parameter algorithm, or FPT-algorithm, for \( \Pi \) is an algorithm for \( \Pi \) deciding instances \( (x,k) \) in time \( f(k) \cdot n^{O(1)} \) for some computable function \( f \). A subexponential fixed-parameter algorithm, or subexponential FPT-algorithm, for \( \Pi \), is a fixed-parameter algorithm for \( \Pi \) with \( f(k) = 2^{o^*(k)} \), where \( f \in o^*(g) \) if there exists an \( n_0 \in \mathbb{N} \) and a computable, nondecreasing and unbounded function \( i : \mathbb{N} \to \mathbb{N} \), such that \( f(n) \leq \frac{s(n)}{\log n} \) for all \( n \geq n_0 \).

For parameterized problems \( \Pi, \Pi' \), a bikernelization from \( \Pi \) to \( \Pi' \) is an algorithm that, given a pair \( (x,k) \in \Sigma^* \times \mathbb{N} \), outputs in time polynomial in \( |x| + k \) a pair \( (x',k') \in \Sigma^* \times \mathbb{N} \) such that \( (x,k) \in \Pi \) if and only if \( (x',k') \in \Pi' \) and \( |x'|, k' \leq g(k) \), where \( g \) is some computable function. We call \( (x',k') \) a bikernel from \( \Pi \) to \( \Pi' \). The function \( g \) is the size of the bikernel, and if \( g(k) = k^{O(1)} \) or \( g(k) = O(k) \) then \( (\Pi, \Pi') \)
is said to admit a polynomial bikernel or linear bikernel, respectively. A kernelization (kernel for) II is a bikernelization (bikernel) from II to itself.

Graphs

For an integer \( r \geq 2 \), an \( r \)-uniform hypergraph is a pair \( G = (V, E) \), where \( V \) is its set of vertices and \( E \) is a collection of subsets of size \( r \) of \( V \), called edges. We also write \( V(G) \) and \( E(G) \) for the sets of vertices and edges of \( G \), respectively. A graph is a 2-uniform hypergraph.

Let \( G \) be a graph. A graph \( G' \) is a subgraph of \( G \) if \( V(G') \subseteq V(G) \) and \( E(G') \subseteq E(G) \). A subgraph \( G' \) of \( G \) is induced if \( E(G') = \{ \{u, v\} \mid u, v \in V(G') \} \); in this case, \( G' \) is called the subgraph induced by \( V(G') \) and denoted by \( G[V(G')] \).

A path of length \( k \) in \( G \) is a sequence \( P = (v_0, v_1, \ldots, v_k) \) of distinct vertices \( v_i \in V(G) \) such that \( \{v_{i-1}, v_i\} \in E(G) \) for \( i = 1, \ldots, k \), and \( v_1, \ldots, v_{k-1} \) are the inner vertices of \( P \). For vertex sets \( V', V'' \subseteq V(G) \), a path from \( V' \) to \( V'' \) is a path \( (v_0, v_1, \ldots, v_k) \) with \( v_0 \in V' \) and \( v_k \in V'' \); if \( V' = \{v_0\} \) (or \( V'' = \{v_k\} \)) write “from \( v_0 \)” (or “to \( v_k \)”). A cycle of length \( k \) in \( G \) is a sequence \( C_k = (v_0, v_1, \ldots, v_k) \) of at least three distinct vertices \( v_i \in V(G) \) such that \( \{v_{i-1}, v_i\} \in E(G) \) for \( i = 1, \ldots, k \) and \( \{v_k, v_0\} \in E(G) \). A cycle of length \( k \) is denoted by \( C_k \), and a cycle of length equal to (at most, at least) \( k \) is called a \( d \)-cycle ((\( \leq d \))-cycle, (\( \geq d \))-cycle). A Hamilton cycle in \( G \) is a \( |V(G)| \)-cycle. A chord of a cycle \( C \) is an edge between two non-consecutive vertices of \( C \). The graph \( G \) is connected if there is a path between any pair of its vertices. For vertices \( u, v \) of \( G \), their distance \( \text{dist}(u, v) \) is the length of the shortest path connecting \( u \) and \( v \) in \( G \), or infinity if no such path exists. For subgraphs \( G_1,G_2 \) of \( G \), the distance \( \text{dist}(G_1,G_2) \) between \( G_1 \) and \( G_2 \) is the minimum distance \( \text{dist}(v_1,v_2) \) over all vertex pairs \( (v_1,v_2) \in V(G_1) \times V(G_2) \).

For all \( d \geq 1 \) and all \( v \in V(G) \), let \( N^d(v) \) be the set of vertices at distance at least one and at most \( d \) from \( v \), and let \( N^d[v] = N^d(v) \cup \{v\} \). The open neighborhood of a vertex \( v \in V \) is the set \( N(v) = N^1(v) \) and the closed neighborhood of \( v \) is the set \( N[v] = N(v) \cup \{v\} \). A vertex \( v \) is universal for \( G \) if \( N[v] = V \). The degree of \( v \) is \( d(v) = |N(v)| \). A vertex of degree equal to (at most, at least) \( d \) is called a \( d \)-vertex ((\( \leq d \))-vertex, (\( \geq d \))-vertex). Call \( G \) twinless if it contains no pair of vertices having the same closed neighborhood, and call \( G \) \( r \)-regular if \( d(v) = r \) for all \( v \in V(G) \). For subsets \( V' \subseteq V \) define \( N^d(V') = \bigcup_{v \in V'} N^d(v) \), \( N^d[V'] = \bigcup_{v \in V'} N^d[v] \), \( N(V') = \bigcup_{v \in V'} N(v) \setminus V' \), and \( N[V'] = \bigcup_{v \in V'} N[v] \). A matching in \( G \) is a set of edges no two of which have an endpoint in common, and an induced matching in \( G \) is an edge set \( M \subseteq E(G) \) such that the graph induced by the vertices of \( M \) is a matching in \( G \). A tree is a connected graph without cycles, whose vertices of degree one are called leaves and whose non-leaf vertices are called internal vertices. A tree \( T \) is spanning for \( G \) if \( V(T) = V(G) \). For \( G \) connected an edge \( e \in E(G) \) is a bridge if \( G - e \) is disconnected. A subdivision of \( G \) is a graph that can be obtained from \( G \) by a sequence of edge subdivisions. The bandwidth \( \text{bw}(G) \) of \( G \) is the minimum, over all permutations of the vertex set of \( G \), of the maximum distance in the order given by a permutation of adjacent vertices of \( G \). The topological bandwidth \( \text{tbw}(G) \) of \( G \) is the minimum, taken over all subdivisions \( G' \) of \( G \), of the bandwidth of \( G' \).

An independent set \( G \) is a subset \( V' \subseteq V(G) \) such that no two vertices of \( V' \) are adjacent in \( G \), a clique of \( G \) is a subgraph of \( G \) in which every two vertices
are adjacent, and a vertex cover of $G$ is a subset $V' \subseteq V(G)$ such that every edge of $G$ is incident to some vertex of $V'$. For integer $r \geq 1$, a distance-$r$ dominating set of $G$ is a subset $V'' \subseteq V(G)$ such that every vertex in $V(G) \setminus V''$ has distance at most $r$ to some vertex of $V''$. A dominating set of $G$ is a distance-1 dominating set of $G$, a connected dominating set of $G$ is a dominating set of $G$ that induces a connected subgraph of $G$, and a dominating clique of $G$ is a dominating set of $G$ that induces a clique. An acyclic vertex set of $G$ is a subset $V' \subseteq V(G)$ such that the graph induced by $V'$ does not contain any cycles. A feedback vertex set of $G$ is the complement of an acyclic vertex set of $G$. A vertex set $V' \subseteq V(G)$ with a property $\pi$ is maximal (minimal) with respect to $\pi$ if no proper superset (subset) of $V'$ has property $\pi$, and is maximum (minimum) if $V'$ has maximum (minimum) size among all subsets with property $\pi$; similarly for edge sets. The size of a maximum independent set (minimum dominating set, minimum connected dominating set) of $G$ is the independence number $\alpha(G)$ (domination number $\gamma(G)$, connected domination number $\gamma_c(G)$) of $G$.

We sometimes color the vertices or edges of $G$. A $c$-vertex-coloring ($c$-edge-coloring) of $G$ is a function from $V(G)$ (from $E(G)$) to $\{1, \ldots, c\}$. A $c$-vertex coloring ($c$-edge-coloring) of $G$ is proper if any two adjacent vertices (any two incident edges) receive distinct colors; and a proper vertex-coloring (proper edge-coloring) of $G$ is a proper $c$-vertex-coloring ($c$-edge-coloring) of $G$ for some $c$. A proper edge-coloring is strong if no edge is adjacent to two edges of the same color.

A graph $G$ is $d$-degenerate if every induced subgraph of $G$ has a vertex of degree at most $d$. A graph $G$ is $k$-connected, for some integer $k \geq 2$, if $G$ contains no set of $k - 1$ vertices whose removal disconnects the graph. A graph $G$ is bipartite if $V(G)$ can be partitioned into two independent sets. A graph $G$ is complete if it has one edge between each pair of vertices. We denote by $K_h$ the complete graph on $h$ vertices, and by $K_{h,h'}$ the complete bipartite graph with independent sets of sizes $h$ and $h'$. A subgraph $G'$ of a graph $G$ is a component of $G$ if $G'$ is connected and $G[V(G') \cup \{v\}]$ is disconnected for every vertex $v \in V(G) \setminus V(G')$; and $G'$ is a block of $G$ if either $G$ is a complete graph on at most vertices, or $G'$ is 2-connected and $G[V(G') \cup \{v\}]$ is not 2-connected for every vertex $v \in V(G) \setminus V(G')$. The half-square of a bipartite graph $G = (V_1 \uplus V_2, E)$, where $V_1, V_2$ are independent sets, is the graph obtained from $G$ by connecting every two vertices in $V_1$ with distance two in $G$ by an edge. For integers $k, n \geq 1$, the generalized Petersen graph $P(n,k)$ is the graph with vertex set $V(P(n,k)) = \{u_0, u_1, \ldots, u_{n-1}, v_0, v_1, \ldots, v_{n-1}\}$ and edge set

$$E(P(n,k)) = \{\{u_i, u_{i+1 \mod n-1}\}, \{u_i, v_i\} \mid i = 0, \ldots, n-1\} \cup \{\{v_i, v_{i+k \mod n-1}\} \mid i = 0, \ldots, n-1, k = 1, \ldots, \lfloor n/2 \rfloor\}.$$

### Treewidth, Minors and Contractions

A tree decomposition of a graph $G$ is a pair $(T, B)$ where $T = (V_T, E_T)$ is a tree and $B = \{B_i \mid i \in V_T\}$ is a collection of subsets of $V(G)$, called bags, such that

- $\bigcup_{i \in V_T} B_i = V(G)$,
- for each edge $\{x, y\} \in E(G)$, $\{x, y\} \subseteq B_i$ for some $i \in V_T$;
- for each $x \in V(G)$ the set $\{i \mid x \in B_i\}$ induces a connected subtree of $T$. 


The width of the tree decomposition is \( \max_{i \in V_T} |B_i| - 1 \). The treewidth of a graph \( G \) is the minimum width \( \text{tw}(G) \) over all tree decompositions of \( G \). A tree decomposition \((T, B)\) can be converted in linear time [174] into a nice tree decomposition of the same width: here, the tree \( T \) is rooted and binary, and its nodes are of four types:

- **Leaf nodes** \( h \) are leaves of \( T \) and have \( |B_h| = 1 \).
- **Introduce nodes** \( h \) have one child \( i \) with \( B_h = B_i \cup \{ v \} \) for some \( v \in V(G) \).
- **Forget nodes** \( h \) have one child \( i \) with \( B_h = B_i \setminus \{ v \} \) for some \( v \in V(G) \).
- **Join nodes** \( h \) have two children \( i, j \) with \( B_h = B_i \cup B_j \).

If, in the above definitions, we restrict the tree \( T \) to be a path then we define the notions of path decomposition and pathwidth \( \text{pw}(G) \) of \( G \).

For a vertex set \( V' \subseteq V \), contracting \( V' \) into \( v' \) means to remove all vertices of \( V' \) from \( G \) and adding a new vertex \( v' \) \( \notin V \) that is adjacent to all vertices of \( V \setminus V' \). For a vertex set \( V' \subseteq V \), contracting \( V' \) into \( v' \) means to remove all vertices of \( V' \) from \( G \) and adding a new vertex \( v' \) \( \notin V \) that is adjacent to all vertices of \( V \setminus V' \) but not adjacent to \( v' \). If \( G \) has a vertex of degree \( 1 \) then we define the contraction of \( G \) by contracting the set \( \{ v \} \).

A graph \( H \) is a **minor** of a graph \( G \) if \( H \) is the contraction of some subgraph of \( G \) and denoted by \( H \subseteq_m G \). If a subdivision of a graph \( H \) is the subgraph of another graph \( G \) then \( H \) is a topological minor of \( G \). A graph \( G \) is **H-minor free (H-topological minor free)** for a graph \( H \) if \( G \) does not contain \( H \) as a minor (topological minor). We also say that a graph class \( G_H \) is **H-minor-free** (or, excludes \( H \) as a minor) when all its members are \( H \)-minor-free.

### Graphs on Surfaces

Given a graph \( G \), its *Euler genus* is the minimum \( g \) such that \( G \) is embeddable into a surface of Euler genus \( g \). An graph is **planar** if it has an embedding in the sphere \( S_0 = \{(x, y, z) \mid x^2 + y^2 + z^2 = 1\} \), and a **plane graph** is an \( S_0 \)-embedded graph.

Let \( G \) be a plane graph. We usually do not distinguish between a vertex of the graph and the point of \( S_0 \) used in the embedding to represent the vertex, or between an edge and the open line segment representing it. We denote the set of faces in the embedding of \( G \) by \( F(G) \). (Every face is an open set.) An edge \( e \) (vertex \( v \)) is **incident** to a face \( f \) if \( e \subseteq \overline{f} \) (if \( v \subseteq \overline{f} \)) where \( \overline{f} \) denotes the closure of \( f \). For a face \( f \) of \( G \), its degree \( \text{d}(f) \) is the number of edges incident to it, with the exception that cut-edges are counted twice. The notions of \( d \)-face, \((\leq d)\)-face and \((\geq d)\)-face are defined analogously as for vertices. A map of \( G \) is a pair \( \mathcal{M} = (G, \varphi) \), where \( \varphi : F(G) \to \{0, 1\} \) is a two-coloring of the faces of \( G \). A map graph \( \mathcal{G}_M \) of a map \( \mathcal{M} \) is the graph on the vertex set \( \{ f \in F(G) \mid \varphi(f) = 1 \} \) in which two vertices \( f_1, f_2 \) are adjacent in \( \mathcal{G}_M \) precisely if \( f_1 \cap f_2 \) contains at least one vertex of \( G \). It was shown [57] that every map graph \( \mathcal{G}_M \) is the half-square of some planar bipartite graph \( H \); we call \( H \) the **witness** of \( \mathcal{G}_M \).

Let \( r \geq 2 \) be an integer. The \((r \times r)\)-grid is the Cartesian product of two paths of lengths \( r - 1 \). A vertex of a grid is a **corner** if it has degree 2. Thus each \((r \times r)\)-grid has 4 corners. A vertex of a \((r \times r)\)-grid is called **internal** if it has degree 4, otherwise it is called **external**. Let \( F_r \) be the graph obtained from a plane \((r \times r)\)-grid...
A directed multigraph is a pair $D = (V, A)$, where $V$ is its set of vertices and $A$ is a multiset of ordered pairs from $V \times V$, called arcs. We also write $V(D)$ and $A(D)$ for the set of vertices and multiset of arcs of $D$, respectively. A directed graph is a directed multigraph $D$ for which $A(D)$ is a set.

Let $D$ be a directed graph. A directed graph $D'$ is a subgraph of $D$ if $V(D') \subseteq V(D)$ and $A(D') \subseteq A(D)$. The subgraph $D'$ is called an induced subgraph of $D$ if $A(D') = \{(u, v) \in A(D) \mid u, v \in V(D')\}$, in this case, $D'$ is also called the subgraph induced by $V(D')$ and denoted by $D[V(D')]$. An arc $(u, v) \in A(D)$ is a loop if $u = v$. The in-neighborhood (out-neighborhood) of a vertex $v \in V$ is the set $N^-(v) = \{u \in V \mid (u, v) \in A\}$ (the set $N^+(v) = \{u \in V \mid (v, u) \in A\}$). The in-degree resp. out-degree of $v$ is $d^-(v) = |N^-(v)|$ resp. $d^+(v) = |N^+(v)|$. The degree of $v$ is $d^-(v) + d^+(v)$; we stipulate that loops make unit contribution to the degree of a vertex. Distinct vertices $u, v$ are called adjacent if $\{(u, v) \mid (v, u)\} \cap A(D)$ is non-empty.

A path of length $k$ in $D$ is a sequence $P = (v_0, v_1, \ldots, v_k)$ of distinct vertices $v_i \in V(D)$ such that $(v_{i-1}, v_i) \in A(D)$ for $1 \leq i \leq k$. A Hamilton path in $D$ is a path in $D$ of length $|V(D)| - 1$. For vertex sets $V', V'' \subseteq V(D)$, a path from $V'$ to $V''$ is a path $(v_0, v_1, \ldots, v_k)$ with $v_0 \in V'$ and $v_k \in V''$; if $V' = \{v_0\}$ (or $V'' = \{v_k\}$) write "from $v_0$" (or "to $v_k$"). A cycle in $D$ is a path from $v_0$ to $v_0$ of length at least one, for some vertex $v_0 \in V$, and a cycle of length $k$ is denoted by $C_k$. For vertices $u, v$ of $D$, the distance from $u$ to $v$ is the length of a shortest path from $u$ to $v$, or infinity if no such path exists. A directed graph $D$ is strong if there is a path from any vertex $u$...
of $D$ to any other vertex $v$ of $D$. A vertex set $I \subseteq V(D)$ is an independent set of $D$ if there are no arcs between any vertices of $D$, and an acyclic vertex set of $D$ if there are no cycles in $D[V(D) \setminus I]$. The complement of an acyclic vertex set of $D$ is a feedback vertex set of $D$.

A tournament is a directed graph with exactly one arc between each pair of distinct vertices. A directed graph $D$ is bipartite if there is a bipartition of $V(D)$ into independent sets. A bipartite tournament is a directed graph $D = (V, A)$ such that there is a bipartition of $V$ into independent sets $V_1$ and $V_2$ and for any pair $(v_1, v_2) \in V_1 \times V_2$ either $(v_1, v_2) \in A$ or $(v_2, v_1) \in A$.

We define the operation of contracting an arc $(u, v)$ as follows: add a new vertex $u'$, and for each arc $(w, v)$ or $(w, u)$ add the arc $(w, u')$ and for an arc $(v, w)$ or $(u, w)$ add the arc $(u', w)$, then remove all arcs incident to $u$ or $v$ and the vertices $u$ and $v$. A directed graph without cycles is called a tree, and a tree is rooted if it has a unique vertex of in-degree zero, called its root.

For graphs and directed graphs, all terminology and notation not defined here follows that of Diestel [75].

1.4 List of Problems

We give a list of problems addressed in this thesis. For problems not defined here we refer to the compendia by Downey and Fellows [82] and Cesati [51].

3-Coloring Given a graph $G$, decide if $G$ has a proper 3-vertex-coloring.

3-Satisfiability Given a Boolean formula $F$ in conjunctive normal form with at most 3 literals per clause, decide if there is a truth assignment satisfying $F$.

Acyclic Subdigraph Given a directed multigraph $D$, find a maximum subset $A' \subseteq A(D)$ of arcs that form an acyclic subgraph of $D$.

Bandwidth Given a graph $G$, find a permutation $\sigma$ of $V(G) = \{1, \ldots, n\}$ minimizing the maximum $|\sigma(u) - \sigma(v)|$ over all edges $(u, v) \in E(G)$.

Betweenness Given a set $V$ of variables and a set of constraints $C$ = “$u$ between $v$ and $w$” for distinct $u, v, w \in V$, find a permutation $\sigma$ of $V$ maximizing the number of constraints $C$ for which $\sigma(v) < \sigma(u) < \sigma(w)$ or $\sigma(w) < \sigma(u) < \sigma(v)$.

Circular Ordering Given a set $V$ of variables and a set $C$ of cyclically ordered triples of $V$, find a cyclic order of $V$ that extends a maximum number of triples from $C$.

Clique Given a graph $G$, find a maximum clique of $G$.

Connected Dominating Set (Vertex Cover, Feedback Vertex Set) Given a graph $G$, find a minimum vertex set $S \subseteq V(G)$ such that $S$ is a dominating set (vertex cover, feedback vertex set) of $G$ and $S$ induces a connected graph.

Cycle Packing Given a graph $G$, find a maximum number of edge-disjoint cycles in $G$. 
1.4. LIST OF PROBLEMS

DOMINATING CLIQUE Given a graph $G$, find a minimum vertex set $D \subseteq V(G)$ such that $D$ is a dominating set of $G$ and $D$ induces a clique.

DOMINATING SET (r-DOMINATING SET) Given a graph $G$, find a minimum vertex set $D \subseteq V(G)$ such that any vertex in $V(G) \setminus D$ has distance at most one (at most $r$) to some vertex in $D$.

EDGE DOMINATING SET Given a graph $G$, find a minimum edge set $E \subseteq E(G)$ such that any edge from $E(G) \setminus E$ is incident to some edge of $E$.

EDGE MULTICUT Given a graph $G$ and a list $\{s_1, t_1\}, \ldots, \{s_l, t_l\}$ of vertex pairs, find a minimum set of edges whose removal from $G$ disconnects $s_i$ from $t_i$ for all $i = 1, \ldots, l$.

EFFICIENT EDGE DOMINATING SET Given a graph $G$, find a minimum edge set $E \subseteq E(G)$ such that no two edges of $E$ are incident in $G$ and any edge from $E(G) \setminus E$ is incident to exactly one edge of $E$.

EQUITABLE COLORING Given a graph $G$, find a proper vertex-coloring of $G$ using a minimum number of colors such that the number of vertices having any two distinct colors differs by at most one.

FEEDBACK ARC SET Given a directed graph $G$, find a minimum arc set $F \subseteq E(G)$ such that the directed graph $(V(G), E(G) \setminus F)$ contains no cycles.

FEEDBACK VERTEX SET Given a directed graph (graph) $G$, find a minimum vertex set $F \subseteq V(G)$ such that the directed graph (graph) induced by $V(G) \setminus F$ contains no cycles.

FULL-DEGREE SPANNING TREE Given a graph $G$, find a spanning tree of $G$ maximizing the number of vertices whose degree in the spanning tree is the same as their degree in $G$.

GENERAL FACTOR Given a graph $G$ and a mapping $\alpha$ that assigns to each vertex $v \in V(G)$ a subset $\alpha(v) \subseteq \{0, \ldots, d(v)\}$, determine if there is a subset $F \subseteq E(G)$ such that for each $v \in V(G)$ the number of edges in $F$ incident with $v$ is an element of $\alpha(v)$?

GENUS Given a graph $G$, find the minimum $g$ such that $G$ embeds into a surface of Euler genus $g$.

GRAPH MINOR Given graphs $G$ and $H$, decide whether $H$ is a minor of $G$.

INDEPENDENT DOMINATING SET Given a graph $G$, find a minimum vertex set $D \subseteq V(G)$ such that $D$ is both an independent set and a dominating set of $G$.

INDEPENDENT SET Given a graph $G$, find a maximum independent set of $G$.

INDUCED MATCHING Given a graph $G$, find a maximum induced matching of $G$.

LINEAR ORDERING Given a directed graph $D = (V, A)$ with positive integral weights on arcs, find an acyclic subgraph of $D$ with maximum weight.
1. INTRODUCTION

**LIST COLORING** Given a graph $G$ and for each vertex $v \in V(G)$ a list $L(v)$ of colors, find a proper vertex-coloring $\varphi$ of $G$ such that $\varphi(v) \in L(v)$ for all $v \in V(G)$.

**MAX-CUT** Given a graph $G$, find a bipartition of $\{V_1, V_2\}$ of $V(G)$ maximizing the number of edges $\{v_1, v_2\} \in E(G)$ with $v_1 \in V_1, v_2 \in V_2$.

**MAX LEAF SPANNING TREE** Given a graph $G$, find a spanning tree of $G$ with a maximum number of leaves.

**MAX-LIN 2** Given a system $S$ of linear equations with positive integral weights in variables over $\mathbb{F}_2$, find an assignment of values to the variables maximizing the total weight of satisfied equations in $S$.

**MAX-SAT (MAX-$r$-SAT)** Given a Boolean formula $F$ in conjunctive normal form (in which each clause has at most $r$ literals), find a truth assignment satisfying a maximum number of clauses in $F$.

**MINIMUM MAXIMAL MATCHING** Given a graph $G$, find a maximal matching in $G$ of minimum size.

**MINIMUM LEAF OUT-BRANCHING** Given a directed graph $D$, find subgraph $T$ of $D$ that is a tree, has $V(T) = V(D)$ and the minimum possible number of leaves.

**MINIMUM MAXIMUM OUTDEGREE** Given a graph $G$, find an orientation $\Lambda : \{u, w\} \in E(G) \mapsto \{(u, v), (v, u)\}$ of $G$ that minimizes the maximum out-degree over all vertices of $G$.

**ODD CYCLE TRANSVERSAL** Given a graph $G$, find a vertex set $T \subseteq V(G)$ of minimum size such that the graph $G[V(G) \setminus T]$ is bipartite.

**$(k)$-PATH** Given a graph $G$, decide if $G$ has a path of length $k$.

**PARTIAL VERTEX COVER** Given a graph $G$ and an integer $t \geq 0$, find a minimum vertex set $C \subseteq V(G)$ such that $t$ edges of $G$ are incident to vertices in $C$.

**PRE-COLORING EXTENSION** Given a graph $G$, a subset $W \subseteq V$ and a proper vertex-coloring $\varphi_W$ of $W$, determine the minimum number of colors in a proper vertex-coloring of $V(G)$ that extends $\varphi_W$.

**QUARTET (TRIPLET) INCONSISTENCY** Given a set $L$ of labels and a set $T$ of unrooted (rooted) binary trees distinctly leaf-labeled by 4-subsets (3-subsets) of $L$, find a minimum set $T' \subseteq T$ such that some unrooted (rooted) binary tree distinctly leaf-labeled by $L$ contains all trees from $T \setminus T'$ as homeomorphic subtree.

**SET COVER** Given a set $S$, a family $\mathcal{F}$ of subsets of $S$, find a minimum subfamily $\mathcal{F}' \subseteq \mathcal{F}$ such that every element in $S$ belongs to some set in $\mathcal{F}'$.

**SET SPLITTING** Given a set $S$ and a family $\mathcal{F}$ of subsets of $S$, find a maximum subfamily $\mathcal{F}' \subseteq \mathcal{F}$ and a bipartition $\{S_1, S_2\}$ of $S$ such that all sets in $\mathcal{F}$ have non-empty intersection with both $S_1$ and $S_2$. 
(k)-Step Halting Given a nondeterministic single-tape Turing machine $M$ with its transition table, a string $x$ and an integer $k \geq 0$, decide whether $M$ accepts $x$ after at most $k$ steps.

Subset-Sum Given positive integers $a_1, \ldots, a_n$ and a value $s$, determine whether there is a subset of the $a_i$ that sums up to $s$.

$q$-Threshold Dominating Set Given a graph $G$, find a minimum subset $V' \subseteq V(G)$ such that for any vertex in $v \in V(G) \setminus V'$ at least $q$ vertices of the closed neighborhood $N[v]$ belong to $V'$.

Topological Bandwidth Given a graph $G$, minimize the maximum of $|a(u) - a(v)|$ over all permutations $a$ of the vertex set $V(G') = \{1, \ldots, n\}$ of subdivisions $G'$ of $G$ and edges $\{u, v\} \in E(G')$.

Traveling Salesman Problem Given a set $S = \{1, \ldots, n\}$ of cities and their pairwise distances $d(i, j)$, find a permutation $a$ of the cities in $S$ minimizing the sum $\sum_{i=1}^{n} d(a^{-1}(i), a^{-1}(i + 1 \mod n))$.

Unsatisfiability Given a Boolean formula in conjunctive normal form, decide if no truth assignment satisfies $F$.

Vertex Cover Given a graph $G$, find a minimum vertex cover of $G$.

Vertex Disjoint Cycles (Triangles) Given a graph $G$, find a maximum number of vertex-disjoint cycles (3-cycles) in $G$. 

Parameterized complexity explores how one parameter affects the complexity of a different parameterized or unparameterized problem. A well-developed example is the investigation of how the parameter treewidth influences the complexity of various graph problems. The reason why such investigations are of general interest is that real-world input distributions for computational problems often inherit structure from the natural computational processes that produce the problem instances, not necessarily in obvious, or well-understood ways. The \textit{max leaf number} $ml(G)$ of a connected graph $G$ is the maximum number of leaves in a spanning tree for $G$. Exploring questions analogous to the well-studied case of treewidth, we can ask: how hard is it to solve 3-\textsc{Coloring}, \textsc{Hamilton Path}, \textsc{Dominating Set}, \textsc{Bandwidth} or other problems, for graphs of bounded max leaf number? What optimization problems are $W[1]$-hard under this parameterization?

The analysis of the complexity of problems, for graphs of bounded treewidth, is well-developed and supports many systematic approaches that have developed over a number of years [25, 33, 37, 61, 82, 205]. For example, determining whether a graph on $n$ vertices has a proper 3-vertex-coloring can be solved in time $O(n)$ for graphs of treewidth at most $k$. In the terminology of parameterized complexity, 3-\textsc{Coloring} is fixed-parameter tractable for the parameter treewidth. In this small example, the asymptotic notation conceals serious costs associated to the treewidth bound $k$, from two sources:

- The complexity of computing a tree-decomposition of width $k$ is $O(2^{3k^3}n)$ for an $n$-vertex graph.

- Once the tree-decomposition is obtained, one would then solve the problem by dynamic programming, in time $O(3^k n)$.

Suppose that we wish to solve 3-\textsc{Coloring} for graphs having a different structural restriction—how should this be done? Here we consider the parameter of bounded
max leaf number for connected graphs $G$. We choose this parameter mainly to illustrate the key issues, and because enough is known of the associated polynomial-time extremal structure theory to provide a good example of the general approach. (Here, by polynomial-time extremal structure theory we mean extremal combinatorial bounds together with algorithms “certifying” these bounds in polynomial time.) We are not aware of any strong direct applications of bounded max leaf number for natural input distributions.

In this chapter we address the question of efficient algorithms for solving various problems on graphs of bounded max leaf number, from both the “better $f(k)$” and “better polynomial-time kernelization” points of view, a dual perspective that is now standard in parameterized algorithmics [205]. We have two main objectives:

First, we describe efficient FPT-algorithms for 3-Coloring and other graphs problems, for input $G$ parameterized by a bound $ml(G) \leq k$. As a motivating example, notice that one way to approach the problem of determining whether a graph has a proper 3-vertex-coloring, parameterized by max leaf number, is to note that graphs of bounded max leaf number exclude a tree minor and therefore have bounded pathwidth. Thus, algorithms for graphs of bounded treewidth as sketched in Subsection 1.1.4 can be used. This classifies 3-Coloring, parameterized by max leaf number, as non-uniform FPT, but this is not an efficient algorithm.

Second, we describe the algorithms in a way that is generally systematic, and that “fits" the study of how parameterized structure affects computational complexity in what we term the “ecology” of parameterized complexity. One can view this issue as a kind of generalized bidimensionality theory in the sense of Demaine and Hajiaghayi [70–72].

### 2.1 A Complexity Matrix of Parameters

An example to illustrate the main idea of this chapter is afforded by the problem of Type Checking of programs written in high-level logic-based programming languages such as ML. This problem has been shown to be complete for EXP [157, 171], and thus is highly intractable from the classical point of view. Nevertheless, the ML compilers (that include type-checking subroutines) work efficiently. The explanation is that human-composed programs typically have a maximum type-declaration nesting depth of $k \leq 5$. The FPT type-checking subroutine that runs in time $O(2^k n)$ is thus entirely adequate in practice. One can reasonably speculate that naturally occurring programs have small nesting depth because the programs would otherwise risk becoming incomprehensible to the programmer creating them.

What this example points to is that often the “inputs" to one computational problem of interest to real-world algorithmics are not at all arbitrary, but rather are produced by other natural computational processes (e.g., the thinking processes and abilities of the programmer) that are themselves subject to computational complexity constraints. In this way, the natural input distributions encountered by abstractly defined computational problems often have inherited structural regularities and restrictions (relevant parameters, in the sense of parameterized complexity) due to the natural complexity constraints on the generative processes. This seems reasonable, although what the resulting relevant “parameters" are may not be obvious. This
2.1. A COMPLEXITY MATRIX OF PARAMETERS

<table>
<thead>
<tr>
<th></th>
<th>TW</th>
<th>BW</th>
<th>VC</th>
<th>DS</th>
<th>G</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>TW</td>
<td>FPT</td>
<td>W[1]-hard</td>
<td>FPT</td>
<td>FPT</td>
<td>?</td>
<td>FPT</td>
</tr>
<tr>
<td>BW</td>
<td>FPT</td>
<td>W[1]-hard</td>
<td>FPT</td>
<td>FPT</td>
<td>?</td>
<td>FPT</td>
</tr>
<tr>
<td>VC</td>
<td>FPT</td>
<td>FPT</td>
<td>FPT</td>
<td>FPT</td>
<td>?</td>
<td>FPT</td>
</tr>
<tr>
<td>ML</td>
<td>FPT</td>
<td>FPT</td>
<td>FPT</td>
<td>FPT</td>
<td>FPT</td>
<td>FPT</td>
</tr>
</tbody>
</table>

Table 2.1: The complexity ecology of parameters, an entry describing the current state of knowledge about the complexity of the problem where the input graph is assumed to have a structural bound described by the row, and the problem described by the column is to be solved to optimality.

connection is what we refer to as the “ecology of computation”.

Our thesis is that it is useful to know how all the various parameterized structural notions interact with all the other computational objectives one might have. Our main objective is to illustrate how such a quest can be systematically engaged. The familiar paradigm of efficiently solving various problems for graphs of bounded treewidth just represents one row of a matrix of algorithmic questions that arise from the relevant parameterized structure theories.

Table 2.1 illustrates the idea of such a matrix of algorithmic questions. We use here the shorthand: TW is Treewidth, BW is Bandwidth, VC is Vertex Cover, DS is Dominating Set, G is Genus and ML is Max Leaf Spanning Tree. The entry in the 2nd row and 4th column indicates that there is an FPT-algorithm to optimally solve the Dominating Set problem for a graph $G$ of bandwidth at most $k$. The entry in the 4th row and 2nd column indicates that it is unknown whether Bandwidth can be solved optimally by an FPT-algorithm when the parameter is a bound on the domination number of the input. An entry in the table describes the current state of knowledge about the complexity of the problem where the input graph is assumed to have a structural bound described by the row, and the problem described by the column is to be solved to optimality. The table just gives a few examples of the unbounded conceptual matrix that we are concerned with.

Most of the research in algorithms so far that pertains to this matrix is concerned with the first row and the diagonal. For the Max Leaf Spanning Tree row, we investigate how to solve various problems optimally on graphs $G$ having bounded max leaf number, $ml(G) \leq k$, exploiting the structure that bounding this parameter yields. The Vertex Cover row is explored by Fellows et al. [99].

One might ask whether these rows are really interesting, since a graph of bounded max leaf number is severely restricted in its structure. To be fair, however, a graph of bounded treewidth is also severely restricted, in contrast to an arbitrary graph. How to determine whether a graph of bounded max leaf number has a proper 3-vertex-coloring in the “best possible” FPT running time is an easily stated problem for which the answer is not obvious. Another observation that points to the interest in these rows is that there are now known to be many examples of problems that are $W[1]$-hard parameterized by a bound on treewidth, including List Coloring, Pre-Coloring Extension and Equitable Coloring [95]; General Factor [233]; Minimum Maximum Outdegree [234]; and Bandwidth [33], which is NP-complete even for very restricted subclasses of trees [196]. One must there-
fore look “below treewidth” for FPT parameterizations for these problems. Note that while both bounded vertex cover number and bounded max leaf number imply bounded treewidth, neither of these structural bounds implies a bound on the other. List Coloring even remains \(W[1]\)-hard for graphs of bounded vertex cover number \([99]\). Lastly, it seems that for severe parameterizations such as bounded vertex cover number and bounded max leaf number, different FPT techniques are brought forward to importance, such as well-quasi-ordering and bounded variable integer linear programming.

2.2 Attacking a Row: Kernelization

We first develop the polynomial-time extremal structure theory for graphs of bounded max leaf number, using the approach proposed by Estivill-Castro et al. \([90]\) and Prieto-Rodriguez \([211]\). We prove a linear bound on the kernel size for (\(k\))-MAX LEAF SPANNING TREE, based on a collection of polynomial-time reduction rules. We then describe kernelizations for various problems in the MAX LEAF SPANNING TREE row of the ecology matrix by (1) deploying similar reduction rules, and (2) adjusting the kernelization bound. We describe what we are doing from a general point of view that suggests how the approach might be adapted to other problems.

2.2.1 Kernelization for (\(k\))-MAX-LEAF SPANNING TREE

In order to articulate the structure that a bound on the max leaf number imposes, we seek to prove the following kernelization lemma.

**Lemma 2.1.** Let \((G, k)\) be a reduced “yes”-instance of (\(k\))-MAX LEAF SPANNING TREE such that \((G, k + 1)\) is a “no”-instance. Then \(|V(G)| \leq ck\).

Here, reduced means that a certain set of reduction rules does not apply to the instance \((G, k)\), and \(c\) is a small constant that we will clarify below.

Proving such a “Boundary Lemma” involves two crucial strategic choices:

1. A choice of witness structure for the hypothesis that \((G, k)\) is a “yes”-instance.
2. A choice of inductive priorities.

Below, we prove two versions of such a result, to illustrate the methodology. The overall structure of the argument is “by minimum counterexample” according to the priorities established by (2), which generally make reference to (1). Given these choices, our proof proceeds by a series of small steps consisting of structural claims that lead to a detailed structural picture at the “boundary”—and thereby to the bound on the size of \(G\) that is the conclusion of the lemma.

2.2.2 Boundary Lemma I

Consider the following two reduction rules, and apply them exhaustively to the instance \((G, k)\).
Reduction Rule 2.1 (Adjacent Degree-2 Rule). For adjacent 2-vertices $u$ and $v$, contract the edge $\{u, v\}$ if it is a bridge, or delete it otherwise.

Lemma 2.2. Let $G$ be a connected graph and let $k \geq 1$ be an integer. If $G'$ is the graph obtained from $G$ by applying Reduction Rule 2.1 then $G'$ has a spanning tree with $k$ leaves if and only if $G$ has a spanning tree with $k$ leaves.

Proof. If edge $\{u, v\}$ is a bridge then neither $u$ nor $v$ is a leaf in any spanning tree $T$ of $G$, and therefore $T / \{u, v\}$ is a spanning tree of $G'$ with the same number of leaves as $G$. Conversely, any spanning tree $T'$ of $G'$ can be extended to a spanning tree of $G$ with the same number of leaves.

If $\{u, v\}$ is not a bridge then either both $u$ and $v$ are leaves in some spanning tree $T$ of $G$, which is hence also a spanning tree of $G'$ with the same number of leaves; or neither $u$ nor $v$ is a leaf of $T$, in which case replacing $\{u, v\}$ by an appropriate edge between two leaves of $T$ on a path between $u$ and $v$ that is edge-disjoint from $\{u, v\}$ yields a spanning tree of $G'$ with the same number of leaves. Conversely, any spanning tree of $G'$ is also a spanning tree of $G$ with the same number of leaves.

Reduction Rule 2.2 (Degree-1 Rule). For a 1-vertex $u$, let $v$ be the vertex adjacent to $u$ and let $w \neq u$ be a vertex adjacent to $v$. If $v$ is a 2-vertex then remove $u$, otherwise add the edge $\{u, w\}$.

Lemma 2.3. Let $G$ be a connected graph on at least three vertices and let $k \geq 1$ be an integer. If $G'$ is the graph obtained from $G$ by applying Reduction Rule 2.2 then $G'$ has a spanning tree with $k$ leaves if and only if $G$ has a spanning tree with $k$ leaves.

Proof. If $u$ is adjacent in $G$ to a 2-vertex $v$ then in any spanning tree of $G$, $u$ is a leaf and $v$ is not a leaf, whereas in any spanning tree of $G' = G - u$, $v$ is a leaf. Therefore, $G$ has a spanning tree with $k$ leaves if and only if $G'$ has a spanning tree with $k$ leaves.

If $u$ is adjacent in $G$ to a vertex $v$ of degree at least 3, and adjacent in $G'$ to another vertex $w \in N(v) \setminus \{u\}$, then any spanning tree of $G$ is a spanning tree of $G'$ with the same number of leaves. Let $T'$ be a spanning tree of $G'$. If $T'$ does not use the edge $\{u, w\}$ then $T'$ is a spanning tree of $G$ with the same number of leaves. If $T'$ uses $\{u, w\}$ and $\{u, v\}$ then replacing $\{u, w\}$ by $\{v, w\}$ yields a spanning tree of $G$ with the same number of leaves. If $T'$ uses $\{u, w\}$ but not $\{u, v\}$ then replacing $\{u, w\}$ by $\{u, v\}$ yields a spanning tree of $G$ with the same number of leaves.

We call an instance $(G, k)$ reduced if none of the two reduction rules applies. Each application of these rules creates an instance $(G', k')$ for $(k)$-Max Leaf Spanning Tree, where $G'$ is the graph obtained from $G$ and $k' = k$ in this case.

While for the moment the choice of those rules is left unmotivated, the proof of the following Boundary Lemma exploits the absence of exactly those structures in the graph.

Lemma 2.4 (Boundary Lemma I). Let $(G, k)$ be a reduced “yes”-instance of $(k)$-Max Leaf Spanning Tree such that $(G, k + 1)$ a “no”-instance. Then $|V(G)| \leq 7.75k$. 
Proof. The proof is by minimum counterexample. If \((G, k)\) is a “yes”-instance then we can assume that we are given as a witness structure a tree subgraph \(T = (V', E')\) of \(G\) that has \(k\) leaves, and we can also assume that \(G\) is connected. We do not assume that \(T\) is a spanning subgraph. (If \(T\) is not spanning, then it clearly extends to a spanning tree \(T'\) for \(G\) that has at least \(k\) leaves.) A counterexample to the lemma would be a graph \(G\) such that

- \((G, k)\) is a reduced “yes”-instance of \((k)\)-Max Leaf Spanning Tree,
- \((G, k + 1)\) is a “no”-instance of \((k)\)-Max Leaf Spanning Tree, and
- \(|V(G)| > 7.75k\).

Among all such counterexamples, consider one where the witness subgraph tree \(T\) is as small as possible. Let \(O = V(G) \setminus V'\) be the set of vertices not in the witness subtree \(T\), which we will refer to as outsiders. Let \(L\) denote the leaves of \(T\), \(I\) the internal (non-leaf) vertices of \(T\), \(B \subseteq I\) the branch vertices of \(T\) (the non-leaf, internal vertices of \(T\) that have degree at least 3 with respect to \(T\)), and let \(J\) denote the subdivider vertices of \(T\) (the non-branch internal vertices of \(T\) that have degree 2 with respect to \(T\)). See Fig. 2.1 for an example.

![Figure 2.1: The witness tree and various sets of vertices.](image)

We will need to discuss the structure of \(T\) in more detail, so we introduce the following further terminology. A path \(P = (b, j_1, \ldots, j_t, b')\) in \(T\) where \(b, b' \in B\) are branch vertices of \(T\), and the vertices \(j_i\) for \(i = 1, \ldots, t\) are subdivider vertices of \(T\), is termed a topological edge or topo-edge of the tree \(T\). In this situation we will say that \(b\) and \(b'\) are topologically adjacent in \(T\), and in order to be able to refer to the length of \(P\), we say that \(b\) and \(b'\) are joined by a \(t\)-topo-edge in \(T\). We will eventually be interested in structures that arise by considering the subtrees of \(T\) induced by 0-topo-edges and 1-topo-edges of \(T\). (Note that a 0-topo-edge is just an ordinary edge of \(T\).)

**Claim 2.1.** No internal vertex of \(T\) is adjacent in \(G\) to a vertex of \(O\).

**Proof.** Otherwise, we could augment \(T\) to a subgraph tree with \(k + 1\) leaves, contradicting that \((G, k + 1)\) is a “no”-instance. (See Fig. 2.2.)

**Claim 2.2.** Any leaf of \(T\) is adjacent to at most one outsider.

**Proof.** Otherwise we contradict that \((G, k + 1)\) is a “no”-instance.

**Claim 2.3.** The subgraph of \(G\) induced by the outsiders is acyclic.
2.2. ATTACKING A ROW: KERNELIZATION

Figure 2.2: No internal vertex of $T$ is adjacent to an outsider, else $k + 1$ leaves.

Proof. Otherwise, since $G$ is connected, we contradict that $(G, k + 1)$ is a “no”-instance. See Fig. 2.3.

Figure 2.3: $G[O]$ is acyclic, else $G$ has a spanning tree with at least $k + 1$ leaves.

Claim 2.4. The subgraph induced by the outsiders has maximum degree at most 2.

Proof. Otherwise we contradict that $(G, k + 1)$ is a “no”-instance. See Fig. 2.4.

Figure 2.4: $G[O]$ has maximum degree at most 2, else $k + 1$ leaves.

By Claims 2.3 and 2.4, the subgraph $G[O]$ induced by the outsiders consists of a union of paths.

Claim 2.5. No leaf of $G$ is adjacent to an inner vertex of a path of $G[O]$.

Proof. Otherwise we contradict that $(G, k + 1)$ is a “no”-instance.

Claims 2.1–2.5 show that $G[O]$ consists of a disjoint union of paths, and that the interior vertices of these paths have degree 2 in $G$. This motivated us to look for a reduction rule that addresses this structure—the “Adjacent Degree-2 Rule”, that is, Reduction Rule 2.1.

Claim 2.6. $G[O]$ consists of a union of paths, where each path has at most 3 vertices.
2. THE COMPLEXITY ECOLOGY OF PARAMETERS

Proof. Otherwise, Reduction Rule 2.1 applies. □

The evident structure of \( G[O] \) suggests looking for a reduction rule that applies to vertices of degree 1 in \( G \)—the “Degree-1 Rule”, that is, Reduction Rule 2.2.

**Claim 2.7.** \(|O| \leq 0.75k\)

**Proof.** Let \( T \) be a witness tree with \( k \) leaves, and induct on the number of path components of \( G[O] \).

Consider some path component \( C \) of \( G[O] \); then by Claim 2.6, \( C \) has at most three vertices. By Claim 2.5, no inner vertex of \( C \) is adjacent to some leaf of \( T \). Any non-inner vertex of \( C \) is not adjacent to any internal vertex of \( T \), by Claim 2.1, and adjacent to at least two leaves of \( T \), by Reduction Rules 2.2 and 2.1. Thus, \( k \geq 4|O|/3 \), proving the claim. □

The picture that has emerged through Claims 2.1–2.7 is starting to give us a handle on how big \( G \) can be. Next we must bound the size of \( T \).

**Claim 2.8.** \(|B| \leq k - 2\).

**Proof.** By straightforward induction on the number of leaves. □

**Claim 2.9.** The subgraph of \( T \) induced by the vertices of a topological edge of \( T \) is an induced path in \( G \).

**Proof.** This is trivially true for an \( t \)-topo-edge where \( t = 0 \), so suppose \( t \geq 1 \). But then we can re-engineer \( T \) to have \( k + 1 \) leaves, as shown in Fig. 2.5. □

![Figure 2.5: Topo-edges are induced subgraphs.](image)

**Claim 2.10.** There are no \( t \)-topological edges in \( T \) for any \( t \geq 6 \).

**Proof.** Suppose we have a path \((b,j_1,\ldots,j_t,b')\) in \( T \) where \( b,b' \in B \) are branch vertices of \( T \), and the vertices \( j_i \) for \( i = 1,\ldots,t \) are subdivider vertices of \( T \), where \( t \geq 6 \). Let \( T_b \) denote the subtree of \( T \) “to the left” of \( b \), and let \( T_{b'} \) denote the subtree of \( T \) “to the right” of \( b' \). The vertex \( j_3 \) cannot be adjacent in \( G \) to a vertex of \( T_{b'} \), otherwise we can re-engineer \( T \) to have \( k + 1 \) leaves as shown in Fig. 2.6. □

The vertex \( j_3 \) cannot be adjacent to a vertex of \( T_{b'} \), for similar reasons. By Claim 2.9, and by symmetry, the vertices \( j_3 \) and \( j_4 \) must have degree 2 in \( G \). But then \( G \) is reducible, either by contracting a bridge or by the “Adjacent Degree-2 Rule”, that is, Reduction Rule 2.1.

**Claim 2.11.** Each leaf in \( T \) is adjacent in \( T \) to a branch vertex.
2.2. ATTACKING A ROW: KERNELIZATION

2.2.1 Long topo-edges of $T$ have middle vertices of degree 2 in $G$.

Figure 2.6: Long topo-edges of $T$ have middle vertices of degree 2 in $G$.

2.2.2 A leaf is adjacent to a branch, else smaller $T$.

Figure 2.7: A leaf is adjacent to a branch, else smaller $T$.

Proof. Otherwise, we would contradict that $T$ is as small as possible. See Fig. 2.7. □

Claim 2.12. $|J| \leq 5(k - 3)$.

Proof. This follows from Claims 2.9 and 2.10. □

We can now conclude the proof of Boundary Lemma I on the basis of Claims 2.7, 2.8 and 2.12. □

2.2.3 Boundary Lemma II

Lemma 2.5 (Boundary Lemma II). Let $(G,k)$ be a reduced “yes”-instance of $(k)$-MAX LEAF SPANNING TREE such that $(G,k+1)$ is a “no”-instance. Then $|V(G)| \leq 5.75k$.

Proof. The proof is by minimum counterexample. Witnessing that $(G,k)$ is a “yes”-instance we have a tree $T$ with $k$ leaves, as in the proof of Boundary Lemma I. Here we will have a little more structure and another inductive priority. We consider that the tree $T$ is equipped with a root vertex $r \in B$. The possible counterexample that we entertain in our argument is one where

1. $T$ is as small as possible, and among all counterexamples satisfying this requirement, one where

2. the sum over all leaves $l \in L$ of the distance in $T$ from $r$ to $l$ is minimized.

All of the structural claims from the proof of Boundary Lemma I hold here as well, since essentially all we have done is add one further inductive priority. This additional priority allows us to establish a strengthening of Claim 2.10.

Claim 2.13. $T$ does not have $t$-topological edges for any $t \geq 4$. 
Proof. Suppose we have a path \((b, j_1, \ldots, j_t, b')\) in \(T\) where \(b, b' \in B\) are branch vertices of \(T\), and the vertices \(j_i\) for \(i = 1, \ldots, t\) are subdivider vertices of \(T\), where \(t \geq 4\). Let \(T_b\) denote the subtree of \(T\) “to the left” of \(b\), and let \(T_{b'}\) denote the subtree of \(T\) “to the right” of \(b'\). We can suppose that the root of \(T\) lies in \(T_{b'}\), without loss of generality. The vertices \(j_1\) and \(j_2\) cannot be adjacent to vertices in \(T_{b'}\), else we can re-engineer \(T\) to have \(k + 1\) leaves, as in the proof of Claim 2.10. By Claim 2.9, and since \(G\) is reduced (in particular, \(j_1\) and \(j_2\) do not both have degree 2 in \(G\)), at least one of \(j_1\) or \(j_2\) is adjacent by an edge \(e\) to a vertex \(x\) of \(T_b\). But then by adding \(e\) to \(T\) and removing the edge \(\{x, u\}\), where \(\{x, u\}\) is the first edge on the path in \(T\) from \(x\) to \(b\), we can obtain a modified tree with \(k\) leaves where the priority (2) has been improved. \(\square\)

That concludes the proof of Boundary Lemma II. \(\square\)

We next turn to the issue of how this understanding can be used to efficiently solve problems in our chosen row of the parameterized complexity ecology matrix.

**Proposition 2.1.** For graphs \(G\) of max leaf number bounded by \(k\), the domination number of \(G\) can be computed in time \(2^{1.44^k} \cdot n^{O(1)}\) based on a polynomial-time reduction to a kernel of size at most \(7.5^k\).

(We will give an asymptotically better result in Section 2.3).

Proof. Since this is an FPT result, we are necessarily (by Proposition 1.1) interested in effective kernelization for this problem. We must therefore develop a polynomial-time extremal account of the boundary case for the induction.

We take the following hypotheses:

1. \((G, k)\) is a “yes”-instance of \((k)\)-Max Leaf Spanning Tree.
2. \((G, k + 1)\) is a “no”-instance of \((k)\)-Max Leaf Spanning Tree.
3. There is a witness structure for (1) that satisfies the inductive priorities of the proof of Boundary Lemma II.
4. \(G\) is “reduced” according to an admissible set of polynomial time kernelization rules.

Here by **reduced** we mean polynomial-time reduction rules that are compatible with the objective of computing a minimum dominating set. Many of the structural claims proved above can be imported to this new situation, modified in some cases because of changes to the admissible set of reduction rules.

Here, because we are computing a minimum dominating set, we are allowed reduction rules that are compatible with this objective: rules that transform \(G\) to \(G'\) in such a way that given a minimum dominating set for \(G'\), we can easily compute a minimum dominating set for \(G\). To the extent that we can find reduction rules for this new computational objective that “mimic” or approximate the ones that were available for the \((k)\)-Max Leaf Spanning Tree problem, the structural claims about the kernel still (with some modifications) carry over, and we can conclude similar kernel size bounds for problems in the row of the complexity ecology matrix that are amenable to this approach.
The reduction rules shown in Fig. 2.8 can be used in this way for the Dominating Set problem, for graphs of bounded max-leaf number. To solve the minimum domination problem on graphs of bounded max-leaf number, we attempt to solve the decision problem \((l)-\text{Dominating Set}\) for each \(l\). For each \(l\), we iteratively apply the reduction rules for \((l)-\text{Dominating Set}\) given in Fig. 2.8 until they can no longer be applied. Since the reduction rules given in Fig. 2.8 can be performed by a series of vertex deletions, edge deletions, and edge contractions, the resulting reduced graph \(G'\) is a minor of the original graph \(G\). Since the max-leaf number of a graph is a monotonically non-decreasing property of the graphs in the minor ordering (that is, if \(H\) and \(G\) are connected and \(H\) is a minor of \(G\), then \(ml(H) \leq ml(G)\)), we have that \(ml(G') \leq ml(G) \leq k\). In particular, the resulting reduced graph \(G'\) has no induced paths of length more than three.

The argument for the bound on the kernel size is by minimum counterexample. One of our hypotheses is that \((G, k)\) is a “yes”-instance for \((k)\)-Max Leaf Spanning Tree. We can assume we are given as a witness structure a tree subgraph \(T = (V', E')\) of \(G\) that has \(k\) leaves, and we can also assume that \(G\) is connected.

We do not assume that \(T\) is a spanning subgraph. (If \(T\) is not spanning, then it clearly extends to a spanning tree \(T'\) for \(G\) that has at least \(k\) leaves.)

A counterexample to our theorem would be a graph \(G = (V, E)\) such that

1. \((G, k)\) is a reduced instance of \((k)\)-Max Leaf Spanning Tree,
2. \((G, k)\) is a “yes”-instance of \((k)\)-Max Leaf Spanning Tree,
3. \((G, k + 1)\) is a “no”-instance, and
4. \(|V(G)| > 7.5k\).

Among all such counterexamples, we consider one where a rooted witness subgraph tree \(T\) is as small as possible, and meeting the secondary inductive priority of Boundary Lemma 2.5. We now consider how the various Claims of the \((k)\)-Max Leaf Spanning Tree kernel size bound fare under the modified notion of “reduced instance” that we must consider here in order to solve the Dominating Set problem (that is: we consider the structure of an instance that is reduced with respect to the similar but modified reduction rules depicted in Fig. 2.8).

Claims 2.1–2.5, as well as 2.8.2.9 and 2.11, hold by the same arguments as before. We next argue that modified versions of the other structural claims hold, yielding our claimed kernelization bound.

**Claim 2.6'.** \(G[O]\) consists of a union of paths, where each path has at most 4 vertices.
2. THE COMPLEXITY ECOLOGY OF PARAMETERS

Proof. Otherwise, \((l)\)-Dominating Set Reduction Rule 1 could be applied, contradicting our hypothesis that the instance is reduced. 

Claim 2.7'. \(|O| \leq 1.5k\).

Proof. The argument for Claim 2.7 is modified by noting that the “worst case” is where a component of \(G[O]\) consists of a three-vertex path, where one endpoint is a vertex of degree 1, and the other endpoint has at least two leaf neighbors. 

Claim 2.13'. The witness tree \(T\) does not have \(t\)-topological edges for any \(t \geq 5\).

Proof. The argument for Claim 2.13 is modified (where we use the same notation to discuss the situation) by noting that none of the vertices \(j_1, j_2\) or \(j_3\) can be adjacent to vertices in \(T_b\), else we can re-engineer \(T\) to have \(k + 1\) leaves. These three vertices cannot all have degree 2, else Reduction Rule 1 applies. By Claim 2.9, at least one of these must be adjacent to a vertex \(x\) in \(T_b\), providing an opportunity to improve the secondary inductive priority.

The kernel can be analyzed by means of an \(O(1.5048^n)\)-time algorithm by van Rooij et al. [240], yielding the running time stated for our algorithm. Knowing the domination number of the kernel allows us to compute the domination number of the input graph by retracing this information backwards along the kernelization path in polynomial time.

What was the best previous result for this problem? A graph of bounded max leaf number has bounded pathwidth, and thus almost all of the entries in the “max leaf row” can be handled by standard bounded pathwidth algorithmics. This is true, but applied in a simple manner, entails a cost of \(2^{35k^3} \cdot n^{O(1)}\) in order to compute the path decomposition. The point of this investigation is to ask for the best possible FPT-algorithms for FPT entries in this row, that is, how does one best (and hopefully, systematically) exploit bounded max leaf number?

The following theorem is based on essentially the same approach, making use of the reduction rules shown in Fig. 2.9.

![Figure 2.9: Reduction rules for \(l\)-INDEPENDENT SET.](image)

**Theorem 2.1.** For graphs \(G\) of max leaf number bounded by \(k\), the independence number of \(G\) can be computed in time \(2.972^k \cdot n^{O(1)}\) based on a polynomial-time reduction to a kernel of size at most \(7k\).
Proof. The imported structural claims give a bound of 4.5k on the size of a vertex cover for the kernel, which yields the claimed running time by using the $O(1.2738^k + kn)$-time algorithm of Chen et al. [56] to analyze the kernel. We must first note that all of the reduction rules shown in Fig. 2.9 are admissible in the sense that they preserve a bound on the max-leaf number. An upper bound of 4.5k on the size of a minimum vertex cover is demonstrated by considering a vertex cover that consists of the leaves of $G$ (at most $k$), the branch vertices of the tree $T$ (at most $k$), together with at most 2$k$ vertices of the $J$ vertices (based on Claim 2.13’, as in the proof of Theorem 2.1), and together with at most $k/2$ vertices of $G[O]$. □

Many other NP-hard problems can be addressed for graphs of bounded max-leaf number in much the same way.

**Theorem 2.2.** For graphs $G$ of max leaf number bounded by $k$, it can be determined in time $54^k \cdot n^{O(1)}$ whether $G$ has a Hamilton cycle, based on a polynomial-time reduction to a kernel of size at most 5.75k.

Proof. Reduction rules admissible for the problem of deciding whether $G$ has a Hamilton cycle (because they preserve the bound on ml($G$)) include:

- If there is a vertex of degree 1, then $G$ is a “no”-instance.

- Edges between vertices of degree 2 can be contracted.

This leads to a kernel size bound identical to that for ($k$)-Max Leaf Spanning Tree (via Boundary Lemma II), since all of the structural claims hold in this case by the same arguments. The analysis of the kernel is by means of the dynamic programming algorithm of Held and Karp [156]. □

### 2.3 Attacking a Row: Win/Wins

Several of our results in the previous section based on the combination of systematic kernelization and exponential analysis of the kernel can be improved by what has been called a “Win/Win” algorithm that in polynomial time provides a useful connection between two different parameters [97, 210]. An early example in the study of parameterized algorithms is the linear time algorithm described by Fellows and Langston that given a graph $G$ outputs either a cycle of length at least $k$, or a tree decomposition of width at most $k$ [96]. To emphasize—the algorithm runs in $O(n)$ time for any $k \leq n$.

**Theorem 2.3.** There is a linear time algorithm that on input a connected graph $G$, outputs either

- a spanning tree of $G$ having at least $k$ leaves, or

- a path-decomposition of $G$ of width at most $2k$.

Although the proof is easy, we have not been able to find this already in the literature.
Theorem 2.3. Compute a breadth-first spanning tree of $G$. If any layer of the tree has population at least $k$, then the breadth-first spanning tree computed in this way has at least $k$ leaves. Otherwise, suppose the layers of the spanning tree are $L_0, L_1, \ldots, L_r$. The series of bags

$$L_0 \cup L_1, L_1 \cup L_2, \ldots, L_{r-2} \cup L_{r-1}, L_{r-1} \cup L_r$$

gives a path decomposition of $G$ of width bounded by $2k$. □

This yields more efficient FPT algorithms for some (but not all) of the problems we have considered. For example, combining the above algorithm with the dynamic programming algorithm for Dominating Set of Telle and Proskurowsky [236] (refined by Alber and Niedermeier [12]), we get an FPT-algorithm for the Dominating Set problem for graphs of max leaf number bounded by $k$ that runs in time $4^k \cdot n^{O(1)}$. Similarly, determining whether a graph of max leaf number bounded by $k$ has a proper 3-vertex-coloring can be accomplished in time $9^k \cdot n^{O(1)}$.

2.4 Attacking a Row: Well-Quasi Ordering

One of the most powerful FPT classification tools is well-quasi ordering. Graphs in general are well-quasi ordered by minors (the celebrated Graph Minor Theorem), and also importantly, determining whether a graph $H$ is a minor of a graph $G$, parameterized by $H$, is non-uniformly FPT (we say that the minor order has FPT order tests) [222, 223]. Bounding a parameter, such as the max leaf number, can lead to “even more powerful”—but generally easier to prove—well-quasi ordering FPT classification tools applicable to a given row of the parameterized complexity ecology matrix. Fellows [97] shows that graphs of bounded vertex cover number are well-quasi ordered by induced subgraphs and admit linear-time FPT order tests. Here we prove an analogous result, and give an application. For background on well-quasi ordering concepts and standard methods of proof in the context of FPT algorithms, we refer the reader to Downey and Fellows [82].

Definition 2.1. The topological order on multigraphs is defined as

$$H \leq_{top} G : \iff G \text{ contains a subgraph that is isomorphic to a subdivision of } H.$$  

Let $F_k$ denote the set of graphs $G$ for which $ml(G) \leq k$.

Theorem 2.4. For every $k$, the set $F_k$

(1) is well-quasi ordered by the topological ordering, and

(2) admits linear time FPT order tests.

Proof. By a theorem of Kleitman and West [173] every graph $G$ with $ml(G) \leq k$ is a subdivision of a graph on at most $4k - 2$ vertices. Suppose there is an infinite bad sequence of graphs of bounded max leaf number in the topological order. Because the number of distinct graphs on at most $4k - 2$ vertices is bounded by a function of $k$, by the Pigeonhole Principle there is an infinite bad subsequence $(G_1, G_2, \ldots)$
where for all \(i\), \(G_i\) is a subdivision of a fixed graph \(G\) on \(k' \leq 4k - 2\) vertices. Let \((e_1, \ldots, e_m)\) be a fixed enumeration of the edges of \(G\). Then each \(G_i\) is essentially described by the information

- the graph \(G\), and
- a length-\(m\) census vector \((s_{i,1}, \ldots, s_{i,m})\) that records the numbers \(s_{i,j}\) of \(2\)-vertices on the path that replaces the edge \(e_j\) in order to produce \(G_i\) from \(G\).

Clearly, if for \(i < i'\), and for all \(j\), \(s_{i,j} \leq s_{i',j}\), then \(G_i \leq_{\text{top}} G_{i'}\). By Higman’s Lemma [158], this has to happen, contradicting that the sequence is bad. This establishes (1). By a theorem of Bienstock et al. [31] (see also Section 2.3) graphs of bounded max leaf number have bounded pathwidth, and since for a fixed graph \(H\), the property of containing \(H\) topologically is expressible in MSO logic, we have (2).

We next describe an application to the problem Topological Bandwidth. Given a graph \(G\), it can happen that a subdivision \(G'\) of \(G\) has smaller bandwidth than the bandwidth of \(G\). Determining if \(\text{tbw}(G) \leq k\), parameterized by \(k\), is hard for \(W[t]\) for all \(t\), even for the restriction to trees [35]. We address the following decision problem in our row of the parameterized complexity ecology matrix:

**Topological Bandwidth By Max-Leaf Number**

**Input:** A graph \(G\) with \(ml(G) \leq k\), and a positive integer \(r\).

**Parameter:** \(k\).

**Question:** Is \(\text{tbw}(G) \leq r\)?

**Theorem 2.5.** Topological Bandwidth By Max-Leaf Number is fixed-parameter tractable.

**Proof.** It is sufficient to show that if \(ml(G) \leq k\) then \(\text{tbw}(G) \leq k'\) where \(k' = f(k)\) depends only on \(k\), where the function \(f(k)\) is computable. Suppose this is shown. Then we compute \(k'\) (in time that depends only on \(k\)) and if \(r \geq k\), then \((G, r)\) is a “yes”-instance. Otherwise, by the well-quasi ordering of \(\mathcal{F}_k\) under \(\leq_{\text{top}}\), there is a finite obstruction set \(O_r\) such that \(\text{tbw}(G) \leq r\) if and only if for every graph \(H \in O_r\), \(H\) is not topologically contained in \(G\). By (2) of the above theorem, this is sufficient to conclude that Topological Bandwidth By Max-Leaf Number is solvable in linear-time non-uniform FPT. This result can be converted into uniform FPT by either of the methods of Fellows and Langston [96, 98], both of which are exposited by Downey and Fellows [82].

To show what we propose, we argue that the bandwidth of \(G\) is bounded by \(k' = (4k - 3) + 2^{\left(\frac{4k-2}{2}\right)}\), so that this is an upper bound on \(\text{tbw}(G)\). For this we can use the fact that \(G\) is a subdivision of a graph \(H\) on at most \(4k - 2\) vertices (by the theorem of Kleitman and West [173]). Let \(v_1, \ldots, v_n\) be the vertices of \(H\), and let \(e_1, \ldots, e_m\) be the edges of \(H\).

A layout of \(G\) is an injection \(l : V(G) \to \mathbb{Z}\), and the bandwidth of a layout \(l\) is the maximum, over all pairs \(u, v\) of adjacent vertices of \(G\), of \(|l(u) - l(v)|\). We describe a layout of \(G\) that achieves the claimed bound:

1. Assign vertex \(v_i\) the position \(l(v_i) = -i\).
(2) Associate to each edge $e_j$ the set of positive integers

$$P_j = \{ r \mid r \equiv j - 1 \mod m \} .$$

(3) As $G$ is obtained from $H$ by replacing each edge $e_j$ with a path $p_j$, complete the description of the layout $l$ of $G$ by assigning the vertices of the path $p_j$ to positions in $P_j$ in the natural way. This means that the path goes “out and back” in an interleaved manner.

It is straightforward to check that the bound is achieved. □

2.5 Attacking an Entry: Bandwidth Parameterized by Max-Leaf Number

The Bandwidth problem, parameterized by solution value, is known to be hard for $W[t]$ for all $t$ [35]. Since a graph of bandwidth at most $b$ has pathwidth at most $b$, it follows that if we parameterize the Bandwidth problem by the pathwidth of the input graph, then this is also hard for $W[t]$ for all $t$. In this section we show that the following problem is FPT:

**Bandwidth By Max-Leaf Number**

*Input:* A graph $G$ for which $ml(G) \leq k$, and a positive integer $r$.

*Parameter:* $k$.

*Question:* Is $bw(G) \leq r$?

**Theorem 2.6.** Bandwidth by Max-Leaf Number is fixed-parameter tractable.

*Proof.* As formal details would be arduous, we provide only a sketch of a proof. The overall structure of our algorithm is that we branch on an exhaustive set of “solution plans” that has size bounded by a function of $k$, and that is sure to capture at least one solution, if any exists. Then, to determine whether a particular solution plan can be realized, we employ a subroutine that consists of solving (in polynomial time) a system of linear equations. We use the fact that if $G$ has $ml(G) \leq k$ then $G$ is a subdivision of a graph $H$ on at most $4k - 2$ vertices [173]. We also use the fact that if $ml(G) \leq k$, then $bw(G) \leq k' = O(k^2)$, established in the proof of Theorem 2.5. In particular, we have the initial observation:

**Observation 2.1.** If $r > k'$ then $(G, r)$ is a “yes”-instance.

A solution plan, or plan, consists of two layers of specification: (A) a topological and local specification of the general shape of a solution, and how it might be carried out in key local areas, and (B) in the remaining gaps between these local specifications, a local plan/formula for negotiation, that contributes to a global calculation of whether the solution plan can be achieved. Part A of a solution plan consists of the following items of information:

(A.1) A permutation of the vertices of $H$ representing the intended order of these vertices in a layout of $G$. (Note that all the other vertices of $G$ are vertices of degree 2 on paths that replace the edges of $H$ in order to create $G$.) We consider the vertices of $H$ to be reference points of the plan.
(A.2) A description of the topological routing of the paths that the edges of $H$ are replaced by in producing $G$ from $H$. For an example, see the upper part of Fig. 2.10. In such a plan, we are only interested in the zones that the routes of the paths cross or enter, and we allow that the route of the path between vertices $u$ and $v$ of $H$ may make up to two changes of direction. We argue below that if there is any solution, then there is a solution in which each path-route makes at most two changes of direction. This issue is illustrated in Fig. 2.11.

Figure 2.10: Part A of a solution plan. The vertical lines indicate an injective mapping of the vertices into integer positions on the real line. The square vertices represent “points of change of direction” of the topological routing.

Figure 2.11: No wiggle-waggle necessary.

(A.3) If a plan calls for a path to make a change of direction, then this creates a new reference point. The information for a plan includes a permutation of all of the reference points (the vertices of $H$, together with all of the points that represent changes of direction of the routes of the paths).

(A.4) A plan also completely specifies (we will use the phrase: reference point locally) the layout of the subgraph of $G$ that is induced by the vertices within distance at most $r$ from any reference point, allowing that these neighborhoods of adjacent reference points may be coalesced, in the case that the
reference points are planned to be close together. This reference-point local specification for a particular reference point, details how various 2-vertices of the edge-paths present in that region of the topological plan are ordered among the 2r layout positions in the (relative) local neighborhood of the reference point. (Note that this reference-point local specification requires that the numbers of degree two vertices on the various paths that build G from H are sufficient to meet the specification, else the plan can be discarded from our search of the possibilities for a solution.) See the lower part of Fig. 2.10.

We claim that the number of solution plans (Part A) is bounded by a function of k. Let us start with the number of possible reference-point local specifications that might be given a specific reference point. Since \( r \leq k \) by the initial step of the algorithm, there are at most 2^k positions to be assigned to the edge-paths routed in the neighborhood of the reference point by the topological part of the solution plan, so we have a bound of \( m^{2k} \), where \( m \) is the number of edges in \( H \), which gives a crude bound of \( O(k^2) \). Since there are \( O(k^2) \) reference points, this gives a bound of \( k^{O(k^2)} \) for the number of possible reference-point local specifications to be explored for any given permutation of the reference points. The number of such permutations (which implicitly bounds the number of topological plans) is bounded by \( O(k^2)! = k^{O(k^2)} \). So the total number of solution plans (part A) is bounded by \( k^{O(k^2)} \). We will use \( k'' \) to designate this bound.

Crucial to the correctness of our algorithm is the following claim.

**Lemma 2.6.** If \( G \) has any layout of bandwidth at most \( r \), then it has one where each edge-path makes at most two turns.

**Proof.** Let \( f : V(G) \to \mathbb{Z} \) be a layout of \( G \) with \( f(v) > 0 \) for all \( v \in V(G) \), witnessing that the bandwidth of \( G \) is at most \( r \). As discussed above, we view \( G \) as a subdivision of a graph \( H \) on at most \( 4k - 2 \) vertices. Suppose \( x \) and \( y \) are adjacent vertices of \( H \), and denote the path of 2-vertices in \( G \) between \( x \) and \( y \) by \( v_i \). Thus in \( G \) we have the “edge-path” \( P(x, y) = (x, v_1, v_2, \ldots, v_m, y) \). We will use \( S \) to denote the set of vertices \( v_i \).

We describe how to modify \( f \) to obtain a different function \( f' \) that lays out the edge-path in a way that involves at most two turns. Let \( P \) denote the set of positions \( f(S) \). Since \( f \) is injective, \( |P| = m \). Our modification of \( f \) consists in mapping \( S \) to \( P \) in a different way that involves at most two turns. Making a similar modification for each edge-path of \( G \), since these modifications can be made independently, yields the lemma.

For convenience, suppose \( f(x) < f(y) \). We can consider that \( P \) is partitioned into three sets

\[
P = P_0 \uplus P_1 \uplus P_2,
\]

where \( P_0 \) is the set of positions in \( P \) that are less than \( f(x) \), \( P_1 \) is the set of positions in \( P \) that are between \( f(x) \) and \( f(y) \), and \( P_2 \) is the set of positions in \( P \) that are greater than \( f(y) \). Let \( m_i = |P_i| \), for \( i = 0, 1, 2 \).

Let \( p(0, i) \) for \( i = 1, \ldots, m_0 \) denote the positions in \( P_0 \), sorted according to increasing distance (“to the left”) from \( f(x) \). Similarly, let \( p(1, i) \) for \( i = 1, \ldots, m_1 \) denote the positions in \( P_1 \) sorted by increasing distance (“to the right”) from \( f(x) \),
and let $p(2,i)$ for $i = 1,\ldots,m_2$ denote the positions in $P_2$ sorted by increasing distance (“to the right”) from $f(y)$.

For simplicity, we will assume that all of the sets of positions $P_i$ are nonempty and have size at least 2. (It is easy to modify the argument to handle the other cases.) Our description of $f'$ is summarized: we assign the vertices of $S$ “out and back” in an interleaved manner to the positions in $P_0$, then progressively to the positions in $P_1$, and then “out and back” in an interleaved manner to the positions in $P_2$. In particular, we make $f'(v_1) = p(0,2), f'(v_2) = p(0,4), \ldots, f'(v_i) = p(0,2i), \ldots$ until we reach the leftmost position of $P_0$, and then progressively back through the alternately skipped positions, and similarly with regards the positions in $P_2$. We say that positions $p(0,i)$ and $p(0,j)$ are nearly consecutive if $|i - j| = 2$. We argue that the distance between nearly consecutive positions in $P_0$ is at most $r$ (and similarly, that the distance between nearly consecutive positions in $P_2$ is at most $r$). This is true, because otherwise, $f$ would fail to be a layout of bandwidth at most $r$. □

Before we discuss what constitutes Part B of a solution plan, we need to take account of what remains to be determined if a solution plan (Part A) is to be realized. What Part A specifies is an ordering of the reference vertices together with a description of what a solution layout should look like in the vicinity of the reference vertices. This implicitly involves a commitment to some numbers of subdivisions on the edges of $G$, and the commitment is feasible only if the number of subdivisions of each edge of $H$ (in the description of $G$) is greater than or equal to the commitment implicit in Part A of a solution plan. Fix attention on solution plan $\mathcal{P}$, partially specified by the Part A information. For each edge $e$ of $H$, let $P(e)$ denote the number of “further” subdivisions (i.e., beyond those already implied in Part A) needed on edge $e$ in order to reach the number in the description of $G$ as a subdivision of $H$. What remains to be determined is schematically illustrated in Fig. 2.12.

![Figure 2.12: The situation at a gap.](image-url)

In the figure, the boxes represent the local solution information specified in Part A (that is, local to the reference vertices). Between these boxes are tracks representing the edge-paths that go between these areas. Each track is part of the routing of a edge-path in the topological specification of Part A. Note that the situation between two consecutive boxes has a simple structure: there is just a set of edge-paths between the two boxes. We refer to the situation between two boxes
as a gap. Let $G_i, i = 1, \ldots, t$ denote the set of gaps of the Part A, partial solution specification. Abusing notation in harmless way, we will treat $G_i$ as denoting the set of edge-paths in the gap.

What remains to be determined is whether subdivisions can be introduced to the edge-paths in the gaps, in a way that is locally consistent with a bandwidth-$r$ layout, and so the total number of subdivisions introduced for each edge $e$ of $H$, sums to $P(e)$, summing over the gaps that include $e$. Let $m(i, e)$ denote the number of subdivisions introduced to the track representing $e$ in the gap $G_i$.

What we require, then, is that

$$\sum_{i: e \in G_i} m(i, e) = P(e), \quad \text{for all } e \in E(G),$$

subject to local consistency with bandwidth at most $r$.

We make this determination, algorithmically, as follows. First of all, for each edge $e$ of $H$, $P(e)$ is a constant that we calculate for $G$ according to Part A of the solution plan specification. The $m(i, e)$ are treated as integer variables, and we assemble a system of linear equations that has a feasible solution if and only if the solution plan partially specified by Part A, can be carried out. Clearly, for each gap $G_i$, local consistency with layout of bandwidth at most $r$ depends on a suitable relationship between the values of the variables

$$M(i) = \{m(i, e) \mid e \in G_i\}.$$

The local consistency constraints are expressed as a set of linear equations, using variables special to Part B of a solution plan specification. We next study the situation for a specific gap $G_i$. Refer to Fig. 2.13. We may consider that we “build”

![Figure 2.13: Example of a gap automaton.](image)

the part of the solution in the gap from left to right. Each step consists of either
terminating the process, or introducing a subdivision on one of the tracks. Our local consistency concern is that no edge of $G$ is stretched more than a distance $r$. Because of the Part A specification, we begin with a state based on how much each track is “already stretched” (to the left) due to the specification by Part A of the box on the left of the gap. Each step changes the state. There are at most $|G| \cdot r$ states that are consistent with layout of bandwidth at most $r$. When we terminate, we have to consider whether the state information concerning “stretch to the left” is compatible with the stretch imposed by the specification of Part A concerning the box on the right of the gap.

We can model the situation with a finite-state machine $M_i$. The alphabet of $M_i$ corresponds to the tracks of the layout. Processing a letter by $M_i$ corresponds to introducing a subdivision (the “next vertex to be laid out in the gap”) to one of the tracks, the accept states correspond to the states (based on left-stretch) that are compatible with the box on the right of the gap (and hence are acceptable for terminating the local construction of a partial solution). Local consistency of the values of the integer variables of $M(i)$ corresponds to $M_i$ accepting a word whose letter content is the same as the values of the variables of $M(i)$. Fig. 2.13 shows an example of a gap automaton.

In a directed graph $D$, define paths $\Delta$ and $\Delta'$ from a vertex $s$ to a vertex $t$ to be arc-equivalent, if for every arc $a$ of $D$, $\Delta$ and $\Delta'$ pass through the arc $a$ the same number of times. We need the following lemma.

**Lemma 2.7.** Any directed path $\Delta$ through a finite directed graph $D$ on $n$ vertices from a vertex $s$ to a vertex $t$ of $D$ is arc-equivalent to a path $\Delta'$ from $s$ to $t$, where $\Delta'$ consists of

1. an underlying path $\rho$ from $s$ to $t$ of length $O(n^2)$, together with
2. some numbers of short loops, where each such short loop $l$ begins and ends at a vertex $v$ of $\rho$, and has length at most $n$.

**Proof.** We give an algorithmic proof. Consider the sequence $\sigma$ of vertices visited by $\Delta$ from $s$ to $t$ in $D$. If $\sigma$ has length greater than $n$, then $\sigma$ contains a short loop relative to a vertex $u$ (i.e., $u$ is repeated in $\sigma$). Let $\sigma'$ denote $\sigma$ reduced by this short loop. Repeating this, one obtains some numbers of short loops rooted at various vertices, together with a final reduced sequence $\sigma^*$ of length at most $n$. Loops are either rooted at a vertex $x$ of $\sigma^*$ (in which case we are done) or not. If not, then an augmentation of $\sigma^*$ by a short loop that includes the root $x$ (for each such $x$) is sufficient to conclude the lemma. 

We are now in a position to describe Part B of a solution plan. Part B specifies, for each gap $G_i$, a transition path $\rho_i$ from the start state, to an accept state of the gap automaton $M_i$, of length at most the square of the number of states of $M_i$.

The number of solution plans (Part A + Part B) is bounded by $k^{O(k)}$. For each fully-articulated solution plan, the only thing yet undetermined is the number of times any possible (short) loop might be executed, in each gap. We express our feasibility constraints in terms of one variable for each such possible loop, for each gap, with a linear equation summing appropriately, the variables of $M(i)$, and we adjust the global linear constraints of the equations of (1) appropriately,
according to the subdivisions of the various edge-paths implicit in Part B of a solution plan specification (not already accounted for by the calculations relative to Part A). The number of variables involved in the system of linear equations for a fully articulated solution plan is crudely bounded by the number of short loops that are possible; for an \( n \) state gap automaton this is bounded by \( n^n \). The number of variables is thus also crudely bounded by \( k^{O(k^2)} \).

We must argue that if there is any solution, then one of our branches will succeed. This follows from Lemma 2.6, and the fact that what a solution \( f \) “does” in a gap corresponds directly to a path \( \rho \) from the start state to an accept state in the automaton for that gap, where the numbers of subdivisions introduced add up to \( p(e) \) for each edge \( e \) of \( G \). Lemma 2.7 shows that \( \rho \) can be replaced by a different path \( \rho' \) that has the same letter-content as \( \rho \), where \( \rho' \) has the form specified by Part B. Corresponding to \( \rho' \) is a different solution \( f' \) that is described by a specification of the form (Part A + Part B), together with the numbers of times various short loops are traversed (which implicitly describes the solution \( f' \) in the various gaps).

In this way we are able to solve the problem in FPT-time by branching on at most \( k^{O(k^2)} \) solution plans, and in each case checking feasibility by the solvability of a system of linear equations. □

2.6 Attacking an Entry: Fixed-Parameter Approximation

Proving a certain entry in the ecology matrix to be \( W[1] \)-hard means that we have to look for algorithmic approaches other than fixed-parameter algorithms to solve the problem \( \Pi \) corresponding to that entry. We explore whether we can at least find fixed-parameter approximation algorithms for parameterized approximation versions of \( \Pi \).

Our first result is that for \((k)\)-CONNECTED DOMINATING SET, which is \( W[2] \)-hard \([87]\), additive FPT-approximation algorithms are unlikely to exist.

**Theorem 2.7.** CONNECTED DOMINATING SET \((k + c)\)-Approximation is \( W[2] \)-hard for any integer \( c \geq 0 \).

*Proof.* We give a parameterized reduction from the \( W[2] \)-hard \((k)\)-SET COVER problem.

Let \((S, F, k)\) be an instance of \((k)\)-SET COVER. We produce a connected graph \( G \) that has a connected dominating set of size at most \( k' + c \) if and only if \((S, F)\) has a set cover of size at most \( k \), where \( k' = (c + 2)k + 1 \). Construct \( G \) by first taking \( c + 2 \) copies of the bipartite set-element incidence graph of \((S, F)\), the copies labeled by \( G_1, \ldots, G_{c+2} \), and second create new vertices \( y, z \) and connect \( y \) by edges to \( z \) and all vertices corresponding to sets in \( F \).

First, suppose that \( F' \) is a size-\( k \) set cover for \((S, F)\). Then the set \( D = \{ y \} \cup \bigcup_{i=1}^{c+2} F'_i \) is a connected dominating set of \( G \) with size \( k' \), where \( F'_i \) is the projection of \( F' \) in copy \( G_i \).

Conversely, if \( D \) is a minimum connected dominating set for \( G \) of size at most \( k' + c \) then we can assume, without loss of generality, that \( z \notin D \) and \( y \in D \). Then
one of the \( c + 2 \) copies \( G_i \) witnesses the fact that \( (S,F) \) has a set cover of size at most \( k \): for if the minimum set cover corresponding to each copy \( G_i \) had size at least \( k + 1 \) then we obtain the contradiction

\[
\gamma_c(G) \geq (c + 2)(k + 1) + 1 = (c + 2)k + c + 2 + 1
\]

\[
> (c + 2)k + c + 1 \geq |D| = \gamma_c(G).
\]

\( \square \)

For \((k)\)-Dominating Clique, we can prove a stronger parameterized inapproximability result. The proof is inspired by Downey et al.'s proof [86] on the parameterized inapproximability of \((k)\)-INDEPENDENT DOMINATING SET.

**Theorem 2.8.** There is no fixed-parameter approximation algorithm for \((k)\)-DOMINATING CLIQUE for any computable function \( g(k) \), unless \( \text{FPT} = \text{W}[2] \).

**Proof.** We give a parameterized reduction from the \( \text{W}[2] \)-complete \((k)\)-DOMINATING SET problem.

Let \( G \) be the graph for which we wish to determine if it has a dominating set of size \( k \). Construct a graph \( G' = (V',E') \) as follows. The vertex set \( V' \) of \( G' \) consists of the following sets:

\[
S = \{ s[r,i] \mid 1 \leq r \leq k, 1 \leq i \leq g(k) + 1 \}, \quad \text{the sentinel vertices,}
\]

\[
C = \{ c[r,u] \mid 1 \leq r \leq k, u \in V(G) \}, \quad \text{the choice vertices,}
\]

\[
T = \{ t[u,i] \mid u \in V(G), 1 \leq i \leq g(k) + 1 \}, \quad \text{the test vertices.}
\]

The edge set \( E' \) of \( G' \) consists of the following sets:

\[
E'(1) = \{ \{ s[r,i], c[r,u] \} \mid 1 \leq r \leq k, 1 \leq i \leq g(k) + 1, u \in V(G) \},
\]

\[
E'(2) = \{ \{ c[r,u], c[r',u] \} \mid 1 \leq r < r' \leq k, u \in V(G) \},
\]

\[
E'(3) = \{ \{ c[r,u], t[v,i] \} \mid 1 \leq r \leq k, u \in V(G), v \in N[u], 1 \leq i \leq g(k) + 1 \}. \]

Central to the construction are \( k \) groups of choice vertices, each forming an independent set. Corresponding to each of these \( k \) independent sets is a set of \( g(k) + 1 \) sentinel vertices, and the edges of \( E'(1) \) connect each sentinel vertex to all of the vertices in its corresponding choice independent set. The sentinel vertices form an independent set in \( G' \), as do the test vertices. The edges of \( E'(3) \) connect the choice vertices to the test vertices in the natural way, reflecting the structure of \( G \).

We argue the correctness of the reduction. First, we claim that if \( G \) has a size-\( k \) dominating set then \( G' \) has a dominating clique of size \( k \). To see that, let \( D \) be a dominating set in \( G \) of size \( k \). It is easy to check that the corresponding vertices of \( D \) in \( G' \), one in each of the choice vertex independent set of \( G' \), form a size-\( k \) dominating clique in \( G' \).

Second, we claim that if \( G' \) has a dominating clique of size \( g(k) \) then \( G \) has a dominating set of size at most \( k \). Suppose that \( G' \) has a dominating clique \( D' \) of size at most \( g(k) \). There must be at least one vertex \( c[i,u] \in D' \) in each of the \( k \) choice vertex independent sets, \( 1 \leq i \leq k \), for otherwise the sentinels could not be
dominated. Since $D'$ is a clique we conclude that there is exactly one vertex of $D'$ in each of the $k$ choice independent sets. Let $D$ be the set of at most $k$ vertices of $G$ that these choice vertices of $D'$ indicate, that is, $D = D' \cap C$. (Notice that while there are $k$ choice independent sets, a vertex could be thus indicated more than once, so we only know that $|D| \leq k$.) We argue that $D$ is a dominating set in $G$. By the construction of $G'$, this follows from the fact that for all $u \in V$, there is at least one test vertex $t[v,i]$ corresponding to $v$ in $G'$ that does not belong to $D'$, and must be dominated by a vertex in $C$. The definition of the set of edges $E'(3)$ of $G'$ allows us to conclude that $D$ is a dominating set in $G$.□

2.7 Concluding Remarks

What we show in this chapter is an example of how to systematically explore a “row” of the parameterized complexity ecology matrix, through the example of bounded max leaf number. There are clearly many more rows of the matrix to be explored. One would like to see further development of systematic approaches as well as improved concrete results. A good starting point for further research seems to replace the question marks in the Genus column of the ecology matrix.
Kernelization

Preprocessing of data is one of the oldest and widely used methods in practical algorithms. Parameterized Complexity provides a natural way to measure the quality of preprocessing, by way of kernelization. Kernelization has been extensively studied, resulting in polynomial kernels for a variety of problems. Notable examples include a kernel for \((k)\)-Vertex Cover with \(2k\) vertices [55], a kernel for \((k)\)-Dominating Set in planar graphs with \(335k\) vertices [9] later improved to \(67k\) [54], and a kernel for \((k)\)-Feedback Vertex Set with \(O(k^5)\) vertices [237].

In this chapter we establish kernels of polynomial size for \((k)\)-Connected Dominating Set on certain sparse graph classes, namely on planar graphs, map graphs and \(K_{3,3}\)-minor free graphs. Let us remark that map graphs are not minor-closed. We further prove that unless \(\text{PH} = \Sigma^P_3\), on \(H\)-minor free graphs no polynomial kernel exists for Connected Dominating Set parameterized by solution size \(k\) and \(|H|\).

Connected Dominating Set is a well-studied NP-hard problem that finds applications in various network design problems. It remains NP-hard when restricted to the class of planar graphs [121], and has an \(O(\log n)\)-approximation algorithm [133]. The parameterized version \((k)\)-Connected Dominating Set is known to be \(W[2]\)-complete for general graphs [97], and admits a subexponential fixed-parameter algorithm for planar graphs [71]. In general graphs the problem has also been studied in the realm of moderately exponential time algorithms, leading to algorithms with running times \(O(1.9407^n)\) [108] and \(O(1.8966^n)\) [102].

3.1 Connected Dominating Set in Planar Graphs

We provide polynomial time reduction rules for \((k)\)-Connected Dominating Set in planar graphs and analyze these to obtain a linear kernel for \((k)\)-Connected Dominating Set in planar graphs. To obtain the desired kernel we introduce a method that we call \textit{reduce or refine}. Our kernelization algorithm analyzes the input graph and either finds an appropriate reduction rule that can be applied, or “zooms in” on a region of the graph which is more amenable to reduction. We find this method of independent interest and believe that it will be useful to obtain linear kernels for other problems on planar graphs.

A significant amount of research has gone into providing linear kernels for NP-hard problems on planar graphs. A foundation for linear kernels in planar graphs was built by Alber et al. [9] who gave a kernel for \((k)\)-Dominating Set on planar
graphs with 335k vertices. The main ingredient in the analysis of the reduced instance was the notion of “region decomposition” for the input planar graph where the number of “regions” depended linearly on the parameter. These ideas were later abstracted by Guo and Niedermeier [138] who gave a framework to obtain linear kernels for planar graph problems possessing a certain “locality property”. This framework has been successfully applied to yield linear kernels for (k)-Connected Vertex Cover, (k)-Edge Dominating Set, (k)-Vertex Disjoint Triangles, (k)-Efficient Edge Dominating Set, (k)-Induced Matching and (k)-Full-Degree Spanning Tree [138, 139, 199]. However, the framework proposed by Guo and Niedermeier [138] is not able to handle problems like (k)-Feedback Vertex Set and (k)-Odd Cycle Transversal because these do not admit the locality property required by the framework. Recently, Bodlaender and Penninkx [38] and Bodlaender et al. [39] have obtained linear kernels for (k)-Feedback Vertex Set and (k)-Vertex Disjoint Cycles on planar graphs respectively.

Until 2009, the list of problems for which linear kernels are known for planar graphs does not cover so-called “connectivity problems”, which demand the solution to be connected. The mere exception was (k)-Connected Vertex Cover, for which the same reduction rules as for (k)-Vertex Cover in planar graphs apply [138]. We filled this void by giving a linear kernel for (k)-Connected Dominating Set on planar graphs, which is presented in this section. We prove that

**Theorem 3.1.** (k)-Connected Dominating Set has a linear kernel on planar graphs.

In the meantime, it has become possible to derive this result also from the “meta”-kernelization of Bodlaender et al. [36].

Theorem 3.1 answers a question asked by Jiong Guo [135] during the visit of Saket Saurabh to Jena in 2007. Our result is based on the reduce-or-refine technique, that we introduce here. Until now the notion of region decomposition of planar graphs was used only in the analysis of the kernel size, and not explicitly applied in the reduction rules. We utilize the fact that a region decomposition can be obtained in polynomial time given a solution set S. In particular, we compute S using the known polynomial time approximation scheme for Connected Dominating Set in planar graphs and compute the region decomposition from S using algorithms described by Alber et al. [9], and Guo and Niedermeier [138].

The main technical part of our proofs is devoted to showing that if a region contains more vertices than a fixed constant, we can in polynomial time find a vertex in this region whose removal will not affect the size of an optimal solution. The idea is to check whether the region contains more than a fixed constant number of copies of a particular structure. If so then we can reduce the graph by removing a vertex in such a structure. If there are few or no copies of the structure in this region then we can zoom in on, or refine, to a smaller region that still contains many vertices but completely excludes the structure. The process is then repeated for a different “bad” structure until the region we have zoomed in on looks so simple that it is easy to identify a vertex to remove.

Since the number of regions in our computed region decomposition is $O(k)$, if the graph has too many vertices then we can identify a region in which a useless vertex can be found. Thus we obtain the desired linear upper bound on the size of the kernel.
3.1. CONNECTED DOMINATING SET IN PLANAR GRAPHS

3.1.1 Reductions and Plane Decompositions

In this subsection we collect necessary definitions and results required to obtain a linear kernel for the problem. We also give a few basic reduction rules for \((k)\)-Connected Dominating Set.

Let \(G\) be a connected plane graph. A path in \(G\) between distinct vertices \(v, w\) is called \([v, w]\)-path.

With respect to connected dominating sets, the following reduction rules will frequently help to simplify the input graph. If \(G\) has a universal vertex \(v\) then \([v]\) is a minimum connected dominating set for \(G\). Henceforth we assume that \(G\) has no universal vertex.

Lemma 3.1. Let \(G\) be a graph and let \(v\) be a vertex of \(G\) contained in some minimum connected dominating set \(S\) of \(G\). Let \(G_v\) be the graph obtained from \(G\) by removing the edges of \(G[N(v)]\). Then \(\gamma_c(G) = \gamma_c(G_v)\).

Proof. Let \(S\) be a minimum connected dominating set of \(G\) containing \(v\). In \(G_v\), any vertex \(u \in N(v) \setminus S\) is still dominated by \(v\); and any pair of vertices in \(N(v) \cap S\) are connected via \(v\). Hence \(S\) is a connected dominating set of \(G_v\).

Now we evoke the notions of a region and a region decomposition that were first introduced by Alber et al. [9].

Definition 3.1. Let \(G\) be a plane graph and let \(v, w\) be distinct vertices of \(G\). A region \(R(v, w)\) between \(v\) and \(w\) is a closed subset of the plane such that

- the boundary of \(R(v, w)\) is formed by two simple \([v, w]\)-paths each of length at most three, and
- all vertices strictly inside region \(R(v, w)\) belong to \(N(v) \cup N(w)\), and are called inner vertices of \(R(v, w)\).

We use \(V(R(v, w))\) to denote the set of inner vertices of the region \(R(v, w)\).

Definition 3.2. Let \(G\) be a plane graph and let \(S \subseteq V(G)\). An \(S\)-region decomposition of \(G\) is a set \(\mathcal{R}\) of regions \(R(v, w)\) between distinct vertices \(v, w \in S\) such that

- no region \(R(v, w)\) contains a vertex from \(S \setminus \{v, w\}\), and
- distinct regions can only intersect in their boundaries.

For an \(S\)-region decomposition \(\mathcal{R}_s\), let \(V(\mathcal{R}) = \cup_{R(v, w) \in \mathcal{R}} V(R(v, w))\). An \(S\)-region decomposition \(\mathcal{R}\) of \(G\) is maximal if there is no region \(R(v, w)\) such that \(\mathcal{R} \cup \{R(v, w)\}\) is an \(S\)-region decomposition of \(G\) satisfying \(V(\mathcal{R}) \subseteq V(\mathcal{R} \cup \{R(v, w)\})\).

We now state two known results about maximal region decompositions. The results say that given a plane graph \(G\) and a dominating set \(S\), one can obtain an \(S\)-region decomposition of \(G\) with \(O(\gamma_c(G))\) regions that together cover all but \(O(\gamma_c(G))\) vertices of \(G\).
Proposition 3.1 ([138]). Let $G$ be a plane graph and let $S$ be a dominating set of $G$. There exists a maximal $S$-region decomposition of $G$ containing at most $3\gamma(G)$ regions.

Proposition 3.1 has a constructive proof by a polynomial-time algorithm.

Proposition 3.2 ([9]). Let $G$ be a plane graph, let $S$ be a dominating set of $G$ and let $R$ be a maximal $S$-region decomposition of $G$. Then at most $170\gamma(G)$ vertices of $G$ do not belong to $R$.

Since any connected dominating set of a graph is also a dominating set, Propositions 3.1 and 3.2 together imply that a planar graph $G$ has a maximal $S$-region decomposition for a connected dominating set $S$ with $O(\gamma_c(G))$ regions covering all but $O(\gamma_c(G))$ vertices of $G$.

### 3.1.2 A Reduce-or-Refine Scheme

In this subsection, we provide a polynomial time algorithm to bound the number of vertices per region by some constant $d$. As long as there exists a region with more than $d$ vertices, this region will either be “refined” into multiple regions or some vertices will be removed from it. We show that in polynomial time the algorithm produces an instance where the number of vertices in each region is bounded by a constant and the total number of regions is $O(k)$.

Lemma 3.2. Let $G$ be a connected plane graph and let $R(v, w)$ be a region between $v, w \in V(G)$ not containing all vertices of $G$. There exists a minimum connected dominating set $S$ of $G$ such that $S$ contains at most two inner vertices from $R(v, w)$.

Proof. Let $S$ be a minimum connected dominating set of $G$ with more than two inner vertices from $R(v, w)$. Let $P_1$ and $P_2$ be the two $[v, w]$-paths forming the boundary of $R(v, w)$. Then either $P_1$ or $P_2$ shares some vertex with $S$, as the graph $G[S]$ contains a path between vertices inside $R(v, w)$ and vertices outside $R(v, w)$. Without loss of generality, let $P_1$ have some vertex in common with $S$. Let $S'$ be the vertex set obtained from $S$ by removing all inner vertices of $R(v, w)$ from it and adding all vertices from $P_1 \setminus S$. Clearly, $|S'| \leq |S|$, since the number of vertices being added is at most three (recall that $P_1$ has at most four vertices including $w$ and $v$), and we remove at least three vertices from $S$. The set $S'$ is also a connected dominating set since every vertex in $R(v, w)$ (including boundary vertices) is adjacent to one of $v$ or $w$, both of which are in $S'$.

Let $N_R(v, w)$ denote the common neighborhood of $v$ and $w$ in the region $R(v, w)$, that is,

$$N_R(v, w) = \{ u \in R(v, w) \mid u \in N(v) \cap N(w) \}.$$

**Case 1:** $N_R(v, w)$ contains at least 106 vertices.

Let $x_1, \ldots, x_\ell$ be a labeling of the vertices in $N_R(v, w)$ such that for all $i = 1, \ldots, \ell - 1$, there is a region $r_i(v, w)$ between $v$ and $w$ with clockwise ordering $(v, x_i, w, x_{i+1})$ of boundary vertices. We define a coloring $c$ on the set \( \{r_1(v, w), \ldots, r_{\ell-1}(v, w)\} \). We color the region $r_i(v, w)$ black, white, black-and-white, or transparent according to the following scheme:
3.1. CONNECTED DOMINATING SET IN PLANAR GRAPHS

Figure 3.1: Regions $r_i = r_i(v, w)$ for $i = 1, 2, 3, 4$. The coloring $c$ colors $r_1$ black-and-white, $r_2$ black, $r_3$ white and $r_4$ transparent.

- black, if $r_i(v, w)$ contains some inner vertices adjacent to $v$ and no inner vertices adjacent to $w$,
- white, if $r_i(v, w)$ contains some inner vertices adjacent to $w$ and no inner vertices adjacent to $v$,
- black-and-white, if $r_i(v, w)$ contains some inner vertices adjacent to $v$ and some inner vertices adjacent to $w$,
- transparent, if $r_i(v, w)$ contains no inner vertices.

Refer to Figure 3.1 for an example. The black (white) weight of $c$ is the number of regions that are colored black or black-and-white (white or black-and-white).

Observation 3.1. Let $G$ be a plane graph and let $R(v, w)$ be a region between $v, w \in V(G)$. If the coloring $c$ has black weight at least 7 then any minimum connected dominating set of $G$ containing at most two inner vertices of $R(v, w)$ contains $v$. Similarly, if the coloring $c$ has white weight at least 7 then any minimum connected dominating set of $G$ containing at most two inner vertices of $R(v, w)$ contains $w$.

Proof. Let $S$ be a minimum connected dominating set of $G$ containing at most two inner vertices of $R(v, w)$, and excluding $v$. Suppose that $c$ has black weight 7 or more. We exhibit a subset that cannot be dominated by $S$, which gives a contradiction. Let $B$ be a set of five regions from $r_2(v, w), \ldots, r_{t-2}(v, w)$ that are colored black or black-and-white. Let $Z(B)$ be a set of vertices containing exactly one vertex $z(r)$ from each region $r \in B$ such that $z(r)$ is a neighbor of $v$ but not a neighbor of $w$. Note that any inner vertex of $R(v, w)$ is adjacent to at most two vertices of $Z(B)$. Thus at most four of the five chosen vertices can be dominated by the inner vertices of $S$. Furthermore, observe that the vertices of $Z(B)$ are not adjacent to any boundary vertices of $R(v, w)$ except $v$. However, $S$ does not contain $v$ and therefore, it follows that at least one vertex in $Z(B)$ is not dominated by $S$. The proof for the case when $c$ has white weight at least 7 is similar. □
Case 1.1: The coloring \( c \) has black weight at least 8 and white weight at least 8.

Let \( S \) be a minimum connected dominating set of \( G \). Note that \( S \) must contain \( v \) and \( w \), by Observation 3.1. Now apply Lemma 3.1 to turn the induced subgraphs \( G[N(v)] \) and \( G[N(w)] \) into independent sets.

Lemma 3.3. (reduce) Let \( G \) be a plane graph and let \( R(v,w) \) be a region between \( v,w \in V(G) \). Suppose that coloring \( c \) has black weight at least 8 and white weight at most 8. Let \( y \) be an inner vertex of a black or black-and-white region such that \( y \) is a neighbor of \( v \) but not a neighbor of \( w \). Then \( \gamma_c(G) = \gamma_c(G - y) \).

Proof. Let \( S \) be a minimum connected dominating set of \( G \). By Observation 3.1, we can choose \( S \) such that \( v,w \in S \). If \( y \in S \) then let \( S' = S \setminus \{y\} \cup \{x_1\} \). Note that \( S' \) has size at most that of \( S \), and using the fact that \( v,w \in S' \) it follows that \( S' \) is a dominating set of \( G \). Moreover, the graph induced by \( S' \) is connected because \( v,w \in S' \) are connected by their common neighbor \( x_1 \in S' \). Thus \( S' \) is a connected dominating set of \( G \) not containing \( y \), and so is a connected dominating set of \( G - y \).

Let \( S' \) be a minimum connected dominating set of \( G' = G - y \). Observe that the coloring \( c' \) of \( G' \) has black weight at least 7 and white weight at least 7. Thus, by Observation 3.1, both \( v,w \in S' \). Hence the vertex \( y \) is dominated in \( G \) by \( v \in S' \), and it follows that \( S' \) is a connected dominating set of \( G \).

An analogous reduction rule applies for inner vertices \( y \) inside some white or black-and-white region.

Case 1.2: The coloring \( c \) has black weight at least 8 and white weight at most 7.

In this case, there exists a region \( r(v,w) \) that is colored black.

Lemma 3.4. (reduce) Let \( G \) be a plane graph and let \( R(v,w) \) be a region between \( v,w \in V(G) \). Suppose that coloring \( c \) has black weight at least 8 and white weight at most 7. Let \( y \) be an inner vertex of a black region such that \( y \) is a neighbor of \( v \) but not a neighbor of \( w \). Then \( \gamma_c(G) = \gamma_c(G - y) \).

Proof. Let \( S \) be a minimum connected dominating set of \( G \). By Observation 3.1, we can choose \( S \) such that \( v \in S \). Notice that each neighbor \( y' \) of \( y \) in \( G \) is also a neighbor of \( v \), and thus any such neighbor \( y' \) is dominated by \( v \). Hence by minimality of \( S \), it holds \( y \notin S \) and thus \( S \) is a connected dominating set of \( G - y \).

Let \( S' \) be a minimum connected dominating set of \( G' = G - y \). The coloring \( c' \) of \( G' \) has black weight at least 7, and thus Observation 3.1 ensures \( v \in S' \). Now \( y \) is a neighbor of \( v \) in \( G' \), and it follows that \( S' \) is a connected dominating set of \( G \).

The case of coloring \( c \) having large white weight and small black weight is similar.

Case 1.3: The coloring \( c \) has black weight at most 7 and white weight at most 7.

Lemma 3.5. (refine) Let \( G \) be a plane graph and let \( R(v,w) \) be a region between \( v,w \in V(G) \) such that \( |N_R(v,w)| \geq 106 \). Suppose that coloring \( c \) has black weight at most 7 and white weight at most 7. Then there exists a region \( R'(v,w) \) such that \( |N_{R'}(v,w)| \geq 8 \) and containing only transparent regions from \( \{r_1(v,w), \ldots, r_{\ell-1}(v,w)\} \).
3.1. CONNECTED DOMINATING SET IN PLANAR GRAPHS

Proof. For all \( i = 1, \ldots, \ell - 1 \), let \( m(i) \) be the largest integer such that \( r(i) = (r_i(v, w), r_{i+1}(v, w), \ldots, r_{i+m(i)}(v, w)) \) is a sequence of consecutive regions that are all colored transparent. Let \( m^* \) be the maximum over all \( m(i) \). Now \( |N_R(v, w)| \geq 106 \) means that there are at least 105 regions of type \( r_i(v, w) \), of which at most 14 are colored either black, white or black-and-white. Hence at least 91 regions are colored transparently; thus \( m^* \geq 7 \). Let \( j \in \{1, \ldots, \ell - 1\} \) be such that \( m(j) = m^* \). Then \( R'(v, w) = r_j(v, w) \cup r_{j+1}(v, w) \cup \ldots \cup r_{j+m^*}(v, w) \) is a region such that \( |N_{R'}(v, w)| \geq 8 \). \( \Box \)

Case 1.4: The coloring \( c \) colors all regions transparent.

Observation 3.2. Let \( G \) be a plane graph and let \( R(v, w) \) be a region between \( v, w \in V(G) \) such that \( |N_R(v, w)| \geq 7 \). If the coloring \( c \) has black weight equal to zero and white weight equal to zero then any minimum connected dominating set of \( G \) containing at most two inner vertices of \( R(v, w) \) contains at least one of \( v \) and \( w \).

Proof. Suppose for contradiction that \( S \) is a minimum connected dominating set of \( G \) containing at most two inner vertices of \( R(v, w) \) and containing neither \( v \) nor \( w \). Notice that for each \( i \in \{2, \ldots, \ell - 1\} \), vertex \( x_i \) can be adjacent to vertices \( x_{i-1} \) and \( x_{i+1} \) but to no other vertices of \( N_R(v, w) \). Also, observe that \( x_2, \ldots, x_{\ell-1} \) are inner vertices of \( R(v, w) \). Since \( G[\{x_1, \ldots, x_{\ell}\}] \) is a subgraph of a path and \( G[S] \) is connected, it follows that if \( x_i \in S \) then either \( x_j \in S \) for all \( j \in \{2, \ldots, i - 1\} \) or \( x_j \in S \) for all \( j \in \{i + 1, \ldots, \ell - 1\} \). Thus, if \( x_4 \in S \) then either \( \{x_2, x_3, x_4\} \subseteq S \) or \( \{x_4, x_5, x_6\} \subseteq S \), a contradiction. Hence either \( x_3 \in S \) or \( x_5 \in S \). If \( x_3 \in S \) then \( x_2 \in S \), since \( x_4 \notin S \). But then \( S \cap \{x_2, \ldots, x_6\} = \{x_2, x_3\} \) and \( x_5 \) is not dominated by \( S \), giving the desired contradiction. Symmetrically, if \( x_5 \in S \) then \( x_6 \in S \) and hence \( x_3 \) is not dominated by \( S \). \( \Box \)

Lemma 3.6. (reduce) Let \( G \) be a plane graph and let \( R(v, w) \) be a region between \( v, w \in V(G) \) such that \( |N_R(v, w)| \geq 8 \). Suppose that the coloring \( c \) has black weight equal to zero and white weight equal to zero. Let \( y \) be an inner vertex of \( R(v, w) \). Then \( \gamma_c(G) = \gamma_c(G - y) \).

Proof. First, by Lemma 3.2 there exists a minimum connected dominating set \( S \) of \( G \) that contains at most two inner vertices of \( R(v, w) \). By Observation 3.2, the set \( S \) contains at least one of \( v \) and \( w \). It then follows by Lemma 3.1 that \( S \) is a connected dominating set of the graph \( G' \) that is obtained from \( G \) by removing all edges between vertices of \( N_R(v, w) \). Since \( R(v, w) \) has at least 6 inner vertices and \( S \) contains at most two of them, there exists an inner vertex \( y' \) of \( R(v, w) \) that is not contained in \( S \). If \( S \) does not contain \( y \) then \( S \) is a connected dominating set of \( G - y \). If \( S \) contains \( y \) then \( S \) \( \backslash \{y\} \cup \{y'\} \) is a connected dominating set of \( G' - y \) and hence also of \( G - y \).

Second, Lemma 3.2 yields that there exists a minimum connected dominating set \( S' \) of \( G' = G - y \) that contains at most two inner vertices of \( R(v, w) \). By Observation 3.2 it follows that \( S' \) contains at least one of \( v \) or \( w \). Thus \( S' \) dominates \( y \) and is a connected dominating set of \( G \). \( \Box \)

We summarize Case 1.
Lemma 3.7. (reduce) There is an algorithm that, given a plane graph \( G \) and a region \( R(v, w) \) between vertices \( v, w \in V(G) \) such that \( |N(v, w)| \geq 106 \), in polynomial time computes a subgraph \( G' \) of \( G \) with fewer vertices than \( G \) such that \( \gamma_c(G') = \gamma_c(G) \).

Proof. The algorithm proceeds as follows. First, it constructs the coloring \( c \) of the regions \( r_1(v, w), \ldots, r_{\ell-1}(v, w) \).

If \( c \) has black weight at least 8 and white weight at least 8 then let \( y \) be an inner vertex of a black or black-and-white region that is a neighbor of \( v \) but not a neighbor of \( w \). Now by Lemma 3.3, \( G' = G - y \) is a subgraph of \( G \) with the desired properties.

If \( c \) has black weight at least 8 and white weight at least 7 then let \( y \) be an inner vertex of a black region and let \( G' = G - y \). By Lemma 3.4 it holds \( \gamma_c(G') = \gamma_c(G) \).

Proceed similarly if \( c \) has black weight at most 7 and white weight at least 8, in which case we let \( G' = G - y' \) for some inner vertex \( y' \) of a white region.

If \( c \) has black weight at most 7 and white weight at most 7 then by Lemma 3.5 there exists a region \( R'(v, w) \) entirely contained in \( R(v, w) \) such that any inner vertex of \( R'(v, w) \) is a common neighbor of \( v \) and \( w \). Thus by Lemma 3.6, letting \( y \) be an inner vertex of \( R'(v, w) \) makes \( G' = G - y \) a subgraph of \( G \) with the desired properties. \( \square \)

Case 2: \( N_R(v, w) \) contains at most 105 vertices.

Lemma 3.8. (refine) Let \( G \) be a plane graph and let \( R(v, w) \) be a region between \( v, w \in V(G) \) such that \( |N_R(v, w)| \leq 105 \). Let \( m \) be the number of vertices in \( R(v, w) \) other than \( v \) and \( w \). Then there is a region \( R'(v, w) \) entirely contained in \( R(v, w) \) that contains at least \( m/104 + 2 \) vertices and such that \( N_{R'}(v, w) = \emptyset \).

Proof. Since \( |N_R(v, w)| \leq 105 \) there are at most 104 regions of type \( r_i(v, w) \). Any region of type \( r_i(v, w) \) has no inner vertices that are adjacent to both \( v \) and \( w \); thus \( N_{r_i}(v, w) = \emptyset \). By the Pigeonhole Principle, one of the at most 104 regions of type \( r_i(v, w) \) contains at least \( m/104 \) vertices other than \( v \) and \( w \). \( \square \)

Case 3: \( N_R(v, w) \) contains no inner vertices of \( R(v, w) \).

Let \( P = \{ P_1, \ldots, P_m \} \) be a maximum-size set of internally vertex-disjoint induced \([v, w]\)-paths of length three entirely contained in \( R(v, w) \), such that for all \( i = 1, \ldots, m - 1 \) the vertices of \( P_i \cup P_{i+1} \) form the boundary of a region \( s_i(v, w) \) not containing vertices from \( P_j \) for any \( j \neq i, i + 1 \). For all \( i = 1, \ldots, m \), let \( v_i \) be the internal vertex of \( P_i \) that is adjacent to \( v \), and let \( w_i \) be the internal vertex of \( P_i \) that is adjacent to \( w \).

Case 3.1: There are at least 12 internally vertex-disjoint \([v, w]\)-paths of length three.

Observation 3.3. Let \( G \) be a plane graph and let \( R(v, w) \) be a region between \( v, w \in V(G) \). If \( |P| \geq 12 \) then every minimum connected dominating set of \( G \) containing at most two inner vertices of \( R(v, w) \) contains both \( v \) and \( w \).

Proof. Observe that \( v \) is the only vertex with any neighbors among \( \{v_3, \ldots, v_9\} \) that is not an inner vertex of \( R(v, w) \). Furthermore, every inner vertex of \( R(v, w) \)
is adjacent to at most 3 vertices among \{v_3, \ldots, v_9\}. Hence if \( S \) is a minimum connected dominating set of \( G \) containing at most two inner vertices of \( R(v, w) \) and not containing \( v \) then \( S \) cannot dominate the set \( \{v_3, \ldots, v_9\} \). Thus \( S \) contains \( v \) and, by a symmetric argument, \( S \) also contains \( w \).

**Lemma 3.9.** (reduce) Let \( G \) be a plane graph and \( R \) be a region between \( v, w \in V(G) \). If \(|P| \geq 12\) then \( \gamma_c(G) = \gamma_c(G - v) \).

**Proof.** First, by Lemma 3.2 there exists a minimum connected dominating set \( S \) of \( G \) that contains at most two inner vertices of \( R(v, w) \). By Observation 3.3, the set \( S \) contains both \( v \) and \( w \). If \( S \) does not contain \( v_3 \) then \( S \) is a connected dominating set of \( G - v \). If \( S \) contains \( v_3 \) and some other inner vertex \( u \) of \( R(v, w) \) then \( v_3 \) is a leaf of \( G[S] \), and hence \( S \setminus \{v_3\} \) is a connected dominating set of \( G - v \). Since every vertex in \( R(v, w) \) is adjacent to either \( v \) or \( w \) and since \( v \) and \( w \) are connected via \( v_3 \) and \( w_2 \) it follows that \( S' = S \setminus \{v_3, u \} \cup \{v_2, w_2\} \) is a connected dominating set of \( G - v \).

Second, Lemma 3.2 implies that there exists a minimum connected dominating set \( S \) of \( G - v_3 \) that contains at most two inner vertices of \( R(v, w) \). By Observation 3.3, the set \( S \) contains both \( v \) and \( w \). Hence \( S \) dominates \( v_3 \) and is a connected dominating set of \( G \).

**Case 3.2:** There are at most 11 internally vertex-disjoint \([v, w]\)-paths of length three.

**Lemma 3.10.** (reduce) Let \( G \) be a plane graph and \( R(v, w) \) be a region between \( v, w \in V(G) \) such that \( N_R(v, w) \) contains no inner vertices of \( R(v, w) \). Let \( m \) be the number of vertices in \( R(v, w) \) other than \( v \) and \( w \). If \(|P| \leq 11\) then there is a region \( R'(v, w) \) entirely contained in \( R(v, w) \) that contains at least \( m/10 + 2 \) vertices such that every \([v, w]\)-path in \( R(v, w) \) contains at least one boundary vertex of \( R(v, w) \) except \( v \) and \( w \).

**Proof.** There are at most 10 regions of type \( s_i(v, w) \). No region of type \( s_i(v, w) \) contains a pair \((v_j, w_j)\) of inner vertices that are internal to some path \( P_j \), for \( j \neq i \). Thus, if \( v' \in N(v) \) is an inner vertex of \( s_i(v, w) \) that is adjacent to \( w_i \) or \( w_{i+1} \) then \( v' \notin \{v_i, v_{i+1}\} \). Symmetrically, if \( w' \in N(w) \) is an inner vertex of \( s_i(v, w) \) that is adjacent to \( v_i \) or \( v_{i+1} \) then \( w' \notin \{w_i, w_{i+1}\} \). Hence by the Pigeonhole Principle, one of the at most 10 regions of type \( s_i(v, w) \) contains at least \( m/10 \) vertices other than \( v \) and \( w \).

**Case 3.3:** There are no internally vertex-disjoint \([v, w]\)-paths of length three.

**Lemma 3.11.** (reduce) Let \( G \) be a plane graph and \( R(v, w) \) be a region between \( v, w \in V(G) \) with at least 1274 vertices such that every \([v, w]\)-path in \( R(v, w) \) contains at least one border vertex of \( R(v, w) \) other than \( v \) and \( w \). There is a polynomial time algorithm that given \( G \) and \( R(v, w) \) computes a subgraph \( G' \) of \( G \) with \(|V(G')| < |V(G)| \) and \( \gamma_c(G') = \gamma_c(G) \).

**Proof.** Since the boundary of \( R(v, w) \) contains at most 6 vertices, either \( v \) or \( w \) has at least 634 neighbors that are inner vertices of \( R(v, w) \). Without loss of generality, assume that \( N(v) \) contains at least 632 inner vertices of \( R(v, w) \). If any vertex \( b \) in \( R(v, w) \) other than \( v \) has at least 106 neighbors in common with \( v \) inside \( R(v, w) \)
then there is a region \( R(b, v) \) such that \( |N_R(b, v)| \geq 106 \). A subgraph \( G' \) of \( G \) with \( |V(G')| < |V(G)| \) and \( \gamma_c(G') = \gamma_c(G) \) is then obtained by applying Lemma 3.7 with \( G \) and \( R(b, v) \).

Suppose now that no such vertex \( b \) exists. Then there is a vertex \( x \in N(v) \) that is an inner vertex of \( R(v, w) \) with no neighbors outside \( N(v) \). We claim that \( \gamma_c(G - x) = \gamma_c(G) \). In one direction let \( S \) be a minimum connected dominating set of \( G \) containing at most 2 inner vertices of \( R(v, w) \). Every boundary vertex of \( R(v, w) \setminus \{v, w\} \) in \( S \) dominates at most 105 inner vertices of \( R(v, w) \) in \( N(v) \). Every inner vertex of \( R(v, w) \) in \( S \) dominates at most 106 inner vertices of \( R(v, w) \) in \( N(v) \), since a vertex dominates itself. Thus, if \( S \) does not contain \( v \) then \( S \) dominates at most \( 4 \cdot 105 + 2 \cdot 106 = 632 < 634 \) vertices of \( N(v) \) that are inner vertices of \( R(v, w) \) yielding a contradiction. Hence, \( S \) contains \( v \). Since \( N(x) \subseteq N(v) \) and \( S \) is a minimum connected dominating set of \( G \), \( S \) does not contain \( x \). Thus, \( S \) is a connected dominating set of \( G - x \).

In the other direction, let \( S \) be a minimum connected dominating set of \( G \) containing at most 2 inner vertices of \( R(v, w) \). By a discussion identical to the one in the previous paragraph, \( S \) must contain \( v \). Hence \( S \) is a connected dominating set of \( G \).

We summarize the case analysis.

**Lemma 3.12.** There is a polynomial time algorithm that, given a plane graph \( G \) and a region \( R(v, w) \) between vertices \( v, w \in V(G) \) with at least 1322672 vertices, computes a subgraph \( G' \) of \( G \) such that \( G' \) has fewer vertices than \( G \) and \( \gamma_c(G') = \gamma_c(G) \).

**Proof.** If \( |N_R(v, w)| \geq 106 \) then Lemma 3.7 applied to \( G \) and \( R(v, w) \) yields the desired subgraph \( G' \). Otherwise, by Lemma 3.8 there exists a region \( R'(v, w) \) with at least 12720 vertices entirely contained in \( R(v, w) \) and such that no common neighbors of \( v \) and \( w \) are inner vertices of \( R'(v, w) \). If \( R'(v, w) \) contains at least 12 internally vertex-disjoint induced \([v, w]-\)paths of length three then by Lemma 3.9 then \( G' = G - v_3 \) is the desired subgraph of \( G \). If \( R'(v, w) \) contains at most 11 internally vertex-disjoint induced \([v, w]-\)paths of length three then by Lemma 3.10 there is a region \( R''(v, w) \) such that \( R''(v, w) \) contains at least 1274 vertices and there is no \([v, w]-\)path such that all its internal vertices are inner vertices of \( R''(v, w) \). In this case Lemma 3.11 implies that a subgraph \( G' \) of \( G \) with \( V(G') < V(G) \) and \( \gamma_c(G') = \gamma_c(G) \) can be computed in polynomial time.

We are now in position to give a proof of our main result.

**Proof of Theorem 3.1:** We prove that \((k)\)-CONNECTED DOMINATING SET on planar graphs admits a kernel of size 3968187 \( k \). We give an algorithm that given an integer \( k \) and a plane graph \( G \) with at least 3968187 \( k \) vertices, in polynomial time either concludes that \( \gamma_c(G) > k \) or computes a subgraph \( G' \) of \( G \) such that \( G' \) has fewer vertices than \( G \) and \( \gamma_c(G') = \gamma_c(G) \). The algorithm proceeds as follows. First, let \( \epsilon = 1/3968186 \) and compute a \((1 + \epsilon)\)-approximation \( S_1 \) of a minimum connected dominating set for \( G \) using the PTAS of Demaine and Hajiaghayi [71]. If \( S_1 \) has strictly more than \((1 + \epsilon)k \) vertices then answer \( \gamma_c(G) > k \) and stop.

Otherwise, compute a maximal \( S_1 \)-region decomposition \( R \) of \( G \) via the algorithm by Guo and Niedermeier [138]. By Proposition 3.1, there are at most \( 3(1 + \epsilon)k \)
regions in \( \mathcal{R} \), and by Proposition 3.2 at most \( 170(1 + \epsilon)k \) vertices do not belong to any region in \( \mathcal{R} \). By the Pigeonhole Principle, there exists a region \( R(v, w) \in \mathcal{R} \) between vertices \( v, w \in V(G) \) that contains at least \( 1322672 \) vertices. Hence by Lemma 3.12, \( G' \) has fewer vertices than \( G \) and \( \gamma_c(G') = \gamma_c(G) \) can be computed in polynomial time.

3.1.3 Extension to Map Graphs

We give a linear kernel for \((k)\)-Connected Dominating Set on map graphs, based on the kernel for planar graphs. Let us remark that map graphs can contain cliques of unbounded size and are thus not closed under taking minors.

The validity of the next lemma relies on the fact that in a witness \( H \) of a map graph \( G \), every vertex of \( V(H) \setminus V(G) \) is adjacent to some vertex of \( V(G) \).

**Lemma 3.13.** Let \( G \) be a map graph and let \( H \) be a witness of \( G \). Then \( G \) has a connected dominating set of size at most \( k \) if and only if \( H \) has a connected dominating set of size at most \( 2k + 1 \).

**Proof.** Let \( H = (V_H^1 \cup V_H^2, E_H) \) with independent sets \( V_H^1 \) and \( V_H^2 \), where \( V_H^1 = V(G) \). Let \( D \) be a connected dominating set of \( G \) with size at most \( k \); then \( D \) is a subset of \( V_H^1 \) of size at most \( k \). Moreover, any vertex in \( V_H^1 \) has distance at most \( 3 - i \) to some vertex of \( D \) in \( H \), for \( i = 1, 2 \). And the distance in \( H \) between any two vertices of \( D \) is at most \( 2 \), since \( G[D] \) is connected. Thus, by adding at most \( k + 1 \) vertices from \( V_H^2 \) to \( D \), we obtain a connected dominating set of \( H \) with at most \( 2k + 1 \) vertices.

Let \( D \) be a connected dominating set of \( H \) with size at most \( 2k + 1 \). Then \( |D \cap V_H^i| \leq k \) for some \( i \in \{1, 2\} \); assume, without loss of generality, \( |D \cap V_H^1| \leq k \). Since \( D \) is both connected and dominating in \( H \), the set \( D \cap V_H^1 \) is a connected dominating set of \( G \) with size at most \( k \). \( \square \)

We combine this lemma with the linear kernel for connected dominating set on planar graphs, and the polynomial-time algorithm by Chen et al. [57] constructing a witness of a map graph in polynomial time, to obtain the following.

**Theorem 3.2.** \((k)\)-Connected Dominating Set has a linear kernel on map graphs.

3.2 Connected Dominating Set in \( K_{3,h} \)-topological minor free Graphs

We generalize the linear kernel for \((k)\)-Connected Dominating Set from planar graphs to \( K_{3,h} \)-topological minor free graphs, for every fixed \( h \geq 1 \). The way to obtain the linear kernel deviates from that for planar graphs; it uses ideas of the linear kernel for \((k)\)-Dominating Set in graphs excluding \( K_{3,h} \) as a topological minor [147]. Throughout the section, \( h \) is fixed and any constants hidden by \( O(\cdot) \) may depend on \( h \).

The linear kernel results from reducing the graph according to a small set of reduction rules, and thereafter exploiting the structure of the such reduced graphs...
which exclude a topological $K_{3,3}$. The reduction rules apply to the neighborhood of single vertices, and to the joint neighborhood of pairs of vertices. For each vertex $v$ of a graph $G$, its open neighborhood $N(v)$ is partitioned into sets $N_1(v), N_2(v)$ and $N_3(v)$, which are defined as

- $N_1(v) = \{ u \in N(v) \mid N(u) \setminus N[v] \neq \emptyset \}$;
- $N_2(v) = \{ u \in N(v) \mid N(u) \cap N_1(v) \neq \emptyset \}$;
- $N_3(v) = N(v) \setminus (N_1(v) \cup N_2(v))$.

Also, for each pair $v,w$ of distinct vertices of a graph their joint neighborhood $N(\{v, w\})$ is partitioned into sets $N_1(v, w), N_2(v, w)$ and $N_3(v, w)$, which are defined as

- $N_1(v, w) = \{ u \in N(\{v, w\}) \mid N(u) \setminus N[\{v, w\}] \neq \emptyset \}$;
- $N_2(v, w) = \{ u \in N(\{v, w\}) \mid N(u) \cap N_1(v, w) \neq \emptyset \}$;
- $N_3(v, w) = N(\{v, w\}) \setminus (N_1(v, w) \cup N_2(v, w))$.

With the neighborhoods partitioned as such, the following proposition is not difficult to prove.

**Proposition 3.3** ([17]). Let $D$ be a dominating set of a graph $G$. If $u \notin N_3(v)$ then there is a path from $u$ to a vertex of $D$ which does not contain $v$ and has length at most 3. If $u \notin N_3(v, w)$ then there is a path from $u$ to a vertex of $D$ which contains neither $v$ nor $w$ and has length at most 3.

Clearly this proposition also holds for connected dominating sets $D$.

We further employ some reduction rules by Gu and Imani [132] for ($k$)-Connected Dominating Set on planar graphs. We rephrase the rules here such that the reduced graph is always a proper subgraph of the input graph $G$. This property bounds the total number of applications of those reduction rules by $O(|V(G)|)$.

**Proposition 3.4** ([132]). Let $G$ be a graph and let $v$ be a vertex of $G$ with $|N_3(v)| > 1$. Let $v' \in N_3(v)$ and let $G'$ be the graph obtained from $G$ by removing all vertices in $N_3(v') \cup N_2(v') \setminus \{v'\}$ and all edges incident to $v'$ except for the edge $\{v, v'\}$. Then $\gamma_c(G) = \gamma_c(G')$.

**Proposition 3.5** ([132]). Let $G$ be a graph and let $v,w$ be adjacent vertices of $G$ such that $|N_3(v, w)| > 2$ and $N_3(v, w)$ cannot be dominated by a single vertex of $N_2(v, w) \cup N_3(v, w)$. Let $G'$ be the graph obtained from $G$ as follows.

- If both $v$ and $w$ dominate $N_3(v, w)$ then let $z \in N_3(v, w)$ and obtain $G'$ by removing all vertices of $N_3(v, w) \cup (N_2(v, w) \cap N(v) \cap N(w))$ and all edges incident to $z$ except for the two edges $\{v, z\}, \{z, w\}$.
- If $v$ dominates $N_3(v, w)$ but $w$ does not dominate $N_3(v, w)$ then let $v' \in N_3(v, w)$ and obtain $G'$ by removing all vertices of $N_3(v, w) \cup (N_2(v, w) \cap N(v))$ and all edges incident to $v'$ except for the edge $\{v, v'\}$. The case that $w$ dominates $N_3(v, w)$ but $v$ does not is treated symmetrically.
3.2. CONNECTED DOMINATING SET IN $K_{3,3}$-TOPOLOGICAL MINOR FREE . . . 77

- If neither $v$ nor $w$ dominates $N_3(v, w)$ then let $v' \in N_2(v, w) \cap N(v), w' \in N_2(v, w) \cap N(w)$ and obtain $G'$ by removing all vertices in $N_2(v, w) \cup N_3(v, w) \setminus \{v', w'\}$ and all edges incident to $v'$ or $w'$ except for the edges $\{v, v'\}, \{w, w'\}$.

Then $\gamma_c(G) = \gamma_c(G')$.

To reduce the joint neighborhood of a pair of vertices at distance 3, some vertices will be classified as “links”. For a graph $G$ and vertices $v, w \in V(G)$ at distance 2, a vertex $x \in N_3(v, w)$ is called a link if it is dominated by a vertex from $N_2(v, w), v$ and $w$, that is $x \in N(N_2(v, w)) \cap N(v) \cap N(w)$. Let $L(v, w)$ denote the set of links for $v$ and $w$.

**Proposition 3.6 ([132]).** Let $G$ be a graph and let $v, w$ be vertices of $G$ at distance 2 such that $|N_3(v, w)| > 3$. Let $G'$ be the graph obtained from $G$ by removing “redundant” vertices from $N_3(v, w)$ but keeping a path between $v$ and $w$, as follows.

- If both $v$ and $w$ dominate $N_3(v, w)$, and $N_3(v, w)$ cannot be dominated by at most two vertices of $N_2(v, w) \cup N_3(v, w)$ then select some vertex of $L(v, w)$ as $p$ (if $L(v, w) \neq \emptyset$) or otherwise select some vertex of $N(v) \cap N(w)$ as $p$. If $|N_3(v, w) \setminus (L(v, w) \cup \{p\})| > 1$ then let $z \in N_3(v, w) \setminus (L(v, w) \cup \{p\})$ and obtain $G'$ by removing all vertices of $N_3(v, w) \setminus (L(v, w) \cup \{p, z\})$ and all edges incident to $z$ except for the two edges $\{v, z\}, \{z, w\}$.

- If $v$ dominates $N_3(v, w)$ but $w$ does not dominate $N_3(v, w)$, and $N_3(v, w)$ cannot be dominated by at most two vertices of $\{v\} \cup N_2(v, w) \cup N_3(v, w)$ then select some vertex of $L(v, w)$ as $p$ (if $L(v, w) \neq \emptyset$) or otherwise select some vertex of $N(v) \cap N(w)$ as $p$. If $|N_3(v, w) \setminus (L(v, w) \cup \{p\})| > 1$ then let $v' \in N_3(v, w) \setminus (L(v, w) \cup \{p\})$ and obtain $G'$ by removing all vertices of $N_3(v, w) \setminus (L(v, w) \cup \{p, v'\})$ and all edges incident to $v'$ except for the edge $\{v, v'\}$. The case that $w$ dominates $N_3(v, w)$ but $v$ does not dominate $N_3(v, w)$, and $N_3(v, w)$ cannot be dominated by at most two vertices of $\{v\} \cup N_2(v, w) \cup N_3(v, w)$ is treated symmetrically.

- If neither $v$ nor $w$ dominates $N_3(v, w)$, and $N_3(v, w)$ cannot be dominated by at most two vertices of $\{v\} \cup N_2(v, w) \cup N_3(v, w)$ or by at most two vertices of $\{w\} \cup N_2(v, w) \cup N_3(v, w)$ then select some vertex of $N(v) \cap N(w)$ as $p$. If $|N_2(v, w) \cup N_3(v, w) \setminus \{p\}| > 2$ then let $v' \in ((N_2(v, w) \cup N_3(v, w)) \setminus \{p\}) \cap N(v), w' \in ((N_2(v, w) \cup N_3(v, w)) \setminus \{p\}) \cap N(w)$ and obtain $G'$ by removing all vertices of $(N_2(v, w) \cup N_3(v, w)) \setminus \{p, v', w'\}$ and all edges incident to $v'$ or $w'$ except for the two edges $\{v, v'\}, \{w, w'\}$.

Then $\gamma_c(G) = \gamma_c(G')$.

The next reduction rule applies to a pair of vertices at distance 3, and it requires the notion of a “key-neighbor”. For a graph $G$ and vertices $v, w \in V(G)$ at distance 3, a vertex $y \in N_2(v, w) \cap N(w)$ is called a key-neighbor of $w$ if $y$ is dominated by a vertex $z \in N_3(v, w) \cap N(v)$, that is $y \in N(w) \cap N_2(v, w) \cap N_3(v, w) \cap N(v)$. The vertex $z$ is called a companion of the key-neighbor $y$.

**Proposition 3.7 ([132]).** Let $G$ be a graph and let $v, w$ be vertices of $G$ at distance 3 such that $N_3(v, w) \geq 4$. Let $G'$ be the graph obtained from $G$ by removing “redundant” vertices from $N_3(v, w)$ but keeping a path between $v$ and $w$, as follows.
• If $v$ dominates $N_3(v, w)$ but $N_3(v, w)$ cannot be dominated by at most three vertices from $\{v\} \cup N_2(v, w) \cup N_3(v, w)$ then let $Z$ be a minimum subset of companions such that $Z$ dominates every key-neighbor of $w$. If $Z \neq \emptyset$ then select some $z \in Z$ as $p$ and a key-neighbor $y$ of $w$ dominated by $z$ as $q$, otherwise select any two vertices $p$ and $q$ such that $(v, p, q, w)$ is a path of $G$. If $N_3(v, w) \setminus (Z \cup \{p, q\}) \neq \emptyset$ then let $v' \in N_3(v, w) \cap N(v)$ and obtain $G'$ by removing all vertices from $N_3(v, w) \setminus (Z \cup \{p, q, v'\})$ and all edges incident to $v'$ except the edge $\{v, v'\}$. The case that $w$ dominates $N_3(v, w)$ but $N_3(v, w)$ cannot be dominated by at most three vertices from $\{v\} \cup N_2(v, w) \cup N_3(v, w)$ is treated symmetrically.

• If neither $v$ nor $w$ dominates $N_3(v, w)$, and $N_3(v, w)$ can neither be dominated by at most three vertices of $\{v\} \cup N_2(v, w) \cup N_3(v, w)$ nor at most three vertices of $\{w\} \cup N_2(v, w) \cup N_3(v, w)$ then let $p, q$ be distinct vertices such that $(v, p, q, w)$ is a path in $G$. Let $v' \in (N_3(v, w) \cap N(v)) \setminus N(w)$, $w' \in (N_3(v, w) \cap N(w)) \setminus N(v)$ and obtain $G'$ by removing all vertices of $(N_2(v, w) \cup N_3(v, w)) \setminus \{p, q, v', w'\}$ and all edges incident to $v'$ and $w'$ except the two edges $\{v, v'\}, \{w, w'\}$.

Then $\gamma_c(G) = \gamma_c(G')$.

Gu and Imani [132] showed that for planar graphs $G$ with $n$ vertices, the graph $G'$ of Lemmas 3.4–3.7 can be constructed in time $O(n^4)$. Their proof shows that for general graphs this time is bounded by $O(n^5)$. Observe that any application of Lemmas 3.4–3.7 to a graph $G$ returns a proper subgraph $G'$ of $G$ with fewer vertices.

A graph $G$ to which none of Lemmas 3.4–3.7 applies is called reduced. We now show that a reduced graph $G$ without $K_{3,3}$ as a topological minor and connected dominating set of size $k$ has $O(k)$ vertices. To this end, observe that in a reduced graph the set $N_3(v, w)$ is dominated by at most three vertices from $N_2(v, w) \cup N_3(v, w)$, for any pair of vertices $v, w \in V(G)$.

**Definition 3.3** ([17]). Let $G$ be a graph and let $D$ be a dominating set of $G$. Denote by $\hat{D}$ the set of vertices in $V \setminus D$ that are adjacent to at least two vertices in $D$. For distinct vertices $d_1, d_2 \in D$ denote by $\text{Inner}(d_1, d_2)$ the set of all inner vertices of paths of length 3 of the type $(d_1, x, y, d_2)$, such that $x, y \in N_3(d_1, d_2) \setminus (D \cup D_{3,3} \cup D)$. Denote $\text{Inner}(D) = \bigcup_{d_1, d_2 \in \hat{D}, d_1 \neq d_2} \text{Inner}(d_1, d_2)$.

We further use that $K_{3,3}$-topological minor-free graphs are $d$-degenerate, for some $d = d(h)$.

**Proposition 3.8** ([41, 177]). There exists a constant $c$ such that for every $h \geq 1$, every $K_h$-topological minor free graph is $ch^2$-degenerate.

Alon and Gutner [17] introduced the following notation for certain vertex-disjoint paths. For a graph $G$, a set $D \subseteq V(G)$, and integers $n, \ell$ define $\hat{D}_{n, \ell}$ as the set of vertices $v \in V(G) \setminus D$ for which there are $n$ vertex-disjoint paths of length at most $\ell$ from $v$ to $n$ different vertices of $D$. That is, $v$ is the starting vertex of all the paths, but any other vertex belongs to at most one of the paths.

**Proposition 3.9** ([17]). Let $G$ be a graph excluding $K_{3,3}$ as a topological minor and let $D \subseteq V(G)$. Then any $\ell \geq 1$ satisfies $|\hat{D}_{3,\ell}| \leq (3c\ell)^{18c^2} |D|$, where $c$ is the constant from Proposition 3.8.
3.2. CONNECTED DOMINATING SET IN $K_{3,3}$-TOPOLOGICAL MINOR FREE . . .

**Lemma 3.14.** Let $G$ be a reduced graph without $K_{3,3}$ as a topological minor and let $D$ be a dominating set of $G$. Then $|\bar{D}| \leq [((9ch)^{162} + c(3 + h)^2(3h + 2))]|D|$, where $c$ is the constant from Proposition 3.8.

**Proof.** Let $v \in \bar{D}$. This means that $v$ is adjacent to at least 2 vertices of $D$, so we distinguish between three cases.

- The vertex $v$ is adjacent to at least 3 vertices of $D$. Thus, by definition, $v \in \tilde{D}_{3,3}$, and it follows from Proposition 3.9 that $|\tilde{D}_{3,3}| \leq (9ch)^{162}|D|$.

- The vertex $v$ is adjacent to exactly two vertices $d_1, d_2 \in D$ and $v \notin N_3(d_1, d_2)$. It follows from Proposition 3.3 that there is a path of length at most 3 from $v$ to a vertex of $D$, and the path does not use the vertices $d_1$ and $d_2$. This implies that $v \in \tilde{D}_{3,3}$ and we proceed as in the previous case.

- The vertex $v$ is adjacent to exactly two vertices $d_1, d_2 \in D$ and $v \in N_3(d_1, d_2)$. The number of pairs $d_1, d_2 \in D$ for which there is a vertex $v \notin D$ such that $N(v) \cap D = \{d_1, d_2\}$ is bounded by $c(3 + h)^2|D|$. To see this, just connect each such pair $d_1, d_2$, in case they were not connected before. Denote the resulting graph by $G'$. The number of edges in $G'|D|$ is at least the number of pairs we are counting. Since $G'|D|$ does not contain $K_{3,3}$ as a topological minor, it has at most $c(3 + h)^2k$ edges, where $c$ is the constant from Proposition 3.8.

It is now enough to prove that there are at most $3h + 2$ vertices $w \in |N_3(d_1, d_2)|$ that are adjacent to both $d_1$ and $d_2$, for every pair of distinct vertices $d_1, d_2 \in D$. For contradiction, assume that there are more than $3h + 2 \geq 4$ such vertices $w$. Then $d_1$ and $d_2$ are at most distance 3 apart, and hence $N_3(v, v)$ is dominated by a set $V' \subseteq N_2(d_1, d_2) \cup N_3(d_1, d_2)$ of at most three vertices, since $G$ is reduced. Note that any vertex $v \in V'$ can possibly belong to $N_3(d_1, d_2)$. This implies that $d_1, d_2$, and $V'$ together with $N_3(d_1, d_2) \setminus V'$ contain a topological $K_{3,3}$, a contradiction. $\square$

**Corollary 3.1.** Let $G$ be a reduced graph excluding $K_{3,3}$ as a topological minor and let $D$ be a dominating set of $G$. Then every subset $U \subseteq V(G) \setminus D$ satisfies $|N[U]| \leq |U| + O(|D|)$.

**Proof.** Clearly, the set $D \cup U$ is a dominating set of $G$. A vertex $v \in N[U] \setminus (D \cup U)$ is adjacent to a vertex of $U$ and also adjacent to a vertex of $D$, since $D$ is a dominating set. This means that $v$ is adjacent to at least two vertices of $D \cup U$. The result now follows from Lemma 3.14. $\square$

**Lemma 3.15.** Let $G$ be a graph without $K_{3,3}$ as a topological minor and let $D$ be a dominating set of $G$. Then there are at most $c(3 + h)^2|D|$ pairs $d_1, d_2 \in D$ for which Inner$(d_1, d_2) \neq \emptyset$.

**Proof.** Consider the pairs $d_1, d_2 \in D$ for which Inner$(d_1, d_2) \neq \emptyset$ in some arbitrary order. For each such pair $d_1, d_2$, there are two vertices $x, y \in N_3(d_1, d_2) \setminus (D \cup \tilde{D}_{3,3} \cup \bar{D})$ that appear on the path $(d_1, x, y, d_2)$. We claim that both $x$ and $y$ do not belong to any other pair Inner$(d'_1, d'_2)$. To see this, suppose for contradiction that $x \in \text{Inner}(d_1, d_2) \cap \text{Inner}(d'_1, d'_2)$ for $\{d_1, d_2\} \neq \{d'_1, d'_2\}$. Since $x \notin \bar{D}$, it has only one neighbor in $D$, so assume, without loss of generality, that $x$ is adjacent to $d_1 = d'_1$.
and $x$ appears on the two paths $(d_1, x, y, d_2)$ and $(d_1, x, z, d'_2)$. This implies that $x \in \hat{D}_{3,3}$, a contradiction, and the claim is proved.

In each case as above, delete the vertices $x$ and $y$, and add an edge between $d_1$ and $d_2$, assuming this edge does not exist. Denote the resulting graph by $G'$. Obviously, $G'[D]$ does not contain $K_{3,h}$ as a topological minor and therefore has at most $c(3 + h)^2 |D|$ edges. The number of edges in the induced subgraph $G'[D]$ is at least the number of pairs for which $\text{Inner}(d_1, d_2) \neq \emptyset$, as claimed.

\textbf{Lemma 3.16.} Let $G$ be a reduced graph without $K_{3,h}$ as a topological minor and let $D$ be a connected dominating set of $G$. Then every two distinct vertices $d_1, d_2 \in D$ satisfy $|\text{Inner}(d_1, d_2)| \leq 18h^2$.

\textit{Proof.} For contradiction, assume that $|\text{Inner}(d_1, d_2)| \geq 18h^2 + 1$. This implies that $|N_3(d_1, d_2)| \geq 2$ and hence $d_1$ and $d_2$ are at most distance 3 apart. Since the graph is reduced, there is a set $V' \subseteq N_2(d_1, d_2) \cup N_3(d_1, d_2)$ of at most three vertices that dominates $N_3(d_1, d_2)$. Let $q$ be the maximum number of internally vertex-disjoint paths of type $(d_1, x, y, d_2)$, such that $x, y \in \text{Inner}(d_1, d_2)$, and denote by $W$ the set of the $2q$ inner vertices of these paths. Note that any vertex $v \in V'$ can possibly belong to $W$. We must have that $q \leq 3h$, since otherwise $d_1, d_2,$ and $V'$ would be part of a graph with a topological minor $K_{3,h}$. Since $|W| = 2q \leq 6h$, there are at least $6h(3h - 1) + 1$ vertices of $\text{Inner}(d_1, d_2) \setminus W$ that appear on a path of the type $(d_1, x, y, d_2)$ together with one of the vertices of $W$. Thus, there is a vertex $w \in W$ that belongs to at least $h$ of these paths. Assuming, without loss of generality, that $w$ is adjacent to $d_1$, there are $3h + 1$ different paths of length 2 from $w$ to $d_2$, and the inner vertices of these paths are from $\text{Inner}(d_1, d_2)$. Thus, $w, d_2$ and $V'$ are part of a graph with a topological minor $K_{3,h}$, a contradiction.

\textbf{Corollary 3.2.} Let $G$ be a reduced graph without $K_{3,h}$ as topological minor and let $D$ be a connected dominating set of $G$. Then $|\text{Inner}(D)| = O(|D|)$.

\textbf{Lemma 3.17.} Let $G$ be a reduced graph without $K_{3,h}$ as topological minor and let $D$ be a connected dominating set of $G$. Then the number of vertices that appear on a path of length 3 between two vertices of $D$ is $O(|D|)$.

\textit{Proof.} We examine the inner vertices of paths of the form $(d_1, v, x, d_2)$, such that $d_1, d_2 \in D$. It follows from Proposition 3.9 and Lemma 3.14 that $|\hat{D}_{3,3} \cup \hat{D}| = O(|D|)$, which means that it remains to count the number of vertices not in $D \cup \hat{D}_{3,3} \cup \hat{D}$. Assume that $v \notin D \cup \hat{D}_{3,3} \cup \hat{D}$. Since $v \notin D$, it is adjacent to exactly one vertex of $D$, and therefore $x \notin D$. If $x \in \hat{D}_{3,3} \cup \hat{D}$ then $v \in N[\hat{D}_{3,3} \cup \hat{D}]$, but it follows from Corollary 3.1 that $|N[\hat{D}_{3,3} \cup \hat{D}]| = O(|D|)$. If either $v$ or $x$ do not belong to $N_3(d_1, d_2)$ then this implies that $x \in \hat{D}_{3,3}$, but this case has already been addressed. The only remaining case is that $v, x \in N_3(d_1, d_2) \setminus (D \cup \hat{D}_{3,3} \cup \hat{D})$, which means that $v \in \text{Inner}(D)$, and we know from Corollary 3.2 that $|\text{Inner}(D)| = O(|D|)$.

We can now state the main result of this section.

\textbf{Theorem 3.3.} \((k)-\text{Connected Dominating Set has a linear kernel on } K_{3,h}\)-topological minor free graphs.
Proof. Suppose that \( G \) contains no \( K_{3,h} \) as topological minor and \( \gamma_c(G) = k \). As long as the conditions of Lemmas 3.4–3.7 are satisfied, apply these lemmas to get a reduced subgraph \( G' \). We know that \( \gamma_c(G') = k \), so let \( D \) be a connected dominating set of \( G' \) of size \( k \). It follows from Lemma 3.14 that \( |D| = O(k) \), so we need to count the number of vertices in \( D \cup \hat{D} \). Assume that \( v \notin D \cup \hat{D} \) is adjacent to \( d_1 \in D \). If \( v \in N_3(d_1) \) then in a reduced graph \( |N_3(d_1)| \leq 1 \), which means that there could be at most \( k \) vertices of this type. Assume now that \( v \notin N_3(d_1) \), so by Proposition 3.3 there is a path of length at most 4 from \( v \) to a vertex \( d_2 \in D \), and \( d_1 \) is not part of this path. We examine a shortest path \( P = (d_1, v, u_1, \ldots, u_{\ell}, d_2) \) from \( d_1 \) to \( d_2 \), for some vertices \( u_1, \ldots, u_{\ell} \) with \( \ell \leq 3 \). Since \( v \notin \hat{D} \), it is adjacent to only one vertex of \( D \), so the path \( P \) has length either 3 or 4.

If \( P \) is of length 3 then it follows from Lemma 3.17 that there are at most \( O(k) \) vertices of this type. If \( P \) is of length 4, denote it by \((d_1, v, x, y, d_2)\), where \( x, y \notin D \). Vertex \( x \) is adjacent to some vertex of \( D \). It cannot be adjacent to \( d_2 \), since a path \( P \) of minimum length was chosen. If \( x \) is adjacent to a vertex of \( D \setminus \{d_1, d_2\} \) then \( x \in D_{3,3} \) and \( v \in N[D_{3,3}] \), but it follows from Corollary 3.1 that \( |N[D_{3,3}]| = O(k) \). The remaining case is that \( d_1 \) is the only vertex in \( D \) adjacent to \( x \). Since \( x \notin D \) is on a path of length 3 from \( d_1 \) to \( d_2 \), it follows from Lemma 3.17 and Corollary 3.1 that the number of vertices \( v \) of this type is also \( O(k) \).

Since the graph \( K_{3,h} \) cannot be embedded in a surface of genus less than \( h - 3 \), the following holds.

**Corollary 3.3.** \((k)-Connected Dominating Set has a linear kernel in graphs of bounded genus.

### 3.3 Kernelization Lower Bounds

The kernels obtained in the previous section have size polynomial in \( k \) and exponential in \( |H| \). In this section we show that on \( H \)-minor free graphs, \textsc{Connected Dominating Set} parameterized by solution size \( k \) and \( |H| \) does not admit a kernel of polynomial size, modulo a collapse of the polynomial hierarchy to its third level. We will show this to hold even if \( H \) is planar, bipartite and of maximum degree three. The result is a consequence of a polynomial-parameter transformation from the \textsc{Colored Small Universe Hitting Set} (Col-SUHS) problem, that was introduced by Dom et al. [78]. The input to \textsc{Col-SUHS} is a set family \( \mathcal{F} \) over a universe \( U \) with \( |U| \leq d \) and a positive integer \( k \), and the elements of \( U \) are colored with colors from the set \( \{1, \ldots, k\} \), and the question to answer is whether there is a subset \( S \subseteq U \) containing exactly one element of each color such that every set in \( \mathcal{F} \) has non-empty intersection with \( S \). Such a set \( S \) is said to be a \textit{colorful hitting set} for \( \mathcal{F} \). Our reduction is justified by the following theorem of Bodlaender et al. [34].

**Proposition 3.10** ([34]). Let \( \Pi, \Pi' \) be parameterized problems such that \( \Pi' \) is \( \text{NP} \)-complete and \( \Pi \) is in \( \text{NP} \). If there is a polynomial parameter transformation from \( \Pi \) to \( \Pi' \) and \( \Pi' \) has a polynomial kernel then \( \Pi \) also has a polynomial kernel.
Theorem 3.4. Even for planar bipartite graphs $H$ of maximum degree three, on $H$-minor free graphs the Connected Dominating Set problem parameterized by solution size and $|H|$ does not admit a polynomial kernel, unless $\text{PH} = \Sigma^p_3$.

Proof. The unparameterized version Connected Dominating Set is well-known to be $\text{NP}$-complete on planar graphs [121], so in particular the problem is $\text{NP}$-complete on $H$-minor free graphs. We give a polynomial parameter transformation from $\text{Col-SUHS}$ for parameters the size $k$ of a minimum colorful hitting set and the size $d$ of the universe. This problem does not admit a polynomial kernel assuming that $\text{PH} \neq \Sigma^p_3$ [78], and its unparameterized version clearly belongs to $\text{NP}$. From an instance $(\mathcal{F}, U; k, d)$, we construct an instance $(G; k+1, d+1)$ as follows. The graph $G$ is constructed by

(1) starting from the bipartite element-set incidence graph of $(\mathcal{F}, U)$,

(2) adding one vertex $v_i$ for each color class $i$ and connecting $v_i$ by edges to every vertex corresponding to an element of $U$ with color $i$,

(3) finally adding two new vertices $y, z$ and connecting $y$ by edges to $z$ and all vertices corresponding to elements in $U$.

We claim that there is a colorful hitting set for $\mathcal{F}$ of size $k$ if and only if there is a connected dominating set for $G$ of size $k+1$.

First, suppose that $S$ is a size-$k$ colorful hitting set for $\mathcal{F}$. Then the set of vertices in $G$ corresponding to $S$, augmented by $y$, is a connected dominating set for $G$ of size $k+1$.

Conversely, if $D$ is a minimum connected dominating set for $G$ of size $k+1$ then we can assume, without loss of generality, that $z \notin D$ and $y \in D$. Further, for every color class $i$ there is at least one vertex $u \in D$ that is colored by $i$, to dominate vertex $v_i$. But then since there are precisely $k$ color classes, by $|D \setminus \{y\}| = k$ for every color class $i$ there is at most one vertex $u \in D$ that is colored by $i$. It follows that $D \setminus \{y\} \subseteq U$. Now since $D \setminus \{y\}$ is a dominating set of the bipartite element-set incidence subgraph of $G$, the elements of $U$ corresponding to vertices in $D \setminus \{y\}$ form a colorful hitting set of $\mathcal{F}$.

Moreover, the set $U \cup \{y\}$ is a size-$(d+1)$ vertex cover for $G$.

Returning to graphs excluding a fixed minor $H$, notice that graphs with vertex cover of size at most $(d+1)$ exclude the $\left(\left\lceil \frac{1}{(d+1)\cdot 3} \right\rceil + 1 \right) \times \left(\left\lceil \frac{1}{(d+1)\cdot 3} \right\rceil + 1 \right)$ grid as minor. Such graphs exclude the $\frac{1}{(d+1)\cdot 3}$ grid as minor, which is planar, bipartite and of maximum degree three.

3.4 Concluding Remarks

We showed that $(k)$-Connected Dominating Set in planar graphs admits a linear kernel. Our result was the first linear kernel for a “connectivity problem” on planar graphs, that could not be obtained from the framework of Guo and Niedermeier [138]. Later, Bodlaender et al. [36] proved a “meta-kernelization” which gives an existential proof for a linear kernel for $(k)$-Connected Dominating Set on planar graphs, but the constant in the kernel size was left unspecified. Very recently,
a linear kernel for \((k)\)-CONNECTED DOMINATING SET on \(K_{3,h}\)-minor free graphs was obtained as a corollary by general kernelization results of Fomin et al. [112] for apex-minor free graphs.

For planar graphs, our kernelization is impractical for two reasons. The first reason is the huge constant in the kernel size, and the second reason is the choice of \(\epsilon = 1/3968186\) in the PTAS for \((k)\)-CONNECTED DOMINATING SET that yields an unmanageable running time. We think that both these problems can be remedied; choosing \(\epsilon = 1\) yields a 2-approximation for \((k)\)-CONNECTED DOMINATING SET in planar graphs that runs quite quickly, at the cost of a factor 2 in the kernel size. Also, the constant in our kernelization can be improved significantly. We focused only on showing the existence of a linear kernel and in many places we deliberately picked a proof that yielded a higher constant but was more readable and understandable. A possible way to attack this problem would be to eliminate the “refine” steps and to re-analyze the cases, taking into account the noise that the “refine” steps removed. That a constant below 1000 is achievable was recently shown by Gu and Imani [132]. Finally, it would be interesting to see whether the reduce or refine technique could be applied to achieve this, or to give kernels for other problems.

In this chapter we have shown the existence of polynomial, even linear, kernels for \((k)\)-CONNECTED DOMINATING SET for certain minor-closed graph classes as well as for map graphs, which are not minor-closed. For \((k)\)-CONNECTED DOMINATING SET on apex-minor free graphs, Fomin et al. [112] give an existential proof for a polynomial kernel. We ask whether there is a polynomial kernel for \((k)\)-CONNECTED DOMINATING SET on \(H\)-minor free graphs for general graphs \(H\).
In this chapter we prove lower and upper bounds on the maximum number of minimal feedback vertex sets in tournaments, and give an algorithm for their fast enumeration in polynomial space. As a consequence, we obtain the fastest known algorithm for finding a feedback vertex set of minimum size in tournaments. We further show that fast algorithms for their enumeration in certain orders are unlikely to exist. We then present an algorithm for finding a minimum feedback vertex set in bipartite directed graphs. Finally, we establish lower and upper bounds on the sizes of induced matchings in planar graphs of maximum degree three, resulting in a linear kernel for the \((k)\)-\textsc{Induced Matching} problem in planar graphs of maximum degree three. The lower bounds are proven using a discharging procedure, a method that was developed to establish the famous Four Color Theorem \cite{24} and that has since found widespread use in proving properties of planar graphs. The rough idea of this method is as follows. Each vertex and face of \(G\) is assigned an initial “charge”, such that the sum of all charges in \(G\) is negative. We then apply certain redistribution rules (the discharging procedure) for exchanging charge between the vertices and faces. These redistribution rules leave the total sum of charges invariant. We then prove that if none of 12 special configurations occurs in \(G\) then each vertex and each face will have non-negative charge after the discharging procedure has finished. This of course contradicts the fact that the total sum of the charges is negative. Hence we must have at least one of the special configurations, for each of which can inductively prove the existence of a large enough induced matching.

Given an undirected graph \(G\), the minimal feedback vertex sets of \(G\) can be enumerated with polynomial delay due to an algorithm by Schwikowski and Speckenmeyer \cite{227}. Equivalently, the algorithm enumerates the maximal acyclic vertex sets of \(G\), by creating and traversing a strong directed graph that contains exactly one vertex for each maximal acyclic vertex set of \(G\) and that contains no other vertices. There is an arc from a vertex \(u\) to a vertex \(v\) if the minimal feedback vertex set corresponding to \(u\) can be transformed into the minimal feedback vertex set corresponding to \(v\), by local operations. Modifications of the algorithm work on directed graphs, where they enumerate the maximal acyclic vertex sets and maximal acyclic arc sets with polynomial delay. The algorithm has the shortcomings of taking space that is within a polynomial factor of the number maximal acyclic vertex sets (and
thus potentially takes space that is exponential in $|V(G)|$, and it cannot guarantee that the maximal acyclic vertex sets are output in a desired order. We address both issues for undirected and directed graphs.

Let us first ask for an algorithm that enumerates the maximal acyclic vertex sets of a (directed or undirected) graph $G$ with polynomial delay and in a specified order. A natural order to consider is the “lexicographic order” defined on the power set of $V(G) = \{v_1, \ldots, v_n\}$. For vertex sets $X \subseteq V(G)$, we write $\chi_X(i) = 1$ if $v_i \in X$ and $\chi_X(i) = 0$ otherwise. Let $<$ denote the total order on $V(G)$ induced by the labels of the vertices. For vertex sets $X, Y \subseteq V(G)$, say that $X$ is lexicographically smaller than $Y$ and write $X < Y$ if for the minimum index $i$ for which $\chi_X(i) \neq \chi_Y(i)$ it holds that $v_i \in X$. If $X < Y$ then say that $Y$ is lexicographically larger than $X$. Because $X$ and $Y$ are totally ordered by the restriction of $<$ to $X$ and $Y$, respectively, $<$ is also a total order and each collection of subsets of $V$ has a unique lexicographically smallest and a unique lexicographically largest element. Thus, the inverse of $<$ exists and is called the reverse lexicographic order of $V(G)$.

Clearly, by enumerating all vertex sets $V' \subseteq V(G)$ in (reverse) lexicographic order and outputting $V'$ if and only if $V'$ is a maximal acyclic vertex set of $G$, we obtain an algorithm enumerating the minimal feedback vertex sets of $G$ in (reverse) lexicographic order. But this procedure cannot be guaranteed to run with polynomial delay. In fact, we establish the following hardness result for undirected graphs.

**Theorem 4.1.** Given a graph $G$ and a maximal acyclic vertex set $S$ of $G$, it is coNP-complete to decide whether $S$ is the lexicographically largest maximal acyclic vertex set of $G$.

**Proof.** The problem is in coNP since if $S$ is not the lexicographically largest, a short proof of this fact can be obtained by exhibiting a maximal acyclic vertex set that is lexicographically larger than $S$.

To show completeness, we give a polynomial-time transformation from Unsatifiability. Let $F$ be a Boolean formula in conjunctive normal form, with clause set $C$ and variables $x_1, \ldots, x_n$. Let $x^+_i, x^-_i$ denote the unnegated and negated literal of variable $x_i$, for $i = 1, \ldots, n$. We assume that no clause $C \in C$ contains both the negated and unnegated literal of some variable, for if there were such a clause $C$ then $F$ is satisfiable if and only if $F - C$ is. We shall construct a graph $G$ with ordered vertices and a maximal acyclic vertex set $S$ of $G$ such that there exists a maximal acyclic vertex set of $G$ lexicographically larger than $S$ if and only if there is a satisfying truth assignment for $F$. Graph $G$ has one vertex for each clause, one vertex for each variable $x_i$, two vertices $x^+_i, x^-_i$ for the literals of each variable $x_i$, and a special vertex $a$. The order of the vertices is as follows: first the clauses, then $a$, then the variables, then the literals in any order. The edges of $G$ are as follows. Vertex $a$ is adjacent to all other vertices. Each variable vertex $x_i$ is connected to both its literal vertices $x^+_i, x^-_i$, and to each clause containing one of its literals. There is an edge between each pair $x^+_i, x^-_i$ of contradicting literals. Finally, each clause is adjacent to all literals it contains. These are all edges of $G$. Let $S = \{a, x_1, \ldots, x_n\}$, so $S$ is a maximal acyclic vertex set of $G$.

Is there a maximal acyclic vertex set lexicographically larger than $S$? If there is then let $S^*$ be the lexicographically largest maximal acyclic vertex set with $S < S^*$. Then $S^*$ does not contain any clause vertex. Consider the following cases.

- If $a \in S^*$ and $x_i \notin S^*$ for all $i = 1, \ldots, n$ then $S^* \setminus \{a\}$ is a subset of the literals.
Moreover, it contains at most one of $x_i^+, x_i^-$ for all $i = 1, \ldots, n$, to be acyclic. For $S^*$ to be maximal, for each clause there is at least one literal in $S^*$ which is adjacent to it. In other words, $S^* \setminus \{a\}$ is a satisfying truth assignment for $F$.

- If $a \in S^*$ and $x_i \in S^*$ for some $i \in \{1, \ldots, n\}$ then $x_i^+, x_i^- \notin S^*$ and there are acyclic vertex sets $S^+ = S^* \setminus \{x_i\} \cup \{x_i^+\}$ and $S^- = S^* \setminus \{x_i\} \cup \{x_i^-\}$. Moreover, $S^* \prec S^+$ and $S^* \prec S^-$, and one of $S^+, S^-$ is maximal: this contradicts the choice of $S^*$.

- If $a \notin S^*$ then let $I \subseteq \{1, \ldots, n\}$ be the set of indices $i$ for which $x_i \notin S^*$. For any $i \in I$, at least one of $x_i^+, x_i^-$ belongs to $S^*$, for $S^*$ to be maximal. Let $C(i)$ be the subset of clause vertices adjacent to $x_i$. By maximality of $S^*$, for any vertex $C \in C(i)$ there is a vertex $x_j$ with $j \notin I$ that is adjacent to $C$. For any $j \notin I$, exactly one of $x_j^+, x_j^-$ belongs to $S^*$, for $S^*$ to be acyclic and maximal.

Thus, if $I' \subseteq I$ is the set of indices for which both $x_i^+, x_i^-$ belong to $S^*$, the set $S^* \setminus (\{x_1, \ldots, x_n\} \cup \{x_i^+ \mid i \in I'\})$ is a satisfying truth assignment for $F$.

Therefore, $S$ is the lexicographically largest maximal acyclic vertex set of $G$ if and only if the given Boolean formula is unsatisfiable. □

**Corollary 4.1.** Given a graph $G$ and a maximal acyclic subset $S$ of $G$, it is NP-hard to generate the lexicographically next maximal acyclic subset of $G$.

**Corollary 4.2.** Unless $P = NP$, there is no algorithm enumerating the maximal acyclic vertex sets of a graph in reverse lexicographic order and with polynomial delay.

For general directed graphs $G$, a simple reduction from a theorem by Johnson et al. [165] shows that the maximal acyclic vertex sets of $G$ cannot be enumerated with polynomial delay and in reverse lexicographic order, unless $P = NP$. We show that even for the structurally rich class of tournaments, such an algorithm is unlikely to exist.

**Theorem 4.2.** Given a tournament $T$ and a maximal acyclic vertex set $S$ of $T$, it is coNP-complete to decide whether $S$ is the lexicographically largest maximal acyclic vertex set of $T$.

**Proof.** The problem is in coNP since if $S$ is not the lexicographically largest, a short proof of this fact can be obtained by exhibiting a maximal acyclic vertex set that is lexicographically larger than $S$.

To show completeness, we give a polynomial-time transformation from Unsatisfiability. Let $F$ be a Boolean formula in conjunctive normal form, with clauses $C_1, \ldots, C_m$ and variables $x_1, \ldots, x_n$. Let $x_i^+, x_i^-$ denote the unnegated and negated literal of variable $x_i$, for $i = 1, \ldots, n$. We assume that no clause $C_p$, $p \in \{1, \ldots, m\}$, contains both the negated and unnegated literal of some variable, for if there were such a clause $C_p$ then $F$ is satisfiable if and only if $F - C_p$ is. We shall construct a tournament $T$ with ordered vertices and a maximal acyclic vertex set $S$ of $T$ such that there exists a maximal acyclic vertex set of $T$ lexicographically larger than $S$ if and only if there is a satisfying truth assignment for $F$. Tournament $T$ has one vertex for each clause, one vertex for each variable, two vertices $x_i^+, x_i^-$ for the literals of each variable $x_i$, and a special vertex $a$. The order of the vertices is as follows:
first the clauses, then $a$, then the variables, then the literals in any order. The arcs of $T$ are as follows:

$$(a, C_p), \quad \text{for } p = 1, \ldots, m,$$

$$(C_p, C_q), \quad \text{for } p, q = 1, \ldots, m \text{ with } p < q,$$

$$(x_i, a), (x_i^+, x_i), (x_i^-, x_i), (a, x_i^+), (a, x_i^-), (x_i^+, x_i^-), \quad \text{for } i = 1, \ldots, n,$$

$$(C_p, x_i), \quad \text{for } x_i^+ \in C_p \land x_i^- \in C_p, \quad \text{if } i = 1, \ldots, n, \ p = 1, \ldots, m,$$

$$(x_i, C_p), \quad \text{for } x_i^+ \notin C_p \land x_i^- \notin C_p, \quad \text{if } i = 1, \ldots, n, \ p = 1, \ldots, m,$$

$$(C_p, x_i^+), (C_p, x_i^-), \quad \text{for } i = 1, \ldots, n, \ p = 1, \ldots, m,$$

$$(x_i^+, x_j^-), (x_i^-, x_j^-), (x_j^+, x_i^-), (x_j^-, x_i^-), \quad \text{for } i, j = 1, \ldots, n \text{ with } i < j,$$

$$(x_i, x_j), (x_j, x_i^+), (x_i, x_j^-), \quad \text{for } i, j = 1, \ldots, n - 1 \text{ with } i < j,$$

$$(x_i, x_j), \quad \text{for } i = 2, \ldots, n - 1,$$

$$(x_1, x_n), (x_n, x_1^+), (x_1^-, x_n).$$

Let $S = \{a, x_1, \ldots, x_n\}$, so $S$ is a maximal acyclic vertex set of $G$.

Is there a maximal acyclic vertex set lexicographically larger than $S$? If there is then let $S^*$ be the lexicographically largest maximal acyclic vertex set with $S \prec S^*$. Then $S^*$ does not contain any clause vertex. Consider the following cases.

- If $a \in S^*$ and $x_i \notin S^*$ for all $i = 1, \ldots, n$ then $S^* \setminus \{a\}$ is a subset of the literals. Moreover, it contains at most one of $x_i^+, x_i^-$ for all $i = 1, \ldots, n$, to be acyclic. For $S^*$ to be maximal, for each clause $C_p$ there is at least one literal $\ell$ in $S^*$ for which $(C_p, 1)$ is an arc. In other words, $S^* \setminus \{a\}$ is a satisfying truth assignment for $F$.

- If $a \in S^*$ and $x_i \in S^*$ for some $i \in \{1, \ldots, n\}$ then $x_i^+, x_i^- \notin S^*$ and there are acyclic vertex sets $S^+ = S^* \setminus \{x_i\} \cup \{x_i^+\}$ and $S^- = S^* \setminus \{x_i\} \cup \{x_i^-\}$. Moreover, $S^+ \prec S^*$ and $S^* \prec S^-$, and one of $S^+, S^-$ is maximal: this contradicts the choice of $S^*$.

- If $a \notin S^*$ then let $I \subseteq \{1, \ldots, n\}$ be the set of indices $i$ for which $x_i \notin S^*$. For any $i \in I$, at least one of $x_i^+, x_i^-$ belongs to $S^*$, for $S^*$ to be maximal. Let $C(i)$ be the subset of clause vertices $C_p$ for which $(C_p, x_i)$ is an arc. Again by maximality of $S^*$, for any vertex $C_p \in C(i)$ there is a vertex $x_j$ with $j \notin I$ that is adjacent to $C_p$. For any $j \notin I$, exactly one of $x_j^+, x_j^-$ belongs to $S^*$, for $S^*$ to be acyclic and maximal. Thus, if $I' \subseteq I$ is the subset of indices for which both $x_j^+, x_j^-$ belong to $S^*$, the set $S^* \setminus \{(x_1, \ldots, x_n) \cup \{x_i^+ \mid i \in I'\})$ is a satisfying truth assignment for $F$.

Therefore, $S$ is the lexicographically largest maximal acyclic vertex set of $G$ if and only if the given Boolean formula is unsatisfiable. □
4.1 Minimal Feedback Vertex Sets in Tournaments

We study combinatorial and algorithmic questions around minimal feedback vertex sets in tournaments. On the combinatorial side, we derive strong upper and lower bounds on the maximum number of minimal feedback vertex sets in an \( n \)-vertex tournament. On the algorithmic side, we design the first polynomial space algorithm that enumerates the minimal feedback vertex sets of a tournament with polynomial delay. The combination of our results yields the fastest known algorithm for finding a minimum size feedback vertex set in a tournament.

Minimal feedback vertex sets in tournaments find applications in social choice theory. The complement of a minimal feedback vertex set \( F \) of \( T \) induces a maximal acyclic subtournament whose unique vertex of in-degree zero is a “Banks winner” [29]: identifying the vertices of \( T \) with candidates in a voting scheme and arcs indicating preference of one candidate over another, the Banks winner of \( T[V \setminus F] \) is the candidate collectively preferred to every other candidate in \( V \setminus F \). Banks winners play an important role in social choice theory.

Our results in this section are as follows.

**Extremal Combinatorics.** We denote the number of minimal feedback vertex sets in a tournament \( T \) by \( f(T) \), and the maximum \( f(T) \) over all \( n \)-vertex tournaments by \( M(n) \). The letter “\( M \)” was chosen in honor of Moon who in 1971 proved [197] that
\[
1.4757^n \leq M(n) \leq 1.7170^n.
\]

Our combinatorial main result are the stronger bounds
\[
1.5448^n \leq M(n) \leq 1.6740^n.
\]

To prove our new lower bound on \( M(n) \), we construct an infinite family of tournaments all having \( 21^{n/7} > 1.5448^n \) minimal feedback vertex sets. To prove our new upper bound on \( M(n) \), we bound the maximum of a convex function bounding \( M(n) \) from above, and otherwise rely on case distinctions and recurrence relations.

Similar combinatorial bounds have been obtained for minimal feedback vertex sets in general undirected graphs. Fomin et al. [106] show that any undirected graph on \( n \) vertices contains at most \( 1.8638^n \) minimal feedback vertex sets, and that infinitely many graphs have \( 105^{n/10} > 1.5926^n \) minimal feedback vertex sets. Lower bounds of roughly \( \log n \) on the size of a maximum acyclic subtournament have been obtained by Reid and Parker [219] and Neumann-Lara [203]. Other bounds on minimal or maximal sets with respect to vertex-inclusion have been obtained for dominating sets [110], bicliques [123], separators [116], potential max-
imal cliques [117], bipartite graphs [49], $r$-regular subgraphs [140], and, of course, independent sets [195, 198].

**Enumeration.** An algorithm by Schwikowski and Speckenmeyer [227] lists the minimal feedback vertex sets of a tournament $T$ with polynomial delay, by traversing the strongly directed graph whose vertices are bijectively mapped to minimal feedback vertex sets of $T$. Unfortunately the Schwikowski-Speckenmeyer-algorithm may use exponential space, and it is not known whether the problem of enumerating minimal feedback vertex sets allows for a polynomial delay algorithm with polynomially bounded space complexity in general graphs. Our algorithmic main result provides such an enumeration algorithm for the family of tournaments. Our algorithm is inspired from that by Tsukiyama et al. for the conceptually simpler enumeration of maximal independent sets [238]. It is based on iterative compression, a technique for parameterized [217] and exact algorithms [105]. We thereby positively answer Fomin et al.’s [105] question if the technique could be applied to other algorithmic areas.

**Exact Algorithms.** Any polynomial-delay enumeration algorithm for minimal feedback vertex sets yields an exact algorithm for finding a feedback vertex set of minimum size. Any combinatorial upper bound on $M(n)$ provides a corresponding bound on the time complexity of this exact algorithm; our bound yields a time complexity of $O(1.6740^n)$. This answers a question of Woeginger [250] who asked whether it is possible to beat the time complexity resulting from Moon’s upper bound. Dom et al. [77] independently answered this question by constructing an iterative–compression algorithm. However, the running time of their algorithm grows at least with $1.708^n$ and hence their result is inherently weaker than ours.

### 4.1.1 Preliminaries

Let $T = (V, A)$ be a tournament. For a vertex subset $V' \subseteq V$, the tournament $T[V']$ induced by $V'$ is called a subtournament of $T$. If there is an arc $(u, v) \in A$ then we say that $u$ beats $v$. A non-strong tournament $T$ has a unique factorization $T = S_1 + \ldots + S_r$ into strong subtournaments $S_1, \ldots, S_r$, where every vertex $u \in V(S_k)$ beats all vertices $v \in V(S_\ell)$, for $1 \leq k < \ell \leq r$. For $n \in \mathbb{N}$ let $T_n$ denote the set of tournaments with $n$ vertices and let $T^*_n$ denote the set of strong tournaments on $n$ vertices.

The score of a vertex $v \in V$ is the size of its out-neighborhood, and denoted by $s_v(T)$ or $s_v$ for short. Consider a labeling $1, \ldots, n$ of the vertices of $T$ such that their scores are non-decreasing, and associate with $T$ the score sequence $s(T) = (s_1, \ldots, s_n)$. If $T$ is strong then $s(T)$ satisfies the Landau inequalities [150, 180]:

$$\sum_{v=1}^{k} s_v \geq \binom{k}{2} + 1 \text{ for all } k = 1, \ldots, n - 1, \text{ and}$$  \hspace{1cm} (4.1)

$$\sum_{v=1}^{n} s_v = \binom{n}{2}$$  \hspace{1cm} (4.2)
For every non-decreasing sequence \( s \) of positive integers that satisfies conditions (4.1)–(4.2), there exists a tournament whose score sequence is \( s \) [180].

Let \( L \) be a set of non-zero elements from the ring \( \mathbb{Z}_n \) of integers modulo \( n \) such that for all \( i \in \mathbb{Z}_n \) exactly one of \( +i \) and \( -i \) belongs to \( L \). The tournament \( T_L = (V_L, A_L) \) with \( V_L = \{1, \ldots, n\} \) and \( A_L = \{(i, j) \in V_L \times V_L \mid (j - i) \mod n \in L \} \) is the circular \( n \)-tournament induced by \( L \). A triangle is a tournament of order 3. The cyclic triangle is denoted \( C_3 \).

Let \( F(T) \) be the collection of minimal feedback vertex sets of \( T \); its cardinality is denoted by \( f(T) \). Acyclic tournaments are sometimes called transitive; the (up to isomorphism unique) transitive tournament on \( n \) vertices is denoted \( TT_n \). Let \( \tau \) be the unique topological order of the vertices of \( TT_n \) such that \( \tau(u) < \tau(v) \) if and only if \( u \) beats \( v \). For such an order \( \tau \) and integer \( i \in \{1, \ldots, n\} \) the subsequence of the first \( i \) values of \( \tau \) is denoted \( \tau_i(V(TT_n)) = (\tau^{-1}(1), \ldots, \tau^{-1}(i)) \); call \( \tau_i(V(TT_n)) \) the source of \( TT_n \). For a minimal feedback vertex set \( F \) of a tournament \( T \) the subtournament \( T[V \setminus F] \) is a maximal transitive subtournament of \( T \) and \( V \setminus F \) is a maximal transitive vertex set.

### 4.1.2 Minimum Number of Minimal Feedback Vertex Sets

We analyze the minimum number of minimal feedback vertex sets in tournaments.

Let the function \( m : \mathbb{N} \to \mathbb{N}, n \mapsto \min_{T \in TT_n} f(T) \) count the minimum number of minimal feedback vertex sets over all tournaments of order \( n \). Since a minimal feedback vertex set always exists, \( m(n) \geq 1 \) for all positive integers \( n \). This bound is attained by the transitive tournaments \( TT_n \) of all orders \( n \).

**Observation 4.1** ([197]). If \( T = S_1 + \ldots + S_r \) is the factorization of a tournament \( T \) into strong subtournaments \( S_1, \ldots, S_r \) then \( f(T) = f(S_1) \cdots f(S_r) \).

Hence from now on we consider only strong tournaments (on at least 3 vertices) and define \( m^* : \mathbb{N} \setminus \{1, 2\} \to \mathbb{N}, n \mapsto \min_{T \in TT_n^*} f(T) \).

**Lemma 4.1.** The function \( m^* \) is constant: \( m^*(n) = 3 \) for all \( n \geq 3 \).

**Proof.** Let \( T \in T_n^* \) be a strong tournament. We show that \( f(T) \geq 3 \). As \( T \) is strong, it contains some cycle and thus some cyclic triangle \( C \), with vertices \( v_1, v_2, v_3 \).

For \( i = 1, 2, 3 \), define the vertex sets \( W_i = \{v_i, v_{(i+1) \mod 3}\} \). Every set \( W_i \) can be extended to a maximal transitive vertex set \( W_i^* \) of \( T \). Note that for \( i = 1, 2, 3 \) and \( j \in \{1, 2, 3\} \setminus \{i\} \), we have \( v_{(i+2) \mod 3} \in W_j^* \setminus W_i^* \). Hence, there are three maximal transitive subtournaments of \( T \) whose complements form three minimal feedback vertex sets of \( T \). Consequently, \( m^*(n) \geq 3 \) for all \( n \geq 3 \).

To complete the proof, construct a family \( \{U_n \in T_n^* \mid n \geq 3\} \) of strong tournaments with exactly three minimal feedback vertex sets. Set \( U_3 \) equal to the cyclic triangle. For \( n \geq 4 \), build the tournament \( U_n \) as follows: start with the transitive tournament \( TT_{n-2} \), whose vertices are labeled \( 1, \ldots, n-2 \) by decreasing scores. Then add two special vertices \( u_1, u_2 \) which are connected by an arbitrarily oriented arc. For \( i \in \{1, 2\} \), add arcs from all vertices \( 2, \ldots, n-2 \) to \( u_i \). Finally, connect vertex 1 to \( u_i \) by an arc \((u_i, 1)\), for \( i = 1, 2 \). The resulting tournament \( U_n \), depicted in Fig. 4.1, has exactly three minimal feedback vertex sets, namely \( \{u_1, u_2\}, \{1\} \) and \( \{2, \ldots, n-2\} \). □
4. ALGORITHMS THROUGH EXTREMAL COMBINATORICS

4.1.3 Lower Bound on the Maximum Number of Minimal Feedback Vertex Sets

We prove a lower bound of $21^{n/7} > 1.5448^n$ on the maximum number of minimal feedback vertex sets of tournaments with $n$ vertices.

Formally, we will bound from below the values of the function $M(n) = \max_{T \in T_n} f(T)$. By convention, set $M(0) = 1$. Note that $M$ is monotonically non-decreasing on its domain: given any tournament $T \in T_n$ and any vertex $v \in V(T)$, for every minimal feedback vertex set $F \in \mathcal{F}(T[V(T) \setminus \{v\}])$ either $F \in \mathcal{F}(T)$ or $F \cup \{v\} \in \mathcal{F}(T)$. As $T$ and $v$ are arbitrary, it follows that $M(n) \geq M(n-1)$.

We will now show that there is an infinite family of tournaments on $n = 7k$ vertices, one for each $k \in \mathbb{N}$, with $21^{n/7} > 1.5448^n$ minimal feedback vertex sets, improving upon Moon’s [197] bound of $1.4757^n$.

Let $ST_7$ denote the Paley directed graph of order 7, i.e. the circular 7-tournament induced by the set $L = \{1, 2, 4\}$ of quadratic residues modulo 7. All maximal transitive subtournaments of $ST_7$ are transitive triangles, of which there are exactly 21, as each vertex is the source of 3 distinct transitive triangles. Thus, all minimal feedback vertex sets for $ST_7$ are minimum feedback vertex sets. We remark that $ST_7$ is the unique 7-vertex tournament without any $TT_4$ as subtournament [219].

Lemma 4.2. There exists an infinite family of tournaments with $21^{n/7}$ minimal feedback vertex sets.

Proof. Let $k \in \mathbb{N}$ and form the tournament $T_0 = ST_7 + \ldots + ST_7$ from $k$ copies of $ST_7 \in T_7^n$. Then $T_0 \in T_n$ for $n = 7k$, and the number of minimal feedback vertex sets in $T_0$ is $f(T_0) = f(ST_7)^k = 21^k = 21^{n/7}$. □

4.1.4 Upper Bound on the Maximum Number of Minimal Feedback Vertex Sets

We give an upper bound of $\beta^n$, where $\beta = 1.6740$, on the maximum number of minimal feedback vertex sets in any tournament $T \in T_n$. This improves the bound of $1.7170^n$ by Moon [197]. Instead of minimal feedback vertex sets we count maximal transitive subtournaments, and with respect to Observation 4.1 we count the maximal transitive subtournaments of strong tournaments.

We start with three properties of maximal transitive subtournaments. First, for a strong tournament $T = (V, A)$ with score sequence $s = (s_1, \ldots, s_n)$ the following
holds: if \( TT_k = (V', A') \) is a maximal transitive subtournament of \( T \) with \( \tau_1(V') = (t) \) then \( T[V' \setminus \{ t \}] \) is a maximal transitive subtournament of \( T[N^+ \{ t \}] \). Hence \( f(T) \leq \sum_{v=1}^n M(s_v) \), where \( s_v \leq n - 2 \) for all \( v \in V \). This allows us to effectively bound \( f(T) \) via a recurrence relation.

Second, there cannot be too many vertices with large score.

**Lemma 4.3.** For \( n \geq 8 \) and \( k \in \{0, 1, 2\} \), any strong tournament \( T \in \mathcal{T}_n^s \) has at most \( 2(k+1) \) vertices of score at least \( n - 2 - k \).

**Proof.** Fix some strong tournament \( T \in \mathcal{T}_n^s \) and \( k \in \{0, 1, 2\} \). Suppose for contradiction that \( T \) contains \( 2k+3 \) vertices with score at least \( n - 2 - k \). Then the Landau inequalities (4.1) and (4.2) imply the contradiction

\[
2 \binom{n}{2} = 2 \left( \sum_{v=1}^{n-(2k+3)} s_v + \sum_{v=n-(2k+3)}^n s_v \right) \\
\geq 2 \left( \frac{(n-(2k+3))}{2} + 1 + (2k+3)(n-2-k) \right) = n^2 - n + 2.
\]

For \( n \leq 7 \), we can explicitly list the strong \( n \)-vertex tournaments for which the Lemma fails: the cyclic triangle for \( k = 0 \), the tournaments \( RT_5, ST_6 \) for \( k = 1 \) and \( ST_7 \) for \( k = 2 \). \( RT_5 \) is the regular tournament of order 5 and \( ST_6 \) is the tournament obtained by arbitrarily removing some vertex from \( ST_7 \) and all incident arcs.

Third, let \( T' \) be a tournament obtained from a tournament \( T \) with \( s(T) = (s_1, \ldots, s_n) \) by reversing all arcs of \( T \). Then, \( f(T) = f(T') \), whereas the score \( s_v(T) \) of each vertex \( v \) turns into \( s_v(T') = n-1-s_v(T) \). This implies that analyzing score sequences with maximum score \( s_n \geq n-1-c \) for some constant \( c \) is symmetric to analyzing score sequences with minimum score \( s_1 \leq c \).

Our proof that any tournament on \( n \) vertices has at most \( \beta^n \) maximal transitive subtournaments consists of several parts. We start by proving the bound for tournaments with few vertices. The inductive part of the proof first considers tournaments with large maximum score (and symmetrically small minimum score), and then all other tournaments.

We begin the proof by considering tournaments with up to 10 vertices. For \( n \leq 4 \) exact values for \( M(n) \) were known before [197]. For \( n = 5, \ldots, 9 \) we obtained exact values for \( M(n) \) with the help of a computer. For these values the extremal tournaments obey the following structure: pick a strong tournament \( T' \in \mathcal{T}_n^{s_{n-2}} \) and construct the strong tournament \( pq(T') \in \mathcal{T}_n^s \) by attaching two vertices to \( T' \) as in Fig. 4.2; namely add vertices \( p \) and \( q \) to \( T' \), and arcs \((q,p), (p,t), (t,q)\) for each vertex \( t \) in \( T' \). Then \( f(pq(T')) = 2f(T') + 1 \).

For \( n = 5 \), there are exactly two non-isomorphic strong tournaments \( QT_5 \cong \mathcal{T}_n^s \). For these, \( f(QT_5) = f(RT_5) = M(5) = 2 \cdot 3 + 1 = 7 \). For \( n = 6 \), \( ST_6 \) is the unique tournament from \( \mathcal{T}_n \) with \( f(ST_6) = M(6) = 12 \) minimal feedback vertex sets. For \( n = 7 \) the previous section showed \( f(ST_7) = 21 \), and in fact \( ST_7 \) is the unique 7-vertex tournament with \( M(7) = 21 \) minimal feedback vertex sets. For \( n \in \{8, 9\} \), \( ST_n \cong \mathcal{T}_n^{-2} \); then \( f(ST_n) = M(n) \). Table 4.1 summarizes that for \( n \leq 9 \), \( M(n) \leq \beta^n \).
Table 4.1: Extremal tournaments of up to 9 vertices

| n  | M(n) | M(n)^{1/n} ≈ | T ∈ T_n : f(T) = M(n) |
|----------------|----------------|------------------------|
| 1  | 1   | 1.00000     | T ∈ T_1                |
| 2  | 1   | 1.00000     | T ∈ T_2                |
| 3  | 3   | 1.44225     | T ∈ T_3 \ \{TT_3\}    |
| 4  | 3   | 1.31607     | T ∈ T_4 \ \{TT_4\}    |
| 5  | 7   | 1.47577     | QT_5 ≜ pq(C_5), RT_5  |
| 6  | 12  | 1.51309     | ST_6 ≜ ST_7 − \{1\}   |
| 7  | 21  | 1.54486     | ST_7                   |
| 8  | 25  | 1.49535     | ST_8 ≜ pq(ST_6)        |
| 9  | 43  | 1.51879     | ST_9 ≜ pq(ST_7)        |

Next, we bound \( M(10) \) by means of \( M(n) \) for \( n ≤ 9 \). Let \( W \) be a maximal transitive vertex set of \( T ∈ T_{10} \). Let \( v^* \) be a vertex of \( T \) with \( \sigma_{v^*}(T) = s_{10} \); then either \( v^* ∈ W \) or \( v^* \notin W \). There are at most \( M(s_{10}) = M(9) \) maximal transitive vertex sets \( W \) such that \( v^* ∈ W \) and at most \( M(9) \) such sets \( W \) for which \( v^* \notin W \). As \( (2M(9))^{1/10} = 86^{1/10} < 1.5612 \), the proof follows for all tournaments with at most 10 vertices.

For the rest of this section we consider tournaments with \( n ≥ 11 \) vertices. Let \( T = (V, A) \) be a strong tournament on \( n ≥ 11 \) vertices; we will show that \( f(T) ≤ \beta^2 \). The proof considers four main cases and several subcases with respect to the minimum and maximum score of the tournament.

The idea of the proof is as follows. By \( W \) we denote a maximal transitive vertex set of \( T \). If there is a vertex \( v \) in \( T \) of large score at least \( n − 3 \), then either \( v \) is the source of \( W \) or only few other vertices can be the source of \( W \). We can then look at the subtournament induced by these few vertices, and branch on their inclusion with respect to \( W \). In this way, we fix the first few elements of the acyclic ordering of \( W \). Moreover, there cannot be too many vertices of large score by Lemma 4.3. Suppose that in one branch, \( \tau_b(W) = (a_1, a_2, \ldots, a_k) \) and for some \( i ∈ \{1, \ldots, k\} \), \( |N^{+(a_i)} \setminus W| ≥ c \), then we can upper bound the number of such maximal transitive vertex sets \( W \) by \( M(a_i − (k − i) − c) \). The case when some vertex \( v \) in \( T \) has small score at most 2 is symmetric.

The tightest case of our proof is the following: \( s_n = n − 3, s_{b_1} = n − 3, s_{b_2} = n − 4 \), where \( b_1, b_2 \) are the two in-neighbors of \( n \) connected by an arc \( (b_1, b_2) \), and \( N^+(b_1) ≠ N^-(b_2) \) \( \setminus \{b_1\} \). Denote by \( c_1, c_2 \) the in-neighbors of \( b_1 \) connected by an arc \( (c_1, c_2) \), and by \( d_1, d_2 \) the in-neighbors of \( b_2 \) connected by an arc \( (d_1, d_2) \). We count the different maximal transitive vertex sets \( W \) depending on the membership or non-membership of \( b_1, b_2 \), and \( n \) in \( W \).

1. If \( b_1, b_2 \notin W \), then \( n ∈ W \) by maximality of \( W \) and \( \tau_1(W) = (n) \) as no vertex in \( W \) beats \( n \). There are at most \( M(s_n) = M(n − 3) \) such \( W \).
2. If \( b_1, n \notin W \) and \( b_2 ∈ W \), then some in-neighbor of \( b_2 \) is in \( W \), otherwise \( W \) were not maximal as \( W \cup \{n\} \) would be a transitive vertex set. There are at most \( M(s_{b_2} − 1) = M(n − 5) \) possibilities for \( \tau_2(W) = (d_2, b_2) \), at most \( M(s_{d_1} − 2) ≤ M(n − 5) \) for \( \tau_2(W) = (d_1, b_2) \), and at most \( M(s_{d_1} − 2) ≤ M(n − 5) \) for \( \tau_3(W) = (d_1, d_2, b_2) \).
4.1. MINIMAL FEEDBACK VERTEX SETS IN TOURNAMENTS

(3) If \( b_1 \notin W \) and \( b_2, n \in W \), then \( \tau_2(W) = (b_2, n) \). There are at most \( M(s_{b_2} - 1) = M(n - 5) \) such \( W \).

(4) If \( n \notin W \) and \( b_1 \in W \), then we consider two subcases. If \( N^-(b_1) \cap W \neq \emptyset \), then some in-neighbor of \( b_1 \) is the source of \( W \). There are at most \( M(s_{c_2} - 1) \leq M(n - 4) \) possibilities for \( \tau_2(W) = (c_2, b_1) \), at most \( M(s_{c_2} - 2) \leq M(n - 5) \) for \( \tau_3(W) = (c_1, c_2, b_1) \), and at most \( M(s_{c_2} - 2) \leq M(n - 5) \) for \( \tau_2(W) = (c_1, b_1) \). Otherwise, no in-neighbor of \( b_1 \) is in \( W \), and thus, \( \tau_1(W) = (b_1) \). Moreover, some in-neighbor of \( b_2 \) is the source of \( T[W \setminus \{b_1\}] \), otherwise \( n \) could be added. This leaves us with a total of at most \( 3M(s_{b_1} - 4) = 3M(n - 7) \) possibilities for which \( \tau_4(W) = (b_1, d_1, d_2, b_2) \), \( \tau_5(W) = (b_1, d_2, b_2) \), or \( \tau_3(W) = (b_1, d_1, b_2) \).

(5) If \( b_2 \notin W \) and \( b_1, n \in W \), then \( \tau_2(W) = (b_1, n) \). There are at most \( M(s_{b_1} - 2) = M(n - 5) \) such \( W \).

(6) If \( b_1, b_2, n \in W \), then \( \tau_3(W) = (b_1, b_2, n) \). As at least one out-neighbor of \( b_2 \) is an in-neighbor of \( b_1 \), there are at most \( M(s_{b_2} - 2) = M(n - 6) \) such \( W \).

Altogether, in this case,

\[
f(T) \leq M(n - 3) + 3M(n - 5) + M(n - 5) + (M(n - 4) + 2M(n - 5) + 3M(n - 7) + M(n - 5) + M(n - 6) \\
\leq 3\beta^n - 7 + \beta^{n-6} + 7\beta^{n-5} + \beta^{n-4} + \beta^{n-3},
\]

which is at most \( \beta^n \) because \( \beta \geq 1.6740 \).

Now suppose that every vertex in \( T \) has score at least three and at most \( n - 4 \). In that case we define a linear function \( G_n \) mapping feasible score sequences \( s = (s_1, \ldots, s_n) \) to \( \sum_{i=1}^n \beta_{s_i} \) for \( \beta = 1.6740 \). We then define special score sequences \( \sigma(n) \) and show that these sequences maximize \( G_n \), based on the strict convexity of \( G_n \). For example,

\[
\sigma(17) = (3, 3, 3, 3, 3, 3, 4, 7, 8, 9, 12, 13, 13, 13, 13, 13, 13).
\]

The proof is completed by bounding \( f(n) \) in terms of \( G(\sigma(n)) \).

We now provide a complete proof of the upper bound on the maximum number of minimal FVSs in tournaments.

Let \( T = (V, A) \) be a strong tournament on \( n \geq 11 \) vertices and let \( s = (s_1, \ldots, s_n) \) be the score sequence of \( T \). We will show that \( f(T) \leq \beta^n \). The proof considers four main cases and several subcases with respect to the minimum and maximum score of the tournament. To avoid a cumbersome nesting of cases, whenever inside a given case we assume that none of the earlier cases applies. By \( W \) we denote a maximal transitive vertex set of \( T \).

**Case 1**: \( s_n = n - 2 \). Let \( b \) be the unique vertex beating vertex \( n \).

If \( b \notin W \) then \( \tau_1(W) = (n) \); there are at most \( M(s_n) = M(n - 2) \) such \( W \).

If \( b \in W \) and \( n \in W \), then \( \tau_1(W \setminus \{b\}) = (n) \) as no vertex except \( b \) beats \( n \). So, \( \tau_2(W) = (b, n) \) and there are at most \( M(s_{b} - 1) \) such \( W \). For the last possibility, where \( b \in W \) and \( n \notin W \), note that \( W \) contains at least one in-neighbor of \( b \), other-
wise $W$ were not maximal as $n$ could be added. We consider 4 subcases depending on the score of $b$.

**Case 1.1:** $s_b = n - 2$. Let $c$ be the unique vertex beating $b$. As at most 2 vertices have score $n - 2$ by Lemma 4.3, $s_c \leq n - 3$. We have that $c \in W$, otherwise $W$ would not be maximal as $W \cup \{n\}$ induces a transitive subtournament of $T$. As $b$ and its unique in-neighbor $c$ are in $W$, $\tau_2(W) = (c,b)$. There are at most $M(s_c - 1) \leq M(n - 4)$ such $W$. In total, $f(T) \leq M(n - 2) + M(n - 3) + M(n - 4) \leq \beta^{n-4} + \beta^{n-3} + \beta^{n-2}$ which is at most $\beta^n$ because $\beta \geq 1.4656$.

In the three remaining subcases, all in-neighbors of $b$ have score at most $n - 3$: if $c_i \in N^-(b)$ had score $n - 2$, then Case 1.1 would apply with $n := c_i$ and $b := n$.

**Case 1.2:** $s_b = n - 3$. Let $N^-(b) := \{c_1, c_2\}$ such that $(c_1, c_2) \in A$. Then either $\tau_1(W) = (c_1)$ or $\tau_1(W) = (c_2)$; there are at most $2M(n - 3)$ such $W$. It follows $f(T) \leq M(n - 2) + M(n - 4) + 2M(n - 3) \leq \beta^{n-4} + 2\beta^{n-3} + \beta^{n-2} \leq \beta^n$ as $\beta \geq 1.6181$.

**Case 1.3:** $s_b = n - 4$. Let $N^-(b) := \{c_1, c_2, c_3\}$. Observe that at most 2 vertices among $N^-(b)$ have score $n - 3$, otherwise $T$ is not strong as $N^-(b) \cup \{b,n\}$ induce a strong component. Either $\tau_1(W) = (c_1)$ or $\tau_1(W) = (c_2)$ or $\tau_1(W) = (c_3)$; there are at most $2M(n - 3) + M(n - 4)$ such $W$. Thus, $f(T) \leq M(n - 2) + M(n - 5) + 2M(n - 3) + M(n - 4) \leq \beta^{n-5} + \beta^{n-4} + 2\beta^{n-3} + \beta^{n-2} \leq \beta^n$ as $\beta \geq 1.6664$.

**Case 1.4:** $s_b \leq n - 5$. Then there are at most $M(n - 1)$ subtournaments not containing $n$. It follows $f(T) \leq M(n - 2) + M(n - 6) + M(n - 1) \leq \beta^{n-6} + \beta^{n-2} + \beta^{n-1} \leq \beta^n$ as $\beta \geq 1.6737$.

**Case 2:** $s_n = n - 3$. Let $b_1, b_2$ be the two vertices beating $n$ such that $(b_1, b_2) \in A$. The tree in Fig. 4.3 pictures our case distinction. Its leaves correspond to six different cases, numbered (1)–(6), for membership or non-membership of $n, b_1$ and $b_2$ in some maximal transitive vertex set $W$ of $T$. The cases corresponding to leaves (2) and (4) will be considered later. Let us now bound the number of possible $W$ for the other cases (1), (3), (5) and (6).

![Figure 4.3: Different possibilities for a maximal transitive vertex set $W$.](image-url)
Claim 4.1. Among all maximal transitive vertex sets $W$ of $T$,

(1) at most $M(n - 3)$ are such that $b_1 \notin W$ and $b_2 \notin W$,

(3) at most $M(s_{b_2} - 1)$ are such that $b_1 \notin W$, $b_2 \in W$ and $n \in W$,

(5) at most $M(s_{b_1} - 2)$ are such that $b_1 \in W$, $b_2 \notin W$ and $n \in W$, and

(6) at most $M(s_{b_1} - 2)$ are such that $b_1 \in W$, $b_2 \in W$ and $n \in W$.

Proof. If (1) $b_1 \notin W$ and $b_2 \notin W$, then $n \in W$ by maximality of $W$ and $n$ is the source of $T[W]$ as no vertex in $W$ beats $n$. Thus, there are at most $M(s_n) = M(n - 3)$ such $W$. If (3) $b_1 \notin W$, $b_2 \in W$ and $n \in W$, then $\tau_1(W \setminus \{b_2\}) = (n)$. Therefore, $\tau_2(W) = (b_2, n)$ and there are at most $M(s_{b_2} - 1)$ such $W$. If (5) $b_1 \in W$, $b_2 \notin W$ and $n \in W$, then $\tau_2(W) = (b_1, n)$, and as $b_1$ beats $b_2$, there are at most $M(s_{b_2} - 2)$ such $W$. If (6) $b_1 \in W$, $b_2 \in W$ and $n \in W$, then $\tau_3(W) = (b_1, b_2, n)$, and there are at most $M(s_{b_1} - 2)$ such $W$. \qed

To bound the number of subtournaments corresponding to the conditions in leaves (2) and (4), we will consider five subcases depending on the scores of $b_1$ and $b_2$. If $b_1$ and $b_2$ have low scores (Cases 2.4 and 2.5), there are few maximal transitive subtournaments of $T$ corresponding to the conditions in the leaves (3), (5) and (6). Then, it will be sufficient to group the cases (2) and (4) into one case where $n \notin W$ and to note that there are at most $M(n - 1)$ such subtournaments. Otherwise, if the scores of $b_1$ and $b_2$ are high (Cases 2.1 – 2.3), we use that in (2), some vertex of $N^{-}(b_2)$ is the source of $W$. If this were not the case, $W$ would not be maximal as $W \cup \{n\}$ would induce a transitive tournament. Similarly, in (4) some vertex of $N^{-}(b_1)$ is the source of $W$ if $b_2 \notin W$.

Let $c_{i_1}, \ldots, c_{|N^{-}(b_1)| - 1}$ be the in-neighbors of $b_1$ such that $(c_i, c_{i+1}) \in A$ for all $i \in \{1, \ldots, |N^{-}(b_1)| - 1\}$ (every tournament has a Hamilton path [216]) and let $d_{i_1}, \ldots, d_{|N^{-}(b_2)| - 1}$ be the in-neighbors of $b_2$ besides $b_1$ such that $(d_i, d_{i+1}) \in A$ for all $i \in \{1, \ldots, |N^{-}(b_2)| - 1\}$.

Let us first bound the number of subtournaments satisfying the conditions of (2) depending on $s_{b_2}$.

Claim 4.2. If $s_{b_2} = n - 3$, there are at most $M(s_{d_1} - 1)$ maximal transitive vertex sets $W$ such that $b_1 \notin W$, $b_2 \in W$ and $n \notin W$.

Proof. As mentioned above, some in-neighbor of $b_2$ is the source of $W$. As $s_{b_2} = n - 3$, $N^{-}(b_2) \setminus \{b_1\} = \{d_1\}$. Thus, $\tau_2(W) = (d_1, b_2)$ and there are at most $M(s_{d_1} - 1)$ such tournaments. \qed

Claim 4.3. If $s_{b_2} = n - 4$, there are at most $M(n - 5) + 2M(s_{d_1} - 2)$ maximal transitive vertex sets $W$ such that $b_1 \notin W$, $b_2 \in W$ and $n \notin W$.

Proof. If $d_1 \notin W$ then $\tau_2(W) = (d_2, b_2)$ and there are at most $M(s_{b_2} - 1) = M(n - 5)$ such $W$. Otherwise, $d_1 \in W$ and either $d_2 \notin W$ in which case $\tau_2(W) = (d_1, b_2)$, or $d_2 \in W$ in which case $\tau_3(W) = (d_1, d_2, b_2)$. There are at most $2M(s_{d_1} - 2)$ such $W$. \qed

The next step is to bound the number of subtournaments satisfying the conditions of (4) depending on $s_{b_1}$.
Claim 4.4. If $s_{b_1} = n - 3$, the number of maximal transitive vertex sets $W$ such that $b_1 \in W$ and $n \notin W$ is at most $2M(n - 5) + M(n - 4)$ if $b_2$ beats no vertex of $N^-(b_1)$, and otherwise at most $2M(n - 5) + M(n - 4) + M(n - 6)$ if $s_{b_2} = n - 3$ and at most $2M(n - 5) + M(n - 4) + 3M(n - 7)$ if $s_{b_2} = n - 4$.

Proof. If $N^-(b_1) \cap W \neq \emptyset$, then $c_1$ or $c_2$ is the source of $W$. The number of subsets $W$ such that $c_1 \notin W$, and thus $\tau_2(W) = (c_2, b_1)$, is at most $M(s_{c_2} - 1) \leq M(n - 4)$. The number of subsets $W$ such that $c_1 \in W$, and thus $\tau_3(W) = (c_1, c_2, b_1)$ or $\tau_2(W) = (c_1, b_1)$, is at most $2M(s_{c_1} - 2) \leq 2M(n - 5)$. If, on the other hand, $N^-(b_1) \cap W = \emptyset$, then $\tau_1(W) = (b_1)$ and some in-neighbor of $b_2$ is the source of $T[W \setminus \{b_1\}]$, otherwise $W$ is not maximal as $n$ can be added. Also note that $b_2$ beats some vertex of $N^-(b_1)$ (we have $N^-(b_2) \setminus N^-(b_1) \neq \emptyset$ as $N^-(b_1) \cap W = \emptyset$ but $N^-(b_2) \cap W \neq \emptyset$). If $s_{b_2} = n - 3$, we upper bound the number of such subsets $W$ by $M(s_{b_2} - 3) = M(n - 6)$ as $\tau_3(W) = (b_1, a_1, b_2)$. If $s_{b_2} = n - 4$, we have that $\tau_4(W) = (b_1, a_1, a_2, b_2)$, $\tau_3(W) = (b_1, a_2, b_2)$ or $\tau_2(W) = (b_1, a_1, b_2)$. Thus, there are at most $3M(s_{b_2} - 4) = 3M(n - 7)$ possible $W$ such that $N^-(b_1) \cap W = \emptyset$ if $s_{b_1} = n - 3$ and $s_{b_2} = n - 4$. Summarizing, there are at most $2M(n - 5) + M(n - 4)$ subsets $W$ if $b_2$ beats no vertex of $N^-(b_1)$, and otherwise at most $2M(n - 5) + M(n - 4) + M(n - 6)$ subsets $W$ if $s_{b_2} = n - 3$ and at most $2M(n - 5) + M(n - 4) + 3M(n - 7)$ subsets $W$ if $s_{b_2} = n - 4$. \[\square\]

Claim 4.5. If $s_{b_1} = n - 4$ and $s_{b_2} = n - 3$, the number of maximal transitive vertex sets $W$ such that $b_1 \in W$ and $n \notin W$ is

- at most $M(n - 7) + \sum_{c \in N^-(b_1)} 2M(s_c - 2)$ if $T[N^-(b_1)]$ is isomorphic to $C_3$,
- at most $\max\{M(n - 3) + M(n - 4) + M(n - 5); M(n - 5) + 6M(n - 6)\}$ if $T[N^-(b_1)]$ is transitive and $d_1 \in N^-(b_1)$, and
- at most $M(n - 3) + M(n - 4) + M(n - 5) + M(n - 7)$ if $T[N^-(b_1)]$ is transitive and $d_1 \notin N^-(b_1)$.

Proof. If $(c_2, c_1) \in A$ then $W$ intersects $N^-(b_1)$ in at most $2^3 - 1 = 7$ possible ways ($N^-(b_1) \subseteq W$ would induce a cycle in $T[W]$). In one of them, $N^-(b_1) \cap W = \emptyset$, which implies $\tau_3(W) = (b_1, a_1, b_2)$; there are at most $M(s_{b_1} - 3) = M(n - 7)$ such $W$. For each $c \in N^-(b_1)$, there are 2 possibilities where $\tau_1(W) = (c)$, one where $\tau_2(W) = (c, b_1)$ and one where $\tau_3(W) = (c, y, b_1)$ where $y$ is the out-neighbor of $c$ in $N^-(b_1)$; there are $2M(s_c - 2)$ such $W$ for each choice of $c$. In total, there are at most $M(n - 7) + \sum_{c \in N^-(b_1)} 2M(s_c - 2)$ possible $W$.

If, on the other hand, $(c_1, c_3) \in A$, first assume that $s_{c_1} \leq n - 3$, $s_{c_2} \leq n - 4$, and $s_{c_3} \leq n - 5$. Then either some vertex of $N^-(b_1)$ is the source of $W$ (at most $M(n - 3) + M(n - 4) + M(n - 5)$ possibilities for $W$), or $\tau_3(W) = (b_1, a_1, b_2)$ (at most $M(n - 7)$ possibilities for $W$). Otherwise, it must be that $s_{c_1} \leq n - 3$, $s_{c_2} \leq n - 4$, $s_{c_3} = n - 4$ and that $d_1 = c_3$. Then, $\tau_2(W) = (c_3, b_1)$, $\tau_2(W) = (c_2, b_1)$, $\tau_3(W) = (c_2, c_3, b_1)$, $\tau_2(W) = (c_1, b_1)$, $\tau_3(W) = (c_1, c_2, b_1)$, or $\tau_4(W) = (c_1, c_2, c_3, b_1)$; there are at most $M(n - 5) + 6M(n - 6)$ such $W$. In total, if $d_1 \in N^-(b_1)$, the number of possible $W$ can be upper bounded by $\max\{M(n - 3) + M(n - 4) + M(n - 5); M(n - 5) + 6M(n - 6)\}$, and if $d_1 \notin N^-(b_1)$, the number of possible $W$ can be upper bounded by $M(n - 3) + M(n - 4) + M(n - 5) + M(n - 7)$. \[\square\]
Armed with Claims 4.2–4.5, we now analyze the five subcases of Case 2, depending on the scores of $b_1$ and $b_2$.

**Case 2.1: $s_{b_1} = n - 3$, $s_{b_2} = n - 3$.** By Claim 4.2, the number of maximal transitive vertex sets $W$ such that $b_1, n \notin W$ and $b_2 \in W$ (leaf (2) in Fig. 4.3) is at most $M(n - 4)$. By Claim 4.4, the number of maximal transitive vertex sets $W$ such that $b_1, n \notin W$ and $b_2 \in W$ (leaf (4) in Fig. 4.3) is at most $2M(n - 5) + M(n - 4)$, at most $2M(n - 5) + M(n - 4) + M(n - 6)$, or at most $2M(n - 5) + M(n - 4) + 3M(n - 7)$. Combined with Claim 4.1,

$$f(T) \leq \max \begin{cases} M(n - 3) + M(n - 4) + M(n - 4) + (2M(n - 5) \\
+ M(n - 4)) + M(n - 5) + M(n - 5) \\
\leq 4\beta^{n-5} + 3\beta^{n-4} + \beta^{n-3} \leq \beta^n \text{ as } \beta \geq 1.6314, \\
M(n - 3) + M(n - 4) + M(n - 4) + (2M(n - 5) \\
+ M(n - 4) + M(n - 6)) + M(n - 5) + M(n - 5) \\
\leq \beta^{n-6} + 4\beta^{n-5} + 3\beta^{n-4} + \beta^{n-3} \leq \beta^n \text{ as } \beta \geq 1.6516, \\
M(n - 3) + M(n - 4) + M(n - 4) + (2M(n - 5) \\
+ M(n - 4) + 3M(n - 7)) + M(n - 5) + M(n - 5) \\
\leq 3\beta^{n-7} + 4\beta^{n-5} + 3\beta^{n-4} + \beta^{n-3} \leq \beta^n \text{ as } \beta \geq 1.6666. \end{cases}$$

**Case 2.2: $s_{b_1} = n - 3$, $s_{b_2} = n - 4$.** If $(c_1, b_2) \in A$ and $(c_2, b_2) \in A$ then $b_1 \notin W$ and $b_2 \in W$ implies that some in-neighbor $c$ of $b_1$ is in $W$, otherwise $W \cup \{b_1\}$ would induce a transitive tournament. But then, $n \notin W$, otherwise $\{c, b_2, n\}$ induces a directed cycle. This means that no maximal transitive vertex set $W$ satisfies the conditions of leaf (3) in Fig. 4.3. We bound the possible $W$ corresponding to leaves (2)+(4) by $M(n - 1)$ and obtain

$$f(T) \leq M(n - 3) + M(n - 1) + M(n - 5) + M(n - 5) \leq 2\beta^{n-5} + \beta^{n-3} + \beta^{n-1} \leq \beta^n \text{ as } \beta \geq 1.6440.$$  

Otherwise, there is some vertex $c \in N^{-}(b_1)$ such that $(b_2, c) \in A$. Then the number of $W$ in leaf (6) of Fig. 4.3 is upper bounded by $M(s_{b_2} - 2) = M(n - 6)$, and by Claims 4.3 and 4.4 those in leaves (2) and (4) are upper bounded by $M(n - 5) + 2M(s_{d_1} - 2)$ and $2M(n - 5) + M(n - 4) + 3M(n - 7)$, respectively. Thus,

$$f(T) \leq M(n - 3) + (M(n - 5) + 2M(n - 5)) + M(n - 5) + (2M(n - 5) \\
+ M(n - 4) + 3M(n - 7)) + M(n - 5) + M(n - 5) \leq 3\beta^{n-7} + 4\beta^{n-5} + 3\beta^{n-4} + \beta^{n-3} \leq \beta^n \text{ as } \beta \geq 1.6740.$$  

**Case 2.3: $s_{b_1} = n - 4$, $s_{b_2} = n - 3$.** By Claim 4.2, at most $M(n - 4)$ subsets $W$ correspond to leaf (2) in Fig. 4.3. If $N^{-}(b_1)$ induces a directed cycle, Claim 4.5 upper bounds the number of subsets corresponding to leaf (4) by $M(n - 7) + 2M(n - 6) + 4M(n - 5)$ as at most 2 vertices except $b_2$ and $n$ have score $n - 3$ by Lemma 4.3.
Together with Claim 4.1, this gives
\[
f(T) \leq M(n-3) + M(n-4) + M(n-4) + (M(n-7) + 2M(n-6) + 4M(n-5)) + M(n-6) + M(n-6)
\leq \beta^{n-7} + 4\beta^{n-6} + 4\beta^{n-5} + 2\beta^{n-4} + \beta^{n-3} \leq \beta^n \text{ as } \beta \geq 1.6670 .
\]

Otherwise, \((c_1, c_3) \in A\). If \((d_1, b_1) \in A\) then Claim 4.5 upper bounds the number of subsets corresponding to leaf (4) by \(M(n-3) + M(n-4) + M(n-5)\) or \(M(n-5) + 6M(n-6)\). Then,
\[
f(T) \leq \max \begin{cases} M(n-3) + M(n-4) + M(n-4) + (M(n-3) + M(n-4) + M(n-5)) + M(n-6) + M(n-6) \\ + 6M(n-6) + M(n-6) \end{cases}
\leq 8\beta^{n-6} + \beta^{n-5} + 2\beta^{n-4} + \beta^{n-3} \leq \beta^n \text{ as } \beta \geq 1.6396 .
\]

Otherwise, \((b_1, d_1) \in A\). For the possible \(W = b_1, b_2, n \in W\), none of \(N^- (b_1) \cup \{d_1\}\) is in \(W\) as these vertices all create cycles with \(b_1, b_2, n\). Thus, the number of possible subsets \(W\) corresponding to leaf (6) is upper bounded by \(M(s_{b_1} - 3) = M(n-7)\). Then, by Claims 4.1 and 4.5,
\[
f(T) \leq M(n-3) + M(n-4) + M(n-4) + (M(n-3) + M(n-4) + M(n-5)) + M(n-6) + M(n-7)
\leq 2\beta^{n-7} + \beta^{n-6} + \beta^{n-5} + 3\beta^{n-4} + 2\beta^{n-3} \leq \beta^n \text{ as } \beta \geq 1.6672 .
\]

**Case 2.4:** \(s_{b_1} = n-4, s_{b_2} \leq n-4\). By grouping leaves (2) and (4) into one possibility where \(n \not\in W\), Claim 4.1 upper bounds the number of such maximal transitive vertex sets by
\[
f(T) \leq M(n-3) + M(n-1) + M(n-5) + M(n-6) + M(n-6)
\leq 2\beta^{n-6} + \beta^{n-5} + \beta^{n-3} + \beta^{n-1} \leq \beta^n \text{ as } \beta \geq 1.6570 .
\]

**Case 2.5:** \(s_{b_1} \leq n-5\). By grouping leaves (2) and (4) into one possibility where \(n \not\in W\), Claim 4.1 upper bounds the number of such maximal transitive vertex sets by
\[
f(T) \leq M(n-3) + M(n-1) + M(n-4) + M(n-7) + M(n-7)
\leq 2\beta^{n-7} + \beta^{n-4} + \beta^{n-3} + \beta^{n-1} \leq \beta^n \text{ as } \beta \geq 1.6679 .
\]

**Case 3:** \(s_n \leq n-4\). We may assume that the score sequence \(s = s(T)\) satisfies
\[
3 \leq s_1 \leq \ldots \leq s_n \leq n-4 . \tag{4.3}
\]

Let \(S_n\) be the set of all score sequences that are feasible for (4.1)–(4.3). The set \(S_n\) serves as domain of the linear map \(G : S_n \to \mathbb{R}_+, s \mapsto \sum_{v=1}^n g(s_v)\) with the strictly
convex terms $g : c \mapsto \beta^c$. Furthermore, for all $n \geq 11$, we define a special score sequence $\sigma(n)$, whose membership in $S_n$ is easy to verify:

$$
\sigma(n) := \begin{cases} 
(3,3,3,3,3,3,3,3,3,3,3,3,3,3) & \text{if } n = 11, \\
(3,3,3,3,3,3,6,9,9,9,9,9) & \text{if } n = 13, \text{ and} \\
(3,3,3,3,3,4,7,7,7,7,7) & \text{if } n = 14.
\end{cases}
$$

**Lemma 4.4.** For $n \geq 11$, the sequence $\sigma(n)$ maximizes the value of $G$ over all sequences in $S_n$: $G(s) \leq G(\sigma(n))$ for all $s \in S_n$.

Once Lemma 4.4 is proved we can bound $f(T)$, for $s = s(T) \in S_n$, from above via

$$
f(T) \leq G(s) \leq G(\sigma(n)) = \begin{cases} 
5\beta^3 + \beta^5 + 5\beta^7, & \text{if } n = 11, \\
6\beta^3 + 6\beta^8, & \text{if } n = 12, \\
6\beta^3 + \beta^6 + 6\beta^9, & \text{if } n = 13, \\
6\beta^3 + \beta^4 + \frac{(\beta^{n-7} - \beta^7)}{\beta - 1} + \beta^{n-5} + 6\beta^{n-4} & \text{if } n \geq 14,
\end{cases}
$$

(4.4)

which is at most $\beta^n$ as $\beta \geq 1.6259$. To prove Lemma 4.4, we choose any sequence $s \in \text{argmax}_{s' \in S_n} G(s')$ and then show that $s = \sigma(n)$. Recall that $s_1 \geq 3$ and $s_n \leq n - 4$, and set $s^*_1 = 3, s^*_n = n - 4$.

**Claim 4.6.** If some score $c$ appears more than once in $s$, then $c \in \{s^*_1, s^*_n\}$.

**Proof.** For contradiction, suppose that $s^*_1 < s_u = s_v = c < s^*_n$ for two vertices $u$ and $v$ such that $1 \leq u < v \leq n$. First, suppose there exists an integer $k \in \{u, \ldots, v - 1\}$ satisfying (4.1) with equality:

$$
\sum_{v=1}^{k} s_v = \binom{k}{2} + 1.
$$

(4.5)

Then (4.1), (4.2) and Lemma 4.3 imply $8 \leq k \leq n - 9$, so $k \notin \{s^*_1, s^*_n\}$. The choice of $k$ among vertices of equal score $c$ now yields

$$
\sum_{v=1}^{k+1} s_v = s_k = \sum_{v=1}^{k} s_v - \sum_{v=1}^{k-1} s_v \leq \binom{k}{2} + 1 - \binom{k-1}{2} - 1 = k - 1.
$$

(4.6)

This however contradicts (4.1):

$$
\sum_{v=1}^{k+1} s_v \leq \binom{k}{2} + 1 + (k - 1) = \binom{k+1}{2}.
$$

It is thus asserted that no vertex $k$ with property (4.5) exists. The score sequence $s'$
differing from \( s \) only in \( s'_u = s_u - 1 = c - 1, s'_v = s_v + 1 = c + 1 \), therefore belongs to \( S_n \). So apply the function \( G \) to it, and use the strict convexity of \( g \):

\[
G(s') - G(s) = (g(c + 1) - g(c)) - (g(c) - g(c - 1)) > 0.
\]

This contradicts the choice of \( s \) as a maximizer of \( G \), and establishes Claim 4.6. \( \square \)

**Claim 4.7.** The values \( s^*_1 = 3 \) and \( s^*_n = n - 4 \) each appear between two and six times as scores in the sequence \( s \).

**Proof.** By Lemma 4.3, \( s^*_n \) is the score of no more than 6 vertices. By symmetry, \( s^*_1 \) is the score of no more than 6 vertices. As a consequence of Claim 4.6, together \( s^*_1 \) and \( s^*_n \) appear at least eight times in \( s \). Hence there are at least two vertices of score \( s^*_1 \) and at least two vertices of score \( s^*_n \). \( \square \)

**Claim 4.8.** If \( n \geq 12 \), each of \( s^*_1 \) and \( s^*_n \) is the score of exactly six of the vertices.

**Proof.** Assuming this were not the case for \( s^*_1 \), by Claim 4.7 it would be the score of two to five vertices. Hence there exists a vertex \( a \in \{3, \ldots, 6 \} \) with score \( s_a > s^*_1 \). It holds \( s^*_n = n - 4 > a + 1 \), which is obvious if \( n \geq 13 \) and follows from (4.2) if \( n = 12 \). So there must be two scores in \( s \) larger than \( s_n \), precisely \( s_a < s_{a+1} < s_{a+2} \). Observe that the sequence \( s' = (s_1, \ldots, s_{a-1}, s_a - 1, s_{a+1} + 1, s_{a+2}, \ldots, s_n) \) is a member of \( S_n \).

The same argument on strict convexity of \( g \) as in Claim 4.6 gives

\[
G(s') - G(s) = (g(y + 1) - g(y)) - (g(x) - g(x - 1)) > 0
\]

for \( x = s_a < s_{a+1} = y \), again contradicting the choice of \( s \) as a maximizer of \( G \). Consequently, the sequence \( s \) starts with six scores \( s^*_1 \). By symmetry, the same argumentation also applies for \( s^*_n \), proving the claim. \( \square \)

**Claim 4.9.** If \( n = 11 \), each of \( s^*_1 \) and \( s^*_n \) is the score of exactly five of the vertices.

**Proof.** As all scores are between 3 and 7, at most 5 vertices have score 3 and at most 5 vertices have score 7 by (4.2). Assume less than 5 vertices have score \( s^*_1 \). By Claim 4.7, \( s^*_1 \) is the score of two to four vertices. Hence there exists a vertex \( a \in \{3, 4, 5 \} \) with score \( s_a > s^*_1 \). Thus, \( s^*_n = 7 > a + 1 \). So there must be two scores in \( s \) larger than \( s_n \), precisely \( s_a < s_{a+1} < s_{a+2} \). To conclude we construct a sequence \( s' \) with \( G(s') > G(s) \) exactly as in the proof of Claim 4.8. \( \square \)

**Claim 4.10.** It holds \( s = c(n) \).

**Proof.** If \( n = 11 \), \( s \) has 5 vertices of score 3 and 5 vertices of score 7 by Claim 4.9. As, \( c(n) \) is the only such sequence not contradicting (4.2), the claim holds for \( n = 11 \). Similarly, \( c(n) \) is the only sequence not contradicting (4.1) and Claim 4.8 if \( 12 \leq n \leq 13 \). Suppose now that \( n \geq 14 \). There are \( n - 12 \) elements of \( s \) being different from both \( s^*_1 \) and \( s^*_n \), which have a score equal to one of the \( n - 8 \) numbers in the range \( 4, \ldots, n - 5 \). Symmetry of the map \( d \mapsto \binom{n}{d} \) around \( d = \frac{n}{2} \) together with (4.2) means that only pairs \( \{h_1, n - 1 - h_1\} \) with \( 4 \leq h_1 < \frac{n-1}{2} \) and \( \{h_2, n - 1 - h_2\} \) with \( 5 \leq h_2 < \frac{n-1}{2} \) of scores are missing in \( s \). Moreover, (4.1) requires \( h_1, h_2 < 7 \), for otherwise \( k = 8 \) violates this relation. Since \( s \) was chosen to be a maximizer of \( G \),
4.1. MINIMAL FEEDBACK VERTEX SETS IN TOURNAMENTS

this leaves $h_1 = 5$ and $h_2 = 6$. Thus $s = \sigma(n)$, completing the proof of the claim and of Lemma 4.4. □

All cases taken together imply the following upper bound on the number of maximal transitive subtournaments.

**Theorem 4.3.** Any strong tournament $T \in \mathcal{T}_n^*$ has at most $1.6740^n$ maximal transitive subtournaments.

**Corollary 4.5.** It holds $1.5448 \leq \lim_{n \to \infty} (M(n))^{1/n} \leq 1.6740$.

We conjecture that the Paley directed graph of order 7, $ST_7$, plays the same role for feedback vertex sets in tournaments as triangles play for independent sets in graphs, i.e. that the tournaments $T$ maximizing $(f(T))^{1/|V(T)|}$ are exactly those whose factors are copies of $ST_7$.

### 4.1.5 Polynomial-Delay Enumeration in Polynomial Space

We give a polynomial-space algorithm for the enumeration of the minimal feedback vertex sets in a tournament with polynomial delay.

Let $T = (V, A)$ be a tournament with $V = \{v_1, \ldots, v_n\}$, and for each $i = 1, \ldots, n$ let $T_i = T[\{v_1, \ldots, v_i\}]$. The algorithm enumerates the maximal acyclic vertex sets of $T$. It performs a depth-first search in a tree $T$ with the maximal acyclic vertex sets of $T$ as leaves, whose forward and backward edges are constructed “on the fly.” The depth of $T$ is $|V|$, and we refer to the vertices of $T$ as the root, nodes and children. The algorithm only needs to keep in memory the path from the root to the current node in the tree and all the children of the nodes on this path. Each node at level $j$ is labeled by a maximal acyclic vertex set $J$ of $T_j$. As for its children, there are two cases. In case $J \cup \{v_{j+1}\}$ is acyclic then $J$’s only child is $J \cup \{v_{j+1}\}$. In case $J \cup \{v_{j+1}\}$ is not acyclic then $J$ has at least one and at most $|j/2| + 1$ children. Let $L_j = (v^1, v^2, \ldots, v^{|L_j|})$ be a labeling of the vertices in $J$ such that $(v^r, v^s) \in A$ for all $1 \leq r < s \leq j$; we view $L_j$ as a sequence of vertices. The children of $J$ are as follows. The first child $J^0$ is a copy of $J$, and is always present. The potential other children are, for $1 \leq z \leq |J| + 1$,

$$J^z = \{v^i \in J \mid i < z \wedge (v^i, v_{j+1}) \in A\} \cup \{v_{j+1}\} \cup \{v^i \in J \mid i \geq z \wedge (v_{j+1}, v^i) \in A\}$$

where set $J^z$ is a potential child of $J$ only if $J^z$ is a maximal acyclic vertex set $T_j$.

Note how we try to insert $v_{j+1}$ at every possible position in $J$. However, only at most $|j/2| + 1$ positions make sense for $v_{j+1}$: before $v^1$ if $(v_{j+1}, v^1) \in A$, between $v^i$ and $v^{i+1}$ if $(v^i, v_{j+1}), (v_{j+1}, v^{i+1}) \in A$, where $1 \leq i \leq |J| - 1$, and after $v^{|L_j|}$ if $(v^{|L_j|}, v_{j+1}) \in A$; all other positions do not give maximal acyclic vertex sets and should not be generated in an actual implementation. Note that, in this case, $J^z$ is a potential child of several sets on the same level in $T$. Of all these sets, $J^z$ is made the child only of the lexicographically smallest such set. To determine this set, we compute by a greedy algorithm the lexicographically smallest maximal acyclic vertex set $H = H(J^z)$ of $T_j$ which contains $J^z \setminus \{v_{j+1}\}$ as a subset. That is, we iteratively build
the set \( H \) by setting
\[
H_0 = J^2 \setminus \{ v_{j+1} \},
\]
\[
H_i = \begin{cases} 
H_{i-1} \cup \{ v_i \}, & \text{if } H_{i-1} \cup \{ v_i \} \text{ is acyclic,} \\
H_{i-1}, & \text{otherwise,}
\end{cases} 
\]
\[
H = H_j. 
\]

Then we make \( f^* \) a child of the node labeled \( J \) only if \( H = J \). This completes the description of the algorithm.

To show that the algorithm is correct, we prove that for every maximal acyclic vertex set \( W \) of \( T \) there is exactly one leaf in \( T \) labeled with \( W \). By construction of the algorithm, it suffices to show that at least one leaf is labeled by \( W \). The proof is by induction on the number \( n = |V| \) of vertices in \( T \). For \( n = 1 \) the claim clearly holds, so suppose that \( n > 1 \) and that the claim is true for all tournaments with fewer vertices. Then from the induction hypothesis we can conclude that for the induced subtournament \( T' := T_{n-1} \) there is a tree \( T' \) constructed by the above algorithm and a bijection \( f' \) from the maximal acyclic vertex sets of \( T' \) to the leaves of \( T' \).

Let \( W \) be a maximal acyclic vertex set of \( T \). If \( v_n \notin W \) then \( W \) is an acyclic vertex set of \( T' \) as removing a vertex from a directed graph does not introduce cycles. In fact, \( W \) is a maximal acyclic vertex set of \( T' \): for any vertex \( v_{\ell} \in V \setminus (W \cup \{ v_n \}), T'[W \cup \{ v_{\ell} \}] \) has a cycle as \( W \) is a maximal acyclic vertex set for \( T \) and \( T'[W \cup \{ v_{\ell} \}] = T[W \cup \{ v_{\ell} \}] \). Hence there exists a leaf \( f'(W) \) in \( T' \) labeled by \( W \). Since \( W \cup \{ v_n \} \) is not acyclic, by maximality of \( W \) for \( T \), the algorithm constructs the child \( W_0 \) of \( f'(W) \) labeled by \( W \), and that child will be a leaf in the final tree constructed by the algorithm.

If \( v_n \in W \) then let \( W' = W \setminus \{ v_n \} \), so \( W' \) is an acyclic vertex set of \( T' \). In case \( W' \) is maximal for \( T' \) then there is a leaf \( f'(W') \) in \( T' \) that is labeled by \( W' \). Since \( W' \cup \{ v_n \} \) is acyclic, the algorithm will create a single child of \( f'(W') \) labeled by \( W' \cup \{ v_n \} = W \), and that child will be a leaf in the final tree constructed by the algorithm. In case \( W' \) is not maximal for \( T' \), let \( N \) be the lexicographically smallest extension of \( W' \) to a maximal acyclic vertex set of \( T' \). Hence there exists a leaf \( f'(N) \) in the tree \( T' \) labeled by \( N \). Observe that the sequence \( L_{W'} \) is a subsequence of \( L_N \), and that \( N \cup \{ v_n \} \) is not acyclic. Hence the algorithm creates children \( N^1, N^2, \ldots \), one of which will be labeled by \( W \).

To see that the algorithm runs with polynomial delay, note that the children and parent of a given node in \( T \) can all be computed in polynomial time. It follows that \( T \) can be traversed in a depth-first manner with polynomial delay per step of the traversal, and thus the leaves of \( T \) can be output with only a polynomial delay.

We show that the algorithm requires only polynomial space. We already observed that each node in \( T \) at level \( j \) has at most \( \lceil j/2 \rceil + 1 \) children. For each node we store the maximal acyclic vertex set by which it is labeled. Because we are traversing \( T \) in a depth-first-search manner, in each step of the algorithm we only need to save data of \( O(n^2) \) nodes: those of the \( O(n) \) nodes on the path from the root to the currently active node labeled by \( J \), and the \( O(n) \) children for each node on this path.
4.2. MINIMUM FEEDBACK VERTEX SET IN BIPARTITE DIRECTED GRAPHS

Theorem 4.4. The described algorithm enumerates all feedback vertex sets of a tournament with polynomial delay and uses polynomial space.

Corollary 4.6. In a tournament with \( n \) vertices a minimum directed feedback vertex set can be found in \( O(1.6740^n) \) time and polynomial space.

4.2 Minimum Feedback Vertex Set in Bipartite Directed Graphs

In this section we give an algorithm for finding a minimum feedback vertex set in a bipartite directed graph in time \( O(1.8621^n) \). The analysis of the running time of this algorithm uses bounds on the number of vertex subsets of small size. Our algorithm is inspired by an algorithm for finding a minimum feedback vertex set in undirected bipartite graphs, by Fomin and Pyatkin [113].

We begin to relate minimum feedback vertex sets in general directed graphs to independent sets.

Lemma 4.5. Let \( D \) be a directed graph, let \( I \subseteq V(D) \) be an independent set of \( D \) and let \( J = V(D) \setminus I \). Let \( F \) be a minimum feedback vertex set of \( D \) such that \( |F \cap J| \) is maximum, and let \( C \subseteq I \) be the set of vertices in \( I \) adjacent to at least three vertices in \( J \setminus F \). Then

1. \( F \cap I \subseteq C \),
2. \( |C \setminus X| \leq \frac{1}{2}|J \setminus F| \) and \( |C| \leq \frac{3}{2}|J \setminus F| \).

Proof. (1) Let \( u \) be a vertex in \( F \cap I \). Then \( u \) is adjacent to at least two vertices of \( J \setminus F \), for otherwise the set \( F \setminus \{u\} \) would be a feedback vertex set of \( D \) of size smaller than \( |F| \). If \( u \) is adjacent to exactly two vertices \( v, w \) in \( J \setminus F \) then the set \( F' = F \setminus \{u\} \cup \{v\} \) is a minimum feedback vertex set of \( D \) with \( |F' \cap J| > |F \cap J| \), a contradiction. It follows that \( F \cap I \subseteq C \).

(2) Observe that \( J \setminus F \) is an acyclic vertex set of \( D \) with at least \( 3|C \setminus F| \) arcs. Hence,

\[
|C \cap F| = |F| - |F \setminus C| \\
\leq |J| - |F \setminus C| \\
= |J \setminus F| + |J \cap F| - |F \setminus C| = |J \setminus F|,
\]

where the last equality follows from (1). The set \( (J \setminus F) \cup (C \setminus F) \) is an acyclic vertex set of \( D \) with at least \( 3|C \setminus F| \) arcs. Thence, \( 3|C \setminus F| \leq |J \setminus F| + |C \setminus F| \), and it follows that \( |C \setminus F| \leq \frac{3}{2}|J \setminus F| \) and \( |C| \leq \frac{3}{2}|J \setminus F| \).

The lemma leads to a fast algorithm finding a minimum feedback vertex set in directed graphs with large independent sets.

Lemma 4.6. Given a directed graph \( D \) on \( n \) vertices and an independent set \( I \) of \( D \), a minimum feedback vertex set of \( D \) can be found in time \( O^*(1.6740^n) \).

Proof. Let \( J = V(D) \setminus I \). We describe a procedure that finds a minimum feedback vertex set \( F \) of \( D \) such that \( |F \cap J| \) is maximum:
(1) We guess the set $I \setminus F$, by considering all subsets $J \subseteq I$. If $D[J]$ contains a cycle then $J'$ is not a candidate for $I \setminus F$. Otherwise, then we augment the directed graph $D[J']$ to a directed graph $D'$ by adding, for every vertex $u \in I$ that has exactly one in-neighbor $v \in J'$ and exactly one out-neighbor $w \in J'$, the arc $(v, w)$. If $D'$ contains a cycle then $J'$ is not a candidate for $I \setminus F$, by Lemma 4.5(1). If $D'$ does not contain cycles and the set $C(J') \subseteq I$ of vertices in $I$ that are adjacent to at least three vertices of $J'$ satisfies $|C(J')| > \frac{3}{2}|J'|$ then $J'$ is not a candidate for $I \setminus F$, by Lemma 4.5(2).

(2) For each candidate set $J'$ of Step (1), we enumerate all subsets $J'' \subseteq C(J')$ that satisfy $|J''| \leq \frac{1}{2}|J'|$. The restriction on the size of $J''$ is justified by Lemma 4.5(2). If the set $J' \cup J''$ is an acyclic set in $D$ then we add the set $F := J \setminus J' \cup C(J') \setminus J''$ to a list $\mathcal{L}$ of potential minimum feedback vertex sets of $D$.

(3) Return an element $F$ of $\mathcal{L}$ with minimum cardinality; it is a minimum feedback vertex set of $D$ for which $|F \cap J|$ is maximum.

Correctness of the procedure follows from the fact that the list $\mathcal{L}$ only contains sets $F$ satisfying the conditions of Lemma 4.5.

For the running time of the procedure, observe the following. Step (1) requires enumerating $2^{n-|I|}$ sets $J' \subseteq I$. Step (2) considers, for every candidate set $J'$, at most
\[
\binom{3|J'|/2}{|J'|/2} = O^*(\left(\frac{3}{2}\right)^{3|J'|/2})
\]
subsets $J'' \subseteq J'$. Step (3) requires time that is within a polynomial factor of the time required by Step (2). Thus, the running time of the procedure is bounded by
\[
\sum_{j=0}^{n-|I|} \binom{n-|I|}{j}^{i/2} \sum_{i=0}^{3/2} \binom{3/2}{i} n^{O(1)} = O^* \left( \sum_{j=0}^{n-|I|} \binom{n-|I|}{j} \left( \frac{3}{2} \right)^{(3/2)^{3/2}} \right) = O^* \left( \sum_{j=0}^{n-|I|} \binom{n-|I|}{j} \left( \frac{3}{2} \right)^{3/2} \right) = O^* \left( \left( 1 + \frac{3}{2} \sqrt{3} \right)^{n-|I|} \right). \quad \Box
\]

Since every bipartite directed graph $D$ on $n$ vertices has an independent set of size $n/2$, and such can be found in polynomial time, we conclude that a minimum feedback vertex set of $D$ can be found in time $O^*((1 + \frac{3}{2} \sqrt{3})^{n/2}) = O^*(1.8968^n)$.

We now improve this time, by using that a bipartite graph has two large independent sets that are vertex-disjoint.

**Theorem 4.5.** For bipartite directed graphs on $n$ vertices, a minimum feedback vertex set can be found in time $O(1.8621^n)$.

**Proof.** Let $D = (I_1 \cup I_2, A)$ be a bipartite directed graph with independent sets $I_1$ and $I_2$. Assume, without loss of generality, that $|I_1| \leq n/2$. We give a procedure
resembling the one in the proof of Lemma 4.6. The procedure starts by executing the following steps for $h = 1, 2$.

1. Enumerate all subsets $F_h' \subseteq F_h$ of size at most $|F_h|/2$ if $h = 1$ and of arbitrary size if $h = 2$. Augment the directed graph $D[F_h']$ to a directed graph $D'$ by adding, for every vertex $w \in F_{(h+1 \mod 2)}$ that has exactly one in-neighbor $u \in F_h'$ and exactly one out-neighbor $v \in F_h'$, the arc $(u, v)$. If $D'$ contains a cycle then $F_h'$ is not a candidate. If $D'$ does not contain cycles then let $C(F_h') \subseteq F_{(h+1 \mod 2)}$ be those vertices that are adjacent to at least three vertices of $F_h'$.

   a. If $h = 1$ and $|C(F_h')| > \frac{3}{2} |F_h'|$ then $F_h'$ is not a candidate.
   b. If $h = 2$ and $|C(F_h')| > \frac{1}{2} (|F_h'| + |F_0|)$ then $F_h'$ is not a candidate.

2. For each candidate set $F_h'$ of Step (1), enumerate all subsets $F_h'' \subseteq C(F_h')$ that satisfy $|F_h''| \leq \frac{1}{4} |F_h'|$. If the set $F_h'' = F_h' \cup C(F_h') \setminus F_h''$ is an acyclic set in $D$ then we add the set $F := F_h \setminus F_h' \cup C(F_h') \setminus F_h''$ to a list $\mathcal{L}$ of potential minimum feedback vertex sets of $D$.

The final step of the procedure is

3. Return an element $F$ of $\mathcal{L}$ with minimum cardinality; it is a minimum feedback vertex set of $D$.

For correctness of the procedure, for $h = 1, 2$ let $F_h$ be a minimum feedback vertex of $D$ such that $|F_h \cap J_0| = \max$. If $|F_h \cap J_0| \geq |J_1|/2$ then $F_1$ is found by executing Steps (1) and (2) for $h = 1$, and Step (3) thereafter. Whereas if $|F_2 \cap J_1| \leq |J_1|/2$ then $F_2$ is found by executing Steps (1) and (2) for $h = 2$, and Step (3) thereafter.

For the running time of the procedure, observe the following. For $h = 1$, Steps (1) and (2) require time that is bounded by

$$\sum_{j=0}^{|J_1|/2} \binom{|J_1|}{j} \left( \sum_{i=0}^{|J_1|/2} \binom{3j/2}{j} \right) n^{O(1)} = O^* \left( \sum_{j=0}^{|J_1|/2} \binom{|J_1|}{j} \left( \frac{3\sqrt{3}}{2} \right)^2 \right)$$

$$= O^* \left( 2^{|J_1|} \left( \frac{3\sqrt{3}}{2} \right)^{|J_1|/2} \right) = O^* \left( 108^{1/4} |J_1|/4 \right).$$

For $h = 2$, Steps (1) and (2) require time that is bounded by

$$\sum_{j=0}^{|J_2|/2} \binom{|J_2|}{j} \left( \sum_{i=0}^{|J_2|/2} \binom{1/2 |J_1| + j}{i} \right) n^{O(1)} = O^* \left( \sum_{j=0}^{|J_2|/2} \binom{|J_2|}{j} 2^{(|J_1|+j)/2} \right)$$

$$= O^* \left( 2^{|J_1|/2} (1 + \sqrt{2}) |J_2| \right).$$

Thus, the running time of the procedure is within a polynomial factor of

$$\max \left\{ 108^{1/4}, 2^{1/2} (1 + \sqrt{2}) \right\}.$$

(4.7)
The claimed running time $O(1.8621^n)$ follows from (4.7) if $|J_1| \geq 0.4856n$, and follows from Lemma 4.6 otherwise. □

4.3 Induced Matchings in Planar Graphs with Maximum Degree Three

In this section we study the parameterized complexity of finding an induced matching of size at least $k$ in planar graphs with maximum degree three.

Recall that for a graph $G$, an induced matching is a set $M \subseteq E(G)$ of edges such that a shortest path in $G$ between any two edges in $M$ has length at least two. The problem of computing the size of a largest induced matching was introduced by Stockmeyer and Vazirani [232] as a variant of the maximum matching problem. They motivated it as the “risk-free” marriage problem: find the maximum number of married couples such that no married person is compatible with a married person other than her/his spouse.

Our focus in this section are planar graphs of maximum degree 3, where the maximum induced matching problem is NP-hard [88]. (The proof of this fact by Ko and Shepherd [176] contains an error.) For general graphs $G$, deciding whether $G$ has an induced matching of size $k$ is W[1]-hard [200]. In planar graphs and graphs of bounded maximum degree, Moser and Sikdar [199] showed this problem to be fixed-parameter tractable. Notably, by examining a greedy algorithm, they showed that for graphs of maximum degree at most 3 there exists a kernel with at most $26k$ vertices. For twinless planar graphs, lower bounds on the size of a maximum induced matchings by Kanj et al. [167] and Erman et al. [89] led to a kernel with $28k$ vertices.

We provide a result similar in spirit to the aforementioned results. In particular, we promote the use of a structural approach to derive kernel size bounds for planar graph classes. Our main result in this section relies on graph properties proven using a discharging procedure.

**Theorem 4.6.** Every planar graph $G$ of maximum degree 3 has an induced matching of size at least $|E(G)|/9$, and such a matching can be found in time $O(|V(G)|)$.

**Corollary 4.7.** Every 3-regular planar graph $G$ has an induced matching of size at least $|V(G)|/6$.

This bound is stronger than the bound $3|V(G)|/22$ for 3-regular graphs $G$ that is given by Zito [253]. In particular, our lower bounds on the size of maximum induced matchings in 3-regular planar graphs and planar graphs with maximum degree three are best possible: consider the disjoint union of multiple copies of the triangular prism.

The condition on maximum degree in Theorem 4.6 cannot be weakened: the disjoint union of multiple copies of the octahedron is a 4-regular planar graph $G$ that has no induced matching with more than $|E(G)|/12$ edges. Also, the condition on planarity in cannot be dropped: the disjoint union of multiple copies of the graph in Figure 4.4 is a graph $G$ of maximum degree 3 that has no induced matching with more than $|E(G)|/10$ edges.
4.3. INDUCED MATCHINGS IN PLANAR GRAPHS

Figure 4.4: A subcubic graph with no induced matching of size 2.

Theorem 4.6 implies that the problem of determining if a planar graph \( G \) of maximum degree three has an induced matching of size at least \( k \) has a kernel of size at most \( 9k \). This follows from the fact that if \( k \leq \frac{|E(G)|}{9} \) then \((G, k)\) is a “yes”-instance, and we can produce an appropriate matching by way of Theorem 4.6; otherwise, \(|E(G)| < 9k\) and we have obtained a kernel with fewer than \( 9k \) edges. Similarly, for 3-regular planar graphs, Corollary 4.7 implies a kernel with at most \( 6k \) vertices.

Considerable interest in induced matchings is due to a close connection with strong edge-colorings. Since a strong \( k \)-edge-coloring of \( G \) partitions the edges of \( G \) into \( k \) induced matchings, a maximum induced matching in \( G \) has size at least \( \frac{|E(G)|}{k} \). Thus, Theorem 4.6 relates to the long-standing Erdős-Nešetřil conjecture, see Faudree et al. [92] and Chung et al. [60]. Our result lends support to Conjecture 4 of a webpage on the strong chromatic index by Douglas West [245], claiming that every planar graph of maximum degree 3 has a strong 9-edge-coloring. This conjecture has an earlier origin: it is implied by one case of a thirty-year-old conjecture of Wegner [243], asserting that the square of a planar graph with maximum degree 4 has a proper 9-vertex-coloring. For that, observe that the line graph of a planar graph with maximum degree 3 is a planar graph with maximum degree 4. Andersen [23] demonstrated that every graph with maximum degree three has a strong 10-edge-coloring, which implies that every graph \( G \) of maximum degree three has an induced matching of size at least \( \frac{|E(G)|}{10} \).

4.3.1 A Linear-time Algorithm

We present an algorithm that, given a planar graph \( G \) with maximum degree three, finds an induced matching of size at least \( \frac{|E(G)|}{9} \) in \( G \).

Let us first introduce some notions. Throughout this section, let \( G \) be a planar graph with maximum degree three. Notice that if an edge set \( M \subseteq E(G) \) is an induced matching in \( G \) then

\[ \text{dist}(e, f) \geq 2 \quad \text{for all distinct} \quad e, f \in M. \]

For each subset \( E' \subseteq E(G) \) define \( \Psi(E') = \{ e \in E \mid \text{dist}(e, E') < 2 \} \), and call \( E' \) good if \( E' \) is an induced matching, \( 1 \leq |E'| \leq 5 \) and \( |\Psi(E')| \leq 9|E'| \).

We need the following observation.

**Lemma 4.7.** If \( E' \subseteq E(G) \) is minimally good then \( 2 \leq \text{dist}(e, f) \leq 15 \) for all distinct \( e, f \in E' \).

**Proof.** That \( \text{dist}(e, f) \geq 2 \) for all distinct \( e, f \in E' \) was already observed above. Note that no subset \( E'' \subseteq E' \) exists with \( \text{dist}(e, f) \geq 4 \) for all \( e \in E'' \) and \( f \in E' \setminus E'' \), for otherwise \( \Psi(E'') \cap \Psi(E' \setminus E'') = \emptyset \). This, however, implies that \( |\Psi(E')| = |\Psi(E'')| + |\Psi(E' \setminus E'')| \).
have computed a matching $M$ this until $H$ queue $Q$ linear time \[194, \text{Exercise 8.1}\].

If a graph is stored as a list, then it is possible to create a neighbor-list array in which each entry for a vertex stores the labels of the preceding and succeeding element. If a graph is stored as a neighbor-list array, which is an array with one entry for each vertex in a variable holding the first element of $Q$ and set $H = G$. We then repeatedly find a minimally good set $E'$ in $H$ and set $M := M \cup E'$ and $H := H \setminus \Psi(E')$. The theorem guarantees that we can keep doing this until $H$ is the empty graph. By definition of a good set of edges, we will then have computed a matching $M$ of size $|E(G)|/9$.

Apart from proving Theorem 4.7, it also remains to show that this greedy procedure can be implemented in linear time. Our algorithm assumes that the graph is stored as a neighbor-list array, which is an array with one entry for each vertex and a list with the labels of its (up to three) neighbors. If a graph is stored as a list of edges or an adjacency matrix then it is possible to create a neighbor-list array in linear time [194, Exercise 8.1].

The algorithm examines the vertices one after another in some order given by a queue $Q$. We store $Q$ by means of (1) a variable holding the first element of $Q$, (2) a variable holding the last element, and (3) an array with $|V(G)|$ entries, where this time the entry for vertex $v$ stores the labels of the preceding and succeeding element in $Q$. This ensures that the operations of deleting an arbitrary element from $Q$, and inserting elements in the first or last positions all take constant time.

As long as $Q$ is non-empty, the algorithm repeats the following steps. Let $v$ denote the first element of $Q$.

(1) If $v$ is a 0-vertex then remove $v$ from $Q$.

(2) If $v$ is an ($\geq 1$)-vertex then check whether there is a minimally good set of edges $E'$ such that $v$ is the endpoint of some edge $e \in E'$.

(2.1) If such an $E'$ does not exist, then we move $v$ to the back of the queue.

(2.2) If $E'$ does exist, then we set $M := M \cup E'$, $H := H \setminus \Psi(E')$, and we put the vertices of $N^{20}(E')$ in the front of $Q$ in an arbitrary order.

Checking for a suitable set $E'$ in Step (2) can be done in constant time: By Lemma 4.7 we only need to consider sets $E'$ of up to 5 edges such that each edge $e \in E'$ has at least one endpoint at distance at most 16 from $v$. Hence all vertices incident with
4.3. INDUCED MATCHINGS IN PLANAR GRAPHS

edges of $\Psi(E')$ will be within distance at most 18 of $v$. Thus, to find a minimally good set $E'$ with at least one edge incident to $v$, we only need to examine the subgraph $H[N^{18}[v]]$ of $H$ induced by all vertices at distance at most 18 from $v$. This subgraph has at most $3 \cdot 2^{17} = O(1)$ vertices, and it can be computed in constant time from the neighbor-list array data structure that $H$ is stored in. (We read in constant time which are the neighbors of $v$, then in constant time which are the neighbors of the neighbors and so on until depth 18.) Since $H[N^{18}[v]]$ has constant size, we can clearly find a set $E'$ of the required form in constant time, if such an $E'$ exists.

Let $N^{20}(E')$ be the set of vertices in $G$ that have distance at most 20 to some vertex of some edge in $E'$. By moving $N^{20}(E')$ to the front of $Q$ in Step (2.1), we make sure that vertices $u$ for which $H[N^{18}[u]]$ has been affected will be examined before other vertices. (When we remove $\Psi(E')$ from $H$ then $H[N^{18}(u)]$ can only be affected if $u \in N^{20}(E')$.) Note that we can again find the vertices of $N^{20}(E')$ in constant time. Also note that removing the edge $\{u,v\}$ from the neighbor-list data structure can be done in constant time, since we just need to update the entries of the array for $u$ and $v$. Hence removing $\Psi(E')$ from $H$ can also be done in constant time.

We finalize the discussion of the algorithm.

**Theorem 4.8.** Given a planar graph $G$ with maximum degree three, the algorithm computes a matching of size at least $|E(G)|/9$ in time $O(|V(G)|)$.

**Proof.** Correctness of the greedy procedure follows from Theorem 4.7. Moreover, by previous discussions we know that each of the Steps (1), (2), (2.1) and (2.2) of the algorithm takes only $O(1)$ operations. It therefore suffices to show that these steps are iterated at most $O(|V(G)|)$ times. We will prove this by showing that each vertex $u$ occurs only $O(1)$ times as the first element of $Q$.

Let us first observe that a vertex $u$ is moved to the front of $Q$ at most $3 \cdot 2^{19}$ times. This is because at each iteration where $u$ was moved to the front of $Q$ some minimally good set $E'$ was found with $u \in N^{20}(E')$. Hence at least one edge in $H[N^{20}(u)]$ was removed, namely the edge of $\Psi(E')$ closest to $u$. Since $H[N^{20}(u)]$ has at most $3 \cdot 2^{19}$ edges, it must indeed hold that $u$ cannot have been moved to the front of $Q$ more than $3 \cdot 2^{19}$ times.

Next, we claim that a vertex $u$ is moved to the back of $Q$ at most $3 \cdot 2^{19} + 1$ times. To see this, let $u$ be the first element of $Q$ and suppose that it was also the first element at some past iteration. Suppose further that at the last iteration when $u$ was the first element of $Q$, it got sent to the back and that it did not get sent to the front of $Q$ in any later iteration. Let us write $Q = (v_0, v_1, \ldots, v_k)$ where $v_0 = u$. Then $v_1, \ldots, v_k$ must have been examined after $u$, and no minimally good set $E'$ was found in each case so that they were sent to the back of $Q$. All other vertices of $G$ must have been deleted already because they became isolated at some point. Moreover, for $l = 0, \ldots, k$, no edge of $H[N^{18}[v_l]]$ was deleted after $v_l$ was examined for the last time: otherwise a minimally good set $E'$ would have been found in the mean time for which $\Psi(E')$ hits $H[N^{18}[v_l]]$, but then we would also have $v_l \in N^{20}(E')$ and $v_l$ would thus have been sent to the front of $Q$. Since we can determine whether there is a minimally good set $E'$ with $v \in e$ for some $e \in E'$ from $H[N^{18}[v]]$ alone, there can be no minimally good set $E'$ in $H$ at all. But this
4. ALGORITHMS THROUGH EXTREMAL COMBINATORICS

contradicts Theorem 4.7.

We have thus just shown that, if a vertex \( u \) is sent to the back of \( Q \), then by the next time we encounter it as the first element of the queue it has been sent to the front of \( Q \) at least once. This implies that the number of times that \( u \) was sent to the back of \( Q \) is at most one more than the number of times that \( u \) is sent to the front of \( Q \), as claimed. The additional plus one arises because the first time we encounter \( u \) we may send it to the back of \( Q \).

Finally, observe that the number of times vertex \( u \) occurs as the first element of \( Q \) is at most the number of times that \( u \) gets sent to the back of \( Q \) plus the number of times that \( u \) gets sent to the front of \( Q \), plus 1. The additional plus one arises when \( u \) is isolated and gets deleted from \( Q \). By our previous observations, this adds up to at most \( 2 \cdot 3 \cdot 2^{19} + 2 = O(1) \), which concludes the proof of Theorem 4.8. □

4.3.2 The proof of Theorem 4.7

Theorem 4.7 is a direct consequence of the following two lemmas.

Lemma 4.8. Let \( G \) be a plane graph of maximum degree three. If \( G \) contains one of the following structures (C1)–(C12) then \( G \) contains a good set of edges.

(C1) A 1-vertex.

(C2) A 2-vertex incident to an \((\leq 7)\)-cycle.

(C3) A 2-vertex at distance at most 2 from a 2-vertex.

(C4) A 2-vertex at distance at most 2 from an \((\leq 5)\)-cycle.

(C5) A 3-cycle in sequence with an \((\leq 6)\)-cycle or a 7-face.

(C6) A 4- or 5-cycle in sequence with a 5- or 6-cycle.

(C7) A 3-cycle at distance 1 from an \((\leq 5)\)-cycle.

(C8) A double 4-face adjacent to an \((\leq 7)\)-cycle.

(C9) A 4-cycle, \((\leq 8)\)-cycle and 4-cycle in sequence.

(C10) A 4-cycle, 7-cycle and 5-cycle in sequence.

(C11) A 3-cycle or double 4-face at distance at most 2 from a 3-cycle or double 4-face.

(C12) A double 4-face at distance 1 from a 5-cycle.

Proof. We prove Lemma 4.8, by analyzing the structures in order, including some intermediate structures. The order of our analysis is significant. The proofs for later structures rely in part on our analysis of earlier structures. For certain structures, we consider several cases. Whenever we are inside a given case, we presume that none of the hypotheses of a preceding case applies.

We give a figure for each structure. For these, we use a visual code: a square represents a \((\leq 2)\)-vertex, a circle represents a \((\leq 3)\)-vertex, a thin solid line represents a present edge, and a bold solid line represents an edge of an induced matching.
Claim 4.11. If \( G \) has a 1-vertex then it contains a good set of edges.

Proof. For a 1-vertex \( u \) with neighbor \( v \) the set \( E' = \{ \{ u, v \} \} \) is good, as \( |\Psi(E')| \leq 7 \). See Fig. 4.5(a).

Claim 4.12. If \( G \) has adjacent 2-vertices then it contains a good set of edges.

Proof. For adjacent 2-vertices \( u, v \) the set \( E' = \{ \{ u, v \} \} \) is good, since \( |\Psi(E')| \leq 7 \). See Fig. 4.5(b).

Claim 4.13. If \( G \) has a 2-vertex on a 3-cycle then it contains a good set of edges.

Proof. For a 2-vertex \( u \) on a 3-cycle \((u, v, w)\) the set \( E' = \{ \{ u, w \} \} \) is good, since \( |\Psi(E')| \leq 7 \). See Fig. 4.5(c).

Claim 4.14. If \( G \) has a 2-vertex on a 4-cycle then it contains a good set of edges.

Proof. For a 2-vertex \( u \) on a 4-cycle \((u, v, w, x)\) the set \( E' = \{ \{ u, x \} \} \) is good, since \( |\Psi(E')| \leq 9 \). See Fig. 4.5(d).

Claim 4.15. If \( G \) has two 2-vertices at distance two then it contains a good set of edges.

Proof. Let \( u, w \) be 2-vertices at distance two. Let \( N(u) \cap N(w) = \{ v \} \), and let \( N(u) \setminus N(v) = \{ u' \} \) and \( N(w) \setminus N(u) = \{ w' \} \). Notice that by Claims 4.13 and 4.14, vertices \( u, u', w, w' \) are distinct. Since \( v \) is a 3-vertex, by Claim 4.12, it has a neighbor \( v' \not\in \{ u, w \} \). Because \( G \) has no 2-vertex on a \((\leq 4)\)-face by Claims 4.13 and 4.14, \( v' \) is distinct from \( u' \) and \( w' \), and neither \( \{ u', v' \} \) nor \( \{ w', v' \} \) is an edge. Consider the following cases, see Fig. 4.6.

1. If \( u' \) and \( w' \) are adjacent then \( E' = \{ \{ u', w' \}, \{ v, v' \} \} \) is a good set, since \( |\Psi(E')| \leq 18 \).

2. Otherwise, \( E' = \{ \{ u, u' \}, \{ w, w' \} \} \) is a good set, since \( |\Psi(E')| \leq 17 \).

Claim 4.16. If \( G \) has a 2-vertex on a 5-cycle then it contains a good set of edges.

Proof. Let \( u \) be a 2-vertex on a 5-cycle \( C = (u, v_1, v_2, v_3, v_4) \). Observe that \( v_1, v_2, v_3, v_4 \) are 3-vertices, by Claims 4.12 and 4.15. By Claims 4.13 and 4.14, \( C \) has no chords. For \( i = 1, 2, 3 \), let \( N(v_i) \setminus \{ v_1, \ldots, v_4 \} = \{ v'_i \} \). By Claim 4.14, \( v'_1 \neq v'_4 \). Consider the following cases, see Fig. 4.7(a).
Claim 4.17. If \( G \) has a 2-vertex on a 6-cycle then it contains a good set of edges.

**Proof.** Let \( u \) be a 2-vertex on a 6-cycle \((u, v_1, v_2, v_3, v_4, v_5)\). By Claims 4.14 and 4.16, neither \( \{v_1, v_3\} \) nor \( \{v_1, v_4\} \) can be an edge. Then \( E' = \{\{u, v_1\}, \{v_3, v_4\}\} \) is a good set, since \( |\Psi(E')| \leq 17 \). See Fig. 4.7(b). \( \square \)

Claim 4.18. If \( G \) has a 2-vertex at distance one from a 3-cycle then it contains a good set of edges.

**Proof.** Let \( u \) be a 2-vertex at distance one from a 3-cycle \((x, y, z)\), where \( \{u, x\} \) is an edge. Then \( E' = \{\{u, x\}\} \) is a good set, since \( |\Psi(E')| \leq 9 \). See Fig. 4.8(a). \( \square \)
Claim 4.19. If $G$ has a 2-vertex at distance one from a 4-cycle then it contains a good set of edges.

Proof. Let $u$ be a 2-vertex at distance one from a 4-cycle $(x_0, x_1, x_2, x_3)$, where $\{u, x_0\}$ is an edge. By Claim 4.14, $u$ is not adjacent to $x_2$. Observe that $x_2$ is a 3-vertex, by Claim 4.14. Let $N(x_2) \setminus \{x_1, x_3\} = \{x'_2\}$. Then $u$ is not adjacent to $x'_2$ by Claim 4.16. Then $E' = \{\{u, x_0\}, \{x_2, x'_2\}\}$ is a good set, since $|\Psi(E')| \leq 17$. See Fig. 4.8(b).

Claim 4.20. If $G$ has a 2-vertex at distance one from a 5-cycle then it contains a good set of edges.

Proof. Let $u$ be a 2-vertex at distance one from a 5-cycle $(x_0, x_1, x_2, x_3, x_4)$, where $\{u, x_0\}$ is an edge. Let $N(u) \setminus \{x_0\} = \{y\}$, then $y \notin \{x_2, x_3\}$ by Claim 4.14. Then $E' = \{\{u, x_0\}, \{x_2, x_3\}\}$ is a good set, since $|\Psi(E')| \leq 17$. See Fig. 4.8(c).

Claim 4.21. If $G$ has a 2-vertex on a 7-face then it contains a good set of edges.

Proof. Let $u$ be a 2-vertex on a 7-face $C = (u, v_1, v_2, v_3, v_4, v_5, v_6)$. By Claims 4.12 and 4.15, $v_1, v_2, v_5, v_6$ are 3-vertices. For $i = 1, 2, 5, 6$, let $N(v_i) \setminus \{u, v_1, \ldots, v_6\} = \{v'_i\}$. By Claims 4.13, 4.14, 4.16 and 4.17, $C$ has no chords. Consider the following cases, see Fig. 4.9.

1. If $v_4$ is a 2-vertex then $v_3$ is a 3-vertex. By Claim 4.16, $v'_4 \neq v'_5$. Then $E' = \{\{u, v_1\}, \{v_4, v_5\}\}$ is a good set, since $|\Psi(E')| \leq 16$. The case that $v_3$ is a 2-vertex is treated symmetrically.

For $i = 3, 4$, let $N(v_i) \setminus \{v_2, \ldots, v_3\} = \{v'_i\}$.

2. If $v'_3$ and $v'_4$ are adjacent or $v'_3 = v'_4$, then since $C$ is a 7-face, it follows from the Jordan Curve Theorem that $v'_3 \neq v'_5$. Then $E' = \{\{u, v_1\}, \{v_3, v'_4\}, \{v_5, v'_5\}\}$ is a good set, since $|\Psi(E')| \leq 27$. The case in which $v'_3$ and $v'_4$ are adjacent or $v'_3 = v'_5$ is handled similarly.

3. Now note that $v'_3 \neq v'_4$ due to Claim 4.19. Then $E' = \{\{u, v_1\}, \{v_3, v'_4\}, \{v_5, v'_5\}\}$ is a good set, since $|\Psi(E')| \leq 27$.

Claim 4.22. If $G$ has a pair of adjacent 3-cycles then it contains a good set of edges.

Proof. Let $(x, y, z)$ and $(x, y, z')$ be adjacent 3-cycles. Then $E' = \{\{x, y\}\}$ is a good set, since $|\Psi(E')| \leq 7$. See Fig. 4.10(a).
Claim 4.23. If \( G \) has a 3-cycle adjacent to a 4-cycle then it contains a good set of edges.

Proof. Let \((x, y, z)\) be a 3-cycle sharing the edge \(\{x, y\}\) with a 4-cycle \((x, u, v, y)\). Observe that the two cycles have no other edges in common, by Claim 4.22. Then \(E' = \{\{x, y\}\}\) is a good set, since \(|\Psi(E')| \leq 9\). See Fig. 4.10(b).

Claim 4.24. If \( G \) has a 3-cycle adjacent to a 5-cycle then it contains a good set of edges.

Proof. Let \((x, y, z)\) be a 3-cycle sharing the edge \(\{x, y\}\) with a 5-cycle \((x, u, v, w, y)\). Observe that \(v\) is a 3-vertex, by Claim 4.16. Let \(N(v) \setminus \{u, w\} = \{v'\}\). Further notice that \(\{x, y\}\) is the only edge common to both cycles, by Claim 4.23. Then \(E' = \{\{x, y\}, \{v, v'\}\}\) is a good set, since \(|\Psi(E')| \leq 17\). See Fig. 4.10(c).

Claim 4.25. If \( G \) has a 3-cycle adjacent to a 6-cycle then it contains a good set of edges.

Proof. Let \((x, y, z)\) be a 3-cycle sharing the edge \(\{x, y\}\) with a 6-cycle \((x, v_1, v_2, v_3, v_4, y)\). Notice that \(\{x, y\}\) is the only edge common to both cycles, by Claim 4.24. Then \(E' = \{\{x, y\}, \{v_2, v_3\}\}\) is a good set, since \(|\Psi(E')| \leq 17\). See Fig. 4.10(d).

Claim 4.26. If \( G \) has a 3-cycle adjacent to a 7-cycle then it contains a good set of edges.

Proof. Let \((x, y, z)\) be a 3-cycle sharing the edge \(\{x, y\}\) with a 7-cycle \(C = (x, v_1, v_2, v_3, v_4, v_5, y)\). Notice that \(\{x, y\}\) is the only edge common to both cycles, by Claim 4.25. By Claims 4.23 and 4.24, \(C\) has no chords. Consider the following cases, see Fig. 4.11.

1. If \(v_2\) is a 2-vertex then \(E' = \{\{x, y\}, \{v_2, v_3\}\}\) is a good set, since \(|\Psi(E')| \leq 16\). The case that \(v_4\) is a 2-vertex is treated symmetrically.
4.3. INDUCED MATCHINGS IN PLANAR GRAPHS

For \( i = 2, 4 \) let \( N(v_i) \setminus \{ v_{i-1}, v_{i+1} \} = \{ v'_i \} \).

(2) If \( v'_i \) and \( v'_4 \) are adjacent then by Claim 4.25, \( v_1 \) and \( v'_4 \) are not adjacent. The set \( E' = \{ \{ v_1, v_2 \}, \{ v_4, v'_4 \}, \{ y, z \} \} \) is good, since \( |\Psi(E')| \leq 24 \).

(3) If \( v'_i = v'_4 \) then \( E' = \{ \{ v_3, v_4 \}, \{ x, y \} \} \) is a good set, since \( |\Psi(E')| \leq 18 \).

(4) The set \( E' = \{ \{ v_2, v'_4 \}, \{ v_4, v'_4 \}, \{ x, y \} \} \) is good, since \( |\Psi(E')| \leq 27 \). \( \square \)

**Claim 4.27.** If \( G \) has a pair of 4-cycles joined along two incident edges then it contains a good set of edges.

**Proof.** Let \( (x_0, x_1, x_2, x_3) \) and \( (x_0, x_1, x_2, x_4) \) be 4-cycles joined along edges \( \{ x_0, x_1 \} \) and \( \{ x_1, x_2 \} \). Suppose that in the embedding of the graph, the vertex \( x_1 \) is in the interior of the curve formed by the cycle \( (x_0, x_3, x_2, x_4) \). Observe that \( x_1, x_3, x_4 \) are 3-vertices, by Claim 4.14. For \( i = 1, 3, 4 \), let \( N(x_i) \setminus \{ x_0, x_1, x_2 \} = \{ x'_i \} \). Consider the following cases, see Fig. 4.12.

(1) If \( x'_3 = x_4 \) then \( E' = \{ \{ x_1, x'_1 \}, \{ x_3, x_4 \} \} \) is a good set, since \( |\Psi(E')| \leq 14 \).

(2) If \( x'_3 = x'_4 \) then \( E' = \{ \{ x_1, x'_1 \}, \{ x_3, x'_3 \} \} \) is a good set, since \( |\Psi(E')| \leq 18 \).

(3) If \( x'_3 \) and \( x'_4 \) are adjacent then, since \( G \) is plane, the embedding of \( \{ x'_3, x'_4 \} \) is necessarily exterior to the curve formed by the cycle \( (x_0, x_1, x_2, x_4) \). By the Jordan Curve Theorem, \( x_1 \) may not be adjacent to \( x'_3 \) or \( x'_4 \). Then \( E' = \{ \{ x_0, x_1 \}, \{ x'_3, x'_4 \} \} \) is a good set, since \( |\Psi(E')| \leq 18 \).

All of the above cases may be repeated with the roles of \( x_3 \) and \( x_4 \) played instead by, respectively, \( x_1 \) and \( x_3 \), or, respectively, \( x_1 \) and \( x_4 \). Then \( E' = \{ \{ x_1, x'_1 \}, \{ x_3, x'_3 \}, \{ x_4, x'_4 \} \} \) is a good set, since \( |\Psi(E')| \leq 27 \). \( \square \)

**Claim 4.28.** If \( G \) has a 5-cycle adjacent to a 4-cycle along two incident edges then it contains a good set of edges.
Fig. 4.14(a).

Proof. Let \((v_1, v_2, v_3, v_4, v_5)\) be a 5-cycle joined to a 4-cycle \((u, v_4, v_5, v_1)\) along edges \(\{v_4, v_5\}\) and \(\{v_2, v_1\}\). Suppose that in the embedding of the graph, the vertex \(v_5\) is in the interior of the curve formed by the cycle \((u, v_1, v_2, v_3, v_4)\). Observe that \(\{v_4, v_5\}\) and \(\{v_5, v_1\}\) are the only edges common to both cycles, by Claim 4.23. By Claims 4.14 and 4.16, \(u, v_2, v_3, v_5\) are 3-vertices. Let \(N(u) \setminus \{v_1, v_4\} = \{u'\}\) and for \(i = 2, 3, 5\), let \(N(v_i) \setminus \{v_{i-1}, v_{i+1}\} = \{v'_i\}\). By Claims 4.23 and 4.24, none of the edges \(\{u, v_2\}, \{u, v_3\}, \{u, v_5\}, \{v_2, v_3\}, \{v_3, v_5\}\) are exterior to this curve. By the Jordan Curve Theorem, none of the edges \(\{v_3, u'\}, \{v_3, u''\}, \{v_5, u'\}, \{v_5, u''\}\) are adjacent then note that \(v''_3\), \(v''_5\) must be in the interior of the curve formed by the cycle \((u, v_1, v_2, v_3, v_4, v_5)\) joined to a 4-cycle \((u, v_4, v_5, v_1)\) along edges \(\{v_4, v_5\}\) and \(\{v_2, v_1\}\).

Consider the following cases, see Fig. 4.13.

1. If \(u'\) and \(v'_3\) are adjacent then note that \(u'\) and \(v'_5\) must be in the interior of the curve formed by the cycle \((u, v_4, v_5, v_1)\), whereas \(v_3\) and \(v'_5\) are exterior to this curve.

2. If \(v'_3 = v'_5\), and \(v'_3\) and \(v'_5\) have a common neighbor \(p\), then \(E' = \{v_3, v'_5\}, \{u, v_1\}\) is a good set, since \(|\Psi(E')| \leq 15\).

3. If \(v'_3 = v'_5\) and \(u' = v'_2\) then \(E' = \{v_3, v'_5\}, \{u, v_1\}\) is a good set, since \(|\Psi(E')| \leq 15\).

4. If \(v'_3 = v'_5\) then by Claim 4.14, \(v'_5\) is a 3-vertex. Let \(N(v'_5) \setminus \{v_3, v_5\} = \{v''_5\}\). Then \(E' = \{v_2, v'_2, v'_5, v''_5\}, \{u, v_1\}\) is a good set, since \(|\Psi(E')| \leq 26\).

5. If \(u' = v'_3\) then \(u'\) is exterior to the curve formed by the cycle \((u, v_1, v_2, v_3, v_4)\), and in particular \(u'\) is not adjacent to \(v_5\) by the Jordan Curve Theorem. Then \(E' = \{u', v_5\}, \{v_5, v'_5\}\) is a good set, since \(|\Psi(E')| \leq 16\). The case for which \(u' = v'_2\) is handled similarly.

6. The set \(E' = \{u', v_2, v_3\}, \{v_5, v'_5\}\) is good, since \(|\Psi(E')| \leq 27\).

\[\square\]

Claim 4.29. If \(G\) has three 4-cycles in sequence then it contains a good set of edges.

Proof. Let \((u_1, u_2, u_3, u_4), (u_1, u_4, u_5, u_6), (u_5, u_7, u_8, u_6)\) be three 4-cycles that are in sequence. Then \(E' = \{u_1, u_2\}, \{u_5, u_6\}\) is a good set, since \(|\Psi(E')| \leq 18\). See Fig. 4.14(a).

\[\square\]
4.3. INDUCED MATCHINGS IN PLANAR GRAPHS

Claim 4.30. If $G$ has a 4-cycle adjacent to two 4-cycles that are in sequence then it contains a good set of edges.

Proof. Let $(u_1, u_2, u_3, u_4), (u_3, u_5, u_6, u_4)$ be a sequence of 4-cycles and let $(v_1, u_5, u_3, u_2)$ be an adjacent 4-cycle. By Claim 4.14, $v_1$ is a 3-vertex, so let $N(v_1) \setminus \{u_2, u_5\} = v_1'$. Then $E' = \{\{u_3, u_4\}, \{v_1, v_1'\}\}$ is a good set, since $|\Psi(E')| \leq 18$. See Fig. 4.14(b).

Claim 4.31. If $G$ has a 5-cycle adjacent to two 4-cycles that are in sequence then it contains a good set of edges.

Proof. Let $(u_1, u_2, u_3, u_4), (u_3, u_5, u_6, u_4)$ be a sequence of 4-cycles and let $(v_1, v_2, u_5, u_3, u_2)$ be an adjacent 5-cycle. Then $E' = \{\{u_3, u_4\}, \{v_1, v_2\}\}$ is a good set, since $|\Psi(E')| \leq 18$. See Fig. 4.14(c).

Claim 4.32. If $G$ has a 6-cycle adjacent to two 4-cycles that are in sequence then it contains a good set of edges.

Proof. Let $(u_1, u_2, u_3, u_4), (u_3, u_5, u_6, u_4)$ be a sequence of 4-cycles and let $(v_1, v_2, v_3, u_5, u_3, u_2)$ be an adjacent 6-cycle. By Claim 4.23, $u_1$ and $v_1$ are not adjacent. By Claim 4.24, $v_1$ and $v_2$ are not adjacent. By Claim 4.28, $u_1$ and $v_3$ are not adjacent. By Claim 4.16, $v_1$ is a 3-vertex, so let $N(v_1) \setminus \{u_2, v_2\} = \{v_1'\}$. By Claim 4.29, $u_1$ and $v_1'$ are not adjacent. Consider the following cases, see Fig. 4.15(a).
(1) If \( v'_1 \) and \( v'_3 \) are adjacent then \( E' = \{ \{ v_1, u_2 \}, \{ v_3, u_5 \} \} \) is a good set, since \( |\Psi(E')| \leq 17 \).

(2) The set \( E' = \{ \{ u_1, u_4 \}, \{ v_3, u_5 \} \} \) is good, since \( |\Psi(E')| \leq 26 \).

Claim 4.33. If \( G \) has a 7-cycle adjacent to two 4-cycles that are in sequence then it contains a good set of edges.

**Proof.** Let \( (u_1, u_2, u_3, u_4), (u_3, u_5, u_6, u_7, u_8) \) be a sequence of 4-cycles and let \( (v_1, v_2, v_3, v_4, v_5, u_5, u_6, u_4) \) be an adjacent 7-cycle. By Claim 4.23, \( u_1 \) and \( v_1 \) are not adjacent. By Claim 4.29, \( u_1 \) and \( v_2 \) are not adjacent. By Claim 4.24, \( v_1 \) and \( v_4 \) are not adjacent. By Claim 4.31, \( v_1 \) and \( v_4 \) are not adjacent. By Claim 4.32, \( v_2 \) and \( v_4 \) are not adjacent. Then \( E' = \{ \{ u_1, u_4 \}, \{ v_1, v_2 \}, \{ v_4, u_5 \} \} \) is a good set, since \( |\Psi(E')| \leq 26 \). See Fig. 4.15(b).

Claim 4.34. If \( G \) has a pair of adjacent 5-cycles and a 4-cycle all incident at a common vertex then it contains a good set of edges.

**Proof.** Let \( (v_1, v_2, v_3, v_4, v_5), (v_2, v_1, v_6, v_7, v_8) \) be 5-cycles and let \( (v_3, v_5, v_8, v_9) \) be a 4-cycle, all incident at vertex \( v_2 \). Notice that \( v_4, v_5, v_6 \) and \( v_7 \) are 3-vertices by Claim 4.16, and \( v_9 \) is a 3-vertex by Claim 4.14. For \( i = 4, 5, 6, 7, 9 \), let \( N(v_i) \setminus \{ v_{i-1}, v_{i+1} \} = \{ v'_i \} \). By Claim 4.24, \( v_5 \) and \( v_6 \) are not adjacent. By Claim 4.28, \( v_4 \) and \( v_6 \) are not adjacent. Consider the following cases, see Fig. 4.16.

1. If \( v'_5 = v'_6 \) then by Claim 4.14, \( v'_5 \) is a 3-vertex, so let \( N(v'_5) \setminus \{ v_5, v_6 \} = \{ v'_6 \} \). Then, set \( E'_1 = \{ \{ v_1, v_2 \}, \{ v_5, v'_5 \}, \{ v'_6, v'_6 \} \} \) and \( E'_2 = \{ \{ v_1, v_2 \}, \{ v_4, v_4 \}, \{ v_7, v'_7 \} \} \), so that both \( |\Psi(E'_1)| \leq 27 \) and \( |\Psi(E'_2)| \leq 27 \). By the Jordan Curve Theorem, either \( E'_1 \) or \( E'_2 \) is a good set.

2. If \( v'_4 = v'_6 \) then, by the Jordan Curve Theorem, neither \( \{ v_5, v_7 \} \) nor \( \{ v_5, v'_7 \} \) are edges. Then \( E' = \{ \{ v_2, v_3 \}, \{ v_5, v'_5 \}, \{ v_6, v_7 \} \} \) is a good set, since \( |\Psi(E')| \leq 25 \).
4.3. INDUCED MATCHINGS IN PLANAR GRAPHS

(3) The set $E' = \{\{v_2, v_8\}, \{v_4, v_5\}, \{v_6, v_6'\}\}$ is good, since $|\Psi(E')| \leq 26$. □

Claim 4.35. If $G$ has a 4-cycle adjacent to a 5-cycle then it contains a good set of edges.

Proof. Let $(u_1, u_2, u_3, u_4)$ be a 4-cycle and let $(u_4, u_3, v_1, v_2, v_3)$ be a 5-cycle sharing the edge $\{u_3, u_4\}$ with the 4-cycle. Notice that $\{u_3, u_4\}$ is the only edge common to both cycles, by Claim 4.28. By Claim 4.14, $u_1$ is a 3-vertex, so let $N(u_1) \setminus \{u_2, u_4\} = \{u_1'\}$. By 4.16, $v_3$ is a 3-vertex, so let $N(v_3) \setminus \{u_4, v_2\} = \{v_3'\}$. By Claim 4.27, $u_1$ is not adjacent to $v_1$. By Claim 4.28, $u_1'$ is not adjacent to $v_1$. By Claim 4.23, $u_1$ is not adjacent to $v_3$. By Claim 4.31, $u_1' \neq v_3'$. By Claim 4.34, $u_1'$ and $v_3'$ are not adjacent. By Claim 4.23, $v_1$ is not adjacent to $v_3$. By Claim 4.28, $v_1$ is not adjacent to $v'_3$. Then $E' = \{\{u_1, u_1'\}, \{u_3, v_1\}, \{v_3, v_3'\}\}$ is a good set, since $|\Psi(E')| \leq 27$. See Fig. 4.17(a).

□

Claim 4.36. If $G$ has a pair of adjacent 5-cycles then it contains a good set of edges.

Proof. Let $(v_1, v_2, v_3, v_4, v_5)$ and $(v_5, v_4, v_6, v_7, v_8)$ be two 5-cycles with common edge $\{v_4, v_5\}$. By Claim 4.28, the cycles have at most two edges in common. Consider the following cases, see Fig. 4.17(b).

(1) If the cycles have exactly two edges in common, say $\{v_1, v_5\}$ in addition to $\{v_4, v_5\}$, then $v_1 = v_5$. By Claim 4.35, $v_2$ and $v_7$ are not adjacent. Then $E' = \{\{v_1, v_2\}, v_7, v_8\}$ is a good set, since $|\Psi(E')| \leq 14$.

(2) By Claim 4.16, $v_8$ is a 3-vertex, so let $N(v_8) \setminus \{v_5, v_7\} = \{v_8'\}$. By Claim 4.28, $v_1$ is not adjacent to $v_6$. By Claim 4.24, $v_1 \neq v_6'$. By Claim 4.35, $v_1$ and $v_6'$ are not adjacent. By Claim 4.28, $v_2$ is not adjacent to $v_6$. By Claim 4.28, $v_2 \neq v_6'$. If $v_2$ and $v_6'$ are adjacent, then we may identify two 5-cycles with exactly two common edges, handled in the subcase above. By Claim 4.23, $v_6 \neq v_6'$. By Claim 4.28, $v_6$ and $v_6'$ are not adjacent. Then $E' = \{\{v_1, v_2\}, \{v_4, v_6\}, \{v_8, v_6'\}\}$ is a good set, since $|\Psi(E')| \leq 27$. □

Claim 4.37. If $G$ has a 5-cycle in sequence with a 6-cycle then it contains a good set of edges.

Proof. Let $(v_1, v_2, v_3, v_4, v_5)$ be a 5-cycle and let $(v_5, v_4, v_6, v_7, v_8, v_9)$ be a 6-cycle sharing the edge $\{v_4, v_5\}$ with the 5-cycle. Observe that $v_2$ is a 3-vertex, by Claims 4.16; let $N(v_2) \setminus \{v_1, v_3\} = \{v'_2\}$. Consider the following cases, see Fig. 4.18.
By Claim 4.32, \( u \) and \( v \) are adjacent. Then by Claim 4.28, the edge \( \{u, v\} \) is adjacent to a 6-cycle, see Claim 4.37.

Proof. Let \( G \) has a 4-cycle adjacent to a 6-cycle. By Claim 4.35, \( v_1 \) and \( v_8 \) are not adjacent. Then \( E' = \{v_1, v_2, v_4, v_6, v_8\} \) is a good set, since \( |\Psi(E')| \leq 26 \). The case that \( v_2 \) and \( v_8 \) are adjacent is handled similarly.

(2) If \( v'_2 \) and \( v_7 \) are adjacent then by Claim 4.28, \( v_3 \) and \( v_6 \) are not adjacent. By Claim 4.24, \( v_1 \) and \( v_9 \) are not adjacent. Then \( E' = \{v_1, v_2, v_4, v_6, v_9\} \) is a good set, since \( |\Psi(E')| \leq 27 \). The case that \( v_2 \) and \( v'_9 \) are adjacent is handled similarly.

(3) The set \( E' = \{v_2, v'_2, v_4, v_8\} \) is good, since \( |\Psi(E')| \leq 27 \). □

Claim 4.38. If \( G \) has a 4-cycle adjacent to a 6-cycle along two incident edges then it contains a good set of edges.

Proof. Let \( \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\} \) be a 6-cycle and let \( \{v_3, v_2, v_1, v_7\} \) be a 4-cycle. By Claim 4.28, \( v_4 \) and \( v_8 \) are not adjacent. Then \( E' = \{v_1, v_4, v_3, v_7\} \) is a good set, since \( |\Psi(E')| \leq 17 \). See Fig. 4.19(a). □

Claim 4.39. If \( G \) has a 4-cycle in sequence with a 6-cycle then it contains a good set of edges.

Proof. Let \( \{u_1, u_2, u_3, u_4\} \) be a 4-cycle and let \( \{u_4, u_3, v_1, v_2, v_3, v_4\} \) be a 6-cycle sharing the edge \( \{u_3, u_4\} \) with the 4-cycle. By Claim 4.27, \( u_1 \) and \( v_1 \) are not adjacent. By Claim 4.38, \( u_1 \) and \( v_3 \) are not adjacent. By Claim 4.23, \( u_1 \) and \( v_4 \) are not adjacent. By Claim 4.28, \( u'_3 \) and \( v_1 \) are not adjacent. By Claim 4.35, \( u'_3 \) and \( v_3 \) are not adjacent. By Claim 4.32, \( u'_3 \) and \( v_4 \) are not adjacent. By Claim 4.24, \( v_1 \) and \( v_3 \) are not adjacent. By Claim 4.29, \( v_1 \) and \( v_4 \) are not adjacent. Then \( E' = \{u_1, u'_3, v_3, v_4\} \) is a good set, since \( |\Psi(E')| \leq 27 \). See Fig. 4.19(b). □

Claim 4.40. If \( G \) has a 2-vertex at distance two from a 3-cycle then it contains a good set of edges.
4.3. INDUCED MATCHINGS IN PLANAR GRAPHS

Proof. Let \( u \) be a 2-vertex at distance two from a 3-cycle \((x, y, z)\), where \( u \) and \( x \) have a common neighbor \( q \). Notice that \( q \) is neither adjacent to \( y \) nor to \( z \), by Claim 4.22. Then \( E' = \{\{u, q\}, \{y, z\}\} \) is a good set, since \(|\Psi(E')| \leq 17\). See Fig. 4.20(a).

Claim 4.41. If \( G \) has a 2-vertex at distance two from a 4-cycle then it contains a good set of edges.

Proof. Let \( u \) be a 2-vertex at distance two from a 4-cycle \((w, x, y, z)\), where \( u \) and \( w \) have a common neighbor \( q \). Notice that \( q \) is neither adjacent to \( x \) nor \( z \), by Claim 4.23. Also, both \( x \) and \( z \) are 3-vertices, by Claim 4.14. By Claim 4.14, \( u \) is neither adjacent to \( x \) nor \( z \). By Claim 4.16, \( u \) is neither adjacent to \( x' \) nor \( z' \). Let \( N(x) \setminus \{w, y\} = \{x'\} \) and \( N(z) \setminus \{w, y\} = \{z'\} \). Then \( q \) is neither adjacent to \( x' \) nor \( y' \), by Claim 4.19. Then \( E' = \{\{u, q\}, \{x, x'\}, \{z, z'\}\} \) is a good set, since \(|\Psi(E')| \leq 27\). See Fig. 4.20(b).

Claim 4.42. If \( G \) has a 2-vertex at distance two from a 5-cycle then it contains a good set of edges.

Proof. Let \( u \) be a 2-vertex at distance two from a 5-cycle \((x_0, x_1, x_2, x_3, x_4)\), where \( u \) and \( x_0 \) have a common neighbor \( q \). By Claim 4.16, \( x_1 \) is a 3-vertex, so let \( N(x_1) \setminus \{x_0, x_2\} = \{x'_1\} \). Notice that \( q \) is neither adjacent to \( x_1 \) nor to \( x_4 \), by Claim 4.24. By Claim 4.35, \( q \) is not adjacent to \( x'_1 \). By Claim 4.28, \( q \) is not adjacent to \( x_3 \). By Claim 4.35, \( u \) is not adjacent to \( x_1 \). Claim 4.36, \( u \) is neither adjacent to \( x'_1 \) nor to \( x_3 \). By Claim 4.35, \( u \) is not adjacent to \( x_4 \). By Claim 4.23, \( x_1 \) is neither adjacent to \( x_3 \) nor to \( x_4 \). By Claim 4.28, \( x'_1 \) is neither adjacent to \( x_3 \) nor to \( x_4 \). Then \( E' = \{\{u, q\}, \{x_1, x'_1\}, \{x_3, x_4\}\} \) is a good set, since \(|\Psi(E')| \leq 27\). See Fig. 4.20(c).

Claim 4.43. If \( G \) has two 3-cycles at distance one then it contains a good set of edges.

Proof. Let \((u, v, w)\) and \((x, y, z)\) be 3-cycles at distance one, with \(\{u, x\}\) being an edge. By Claim 4.23, \( y \) is neither adjacent to \( v \) nor to \( w \). Then \( E' = \{\{v, w\}, \{x, y\}\} \) is a good set, since \(|\Psi(E')| \leq 17\). See Fig. 4.21(a).

Claim 4.44. If \( G \) has a 3-cycle at distance one from a 4-cycle then it contains a good set of edges.
Proof. Let \((x, y, z)\) be a 3-cycle at distance one to a 4-cycle \((v_1, v_2, v_3, v_4)\), with \(\{x, v_1\}\) being an edge. Observe that \(v_2, v_4\) are 3-vertices by Claim 4.14. Let \(N(v_i) \setminus \{v_{i-1}, v_{i+1}\} = \{v'_i\}\), for \(i = 2, 4\). By Claim 4.23, \(z\) is neither adjacent to \(v_2\) nor to \(v_4\).

By Claim 4.24, \(z\) is neither adjacent to \(v'_2\) nor to \(v'_4\). By Claim 4.22, \(v_2\) and \(v_4\) are not adjacent. Then \(E' = \{\{x, z\}, \{v_2, v'_2\}, \{v_4, v'_4\}\}\) is a good set, since \(|\Psi(E')| \leq 27\). See Fig. 4.21(b). □

**Claim 4.45.** If \(G\) has a 3-cycle at distance one from a 5-cycle then it contains a good set of edges.

Proof. Let \((x, y, z)\) be a 3-cycle at distance one to a 5-cycle \((v_1, v_2, v_3, v_4, v_5)\), with \(\{x, v_1\}\) being an edge. Observe that \(v_2\) is a 3-vertex by Claim 4.16, so let \(N(v_2) \setminus \{v_1, v_4\} = \{v'_2\}\). By Claim 4.23, \(z\) is neither adjacent to \(v_2\) nor to \(v_5\). By Claim 4.24, \(z\) is neither adjacent to \(v'_2\) nor to \(v_4\). By Claim 4.23, \(v'_2\) is neither \(v_4\) nor \(v_5\). By Claim 4.28, \(v'_2\) is neither adjacent to \(v_4\) nor to \(v_5\). Then \(E' = \{\{x, z\}, \{v_2, v'_2\}, \{v_4, v_5\}\}\) is a good set, since \(|\Psi(E')| \leq 27\). See Fig. 4.21(c). □

**Claim 4.46.** If \(G\) has a 5-cycle at distance one from a double 4-face then it contains a good set of edges.

Proof. Let \((u_1, u_2, u_3, u_4, u_5)\) be a 5-cycle at distance one from a double 4-face \((v_1, v_2, v_3, v_4, v_5)\), with \(\{u_1, v_1\}\) being an edge. Observe that \(u_5\) is a 3-vertex, by Claim 4.16, so let \(N(u_5) \setminus \{u_1, u_4\} = \{u'_5\}\). Consider the following cases, see Fig. 4.22(a).
Claim 4.35.\footnote{Note that, by Claim 4.29, there are no edges among $u$.}

Proof. Let $v$.

(1) If $u$ and $v$ are adjacent then by Claim 4.23, $u$ is neither adjacent to $u_2$ nor to $u_3$. By Claim 4.28, $u_2^\prime$ is neither adjacent to $u_2$ nor to $u_3$. By the Jordan Curve Theorem, $v_4$ and $v_5$ are not adjacent. By Claim 4.35, $u_2$ is not adjacent to $u_4$. By Claim 4.36, $u_3$ and $v_4$ are not adjacent. By Claim 4.36, $u_2$ is not adjacent to $v_5$. By Claim 4.35, $u_3$ is not adjacent to $v_5$. By Claim 4.35, $u_4^\prime \neq v_4$. By Claim 4.36, $u_5^\prime \neq v_5$. By Claim 4.37, $u_5^\prime$ and $v_5$ are not adjacent. Then $E' = \{\{u_2, u_3\}, \{u_5, u_5'\}, \{v_1, v_4\}, \{v_5, v_6\}\}$ is a good set, since $|\Psi(E')| \leq 33$. The case where $u_3$ and $v_6$ are adjacent is treated similarly.

(2) If neither $\{u_4, v_6\}$ nor $\{u_3, v_6\}$ are edges then $E' = \{\{u_3, u_4\}, \{u_1, v_1\}, \{v_3, v_6\}\}$ is a good set, since $|\Psi(E')| \leq 26$.

Claim 4.47. If $G$ has a pair of double 4-faces at distance one then it contains a good set of edges.

Proof. Let $(u_1, u_2, u_3, u_4)$, $(u_3, u_2, u_5, u_6)$ and $(v_1, v_2, v_3, v_4)$, $(v_4, v_3, v_5, v_6)$ be double 4-faces at distance one, with $\{u_1, v_1\}$ being an edge. By Claim 4.23, $v_5$ and $v_6$ are not adjacent. By Claim 4.27, $v_2$ and $v_4$ are not adjacent. Then $E' = \{\{u_2, u_3\}, \{v_1, v_2\}, \{v_5, v_6\}\}$ is a good set, since $|\Psi(E')| \leq 27$.

Claim 4.48. If $G$ has a 4-cycle, $(\leq 8)$-cycle, and 4-cycle in sequence then it contains a good set of edges.

Proof. Let $(u_1, u_2, u_3, u_4), (v_1, v_2, v_3, v_4)$ be 4-cycles at distance one, with $\{u_1, v_1\}$ being an edge, and $C$ be the adjacent $(\leq 8)$-cycle. Suppose, without loss of generality, that $C$ contains both $u_4$ and $v_2$. Observe that $C$ is either a 7-cycle or an 8-cycle, by Claims 4.29, 4.35 and 4.39. Also, $u_2, u_3, u_4$ and $v_2, v_3, v_4$ are 3-vertices, by Claim 4.14. For $i = 2, 3, 4$, let $N(u_i) \setminus \{u_i, u_2, u_3, u_4\} = \{u_i'\}$ and $N(v_i) \setminus \{v_i, v_2, v_3, v_4\} = \{v_i'\}$. Note that, by Claim 4.29, there are no edges among $u_2, u_4, v_2, v_4$. Furthermore, by Claim 4.35, $|\{u_2', u_4', v_2', v_4'\}| = 4$. Also, by Claim 4.39, there are no edges among $u_i', u_i', v_i', v_i'$. Consider the following cases, see Fig. 4.23.

![Figure 4.23: A 4-cycle, $(\leq 8)$-cycle, and 4-cycle in sequence, see Claim 4.48.](image-url)
Proof. If $G$ has a $v$-cycle, then $E' = \{\{u_2, u_3\}, \{v_1, v_4\}, \{v'_2, v'_3\}\}$ is a good set, since $|\Psi(E')| \leq 25$. The cases in which $\{v'_3, u'_3\}$ are edges, $\{v'_3, u'_4\}$ are edges, $\{v'_3, u'_2\}$ are edges, or $\{v'_3, u'_3\}$ and $\{u'_3, u'_4\}$ are edges are handled similarly.

(2) If $v'_3$ and $u'_3$ are adjacent then none of the edges $\{u_2, u'_3\}, \{u_4, u'_3\}, \{v_2, v'_3\}, \{v_4, v'_3\}$ is present by Claim 4.23. None of the edges $\{u_2, v'_3\}, \{u_4, v'_3\}, \{v_2, u'_3\}, \{v_4, u'_3\}$ is present by Claim 4.39. None of the edges $\{u_2, v'_3\}, \{u'_4, v'_3\}, \{v'_2, u'_3\}, \{v'_4, u'_3\}$ is present by Claim 4.35. Then it follows that $E' = \{\{u_2, u'_3\}, \{u_4, u'_4\}, \{v_2, v'_2\}, \{v_4, v'_4\}, \{u'_3, v'_3\}\}$ is a good set, since $|\Psi(E')| \leq 45$.

(3) If $u'_3 = v'_3$ then $E' = \{\{u_1, v_1\}, \{u_3, u'_3\}\}$ is a good set, since $|\Psi(E')| \leq 18$.

(4) The set $E' = \{\{u_1, v_1\}, \{u_3, u'_3\}, \{v_3, v'_3\}\}$ is good, since $|\Psi(E')| \leq 27$. \hfill \Box

Claim 4.49. If $G$ has a 4-cycle, 4-cycle and 7-cycle in sequence then it contains a good set of edges.

Proof. Let $(u_1, u_2, u_3, u_4), (u_4, u_3, u_5, u_6), \text{ and } (u_6, u_5, v_1, v_2, v_3, v_4, v_5)$ be the sequence of cycles. By Claim 4.14, $u_1$ and $u_2$ are 3-vertices, so let $N(u_i) \setminus \{u_1, u_2, u_3, u_4\} = \{u'_i\}$ for $i = 1, 2$. By Claim 4.41, $v_2$ and $v_4$ are 3-vertices, so let $N(v_i) \setminus \{v_1, v_3, v_5\} = \{v'_i\}$ for $i = 2, 4$. By Claims 4.25 and 4.35, note that the 7-cycle may not have any chords. Consider the following cases, see Fig. 4.24.

(1) If $v'_2 = v'_4$ then $E' = \{\{u_3, u'_4\}, \{v_1, v_2\}, \{v_4, v'_5\}\}$ is a good set, since $|\Psi(E')| \leq 25$.

(2) If $v'_2$ and $v'_4$ are adjacent then $E' = \{\{u_3, u'_4\}, \{v_1, v_2\}, \{v_4, v'_5\}\}$ is a good set, since $|\Psi(E')| \leq 27$.

(3) If $u'_2 = v'_4$ then $E' = \{\{u_3, u'_4\}, \{v_1, v_2\}, \{v_4, v'_5\}\}$ is a good set, since $|\Psi(E')| \leq 27$. The case for which $u'_1 = v'_2$ is treated similarly.

(4) By Claim 4.35, the edges $\{u_1, v_1\}, \{u_2, v_2\}$ are not present. By Claim 4.39, $u'_1 \neq v'_4$ and $u'_2 \neq v'_2$, and the edges $\{u_1, v_2\}, \{u_2, v_4\}$ are not present. Then $E' = \{\{u_1, u_2\}, \{u_5, u'_6\}, \{v_2, v'_2\}, \{v_4, v'_4\}\}$ is a good set, since $|\Psi(E')| \leq 36$. \hfill \Box
4.3. INDUCED MATCHINGS IN PLANAR GRAPHS

Figure 4.25: A 4-cycle, 7-cycle and 5-cycle in sequence, see Claim 4.50.

(a) Two 3-cycles at distance two. (b) A 3-cycle at distance two from a double 4-face.

Figure 4.26: Figures for Claims 4.51 and 4.52

Claim 4.50. If \( G \) has a 4-cycle, 7-cycle and 5-cycle in sequence then it contains a good set of edges.

Proof. Let \((u_1, u_2, u_3, u_4)\) be a 4-cycle, let \((v_1, v_2, v_3, v_4, v_5)\) be a 5-cycle at distance one with \(\{u_1, v_1\}\) being an edge, and let \((u_1, u_4, v_1, v_2, v_3, v_2, v_1)\) by the adjacent 7-cycle. By Claim 4.14, \(u_2, u_3, u_4\) are 3-vertices, so let \(N(u_i) \setminus \{u_1, u_2, u_3, u_4\} = \{u'_i\}\) for \(i = 2, 3, 4\). Also, \(v_3, v_4\) are 3-vertices by Claim 4.16, so let \(N(v_i) \setminus \{v_2, v_3, v_4, v_5\} = \{v'_i\}\), for \(i = 3, 4\). Consider the following cases, see Fig. 4.25.

1. If \(u'_3 = v'_3\) then neither \(u'_2\) nor \(u'_4\) is adjacent to \(v_1\) by the Jordan Curve Theorem. By Claim 4.39, \(u'_2\) and \(v'_4\) are not adjacent. By Claim 4.49, \(u'_2\) and \(v'_4\) are not adjacent. By Claim 4.33, \(v'_4\) and \(v_1\) are not adjacent. Then \(E' = \{\{u_2, u'_2\}, \{u_4, u'_4\}, \{v_1, v_2\}, \{v_4, v'_4\}\}\) is a good set, since \(|\Psi(E')| \leq 33\).

2. If \(u'_3 = v'_3\) then none of the edges \(\{w_1, u_2\}, \{w_1, u'_2\}, \{w_1, v_3\}\) is present by the Jordan Curve Theorem. By Claim 4.23, \(u_2\) is not adjacent to \(v'_3\). By Claim 4.35, \(u_2\) is not adjacent to \(v_5\) or \(v'_5\). By Claim 4.33, \(u'_2\) and \(v'_4\) are not adjacent. By Claim 4.28, \(v_3\) and \(v_5\) are not adjacent. By Claim 4.33, \(v_3\) and \(v_5\) are not adjacent. Then \(E' = \{\{u_2, u'_2\}, \{u_4, u'_4\}, \{v_1, v_5\}, \{v_3, v'_3\}\}\) is a good set, since \(|\Psi(E')| \leq 35\).

3. The set \(E' = \{\{u_3, u'_3\}, \{u_4, u'_4\}, \{v_3, v'_4\}\}\) is a good set, since \(|\Psi(E')| \leq 27\).

Claim 4.51. If \( G \) has a pair of 3-cycles at distance two then it contains a good set of edges.

Proof. Let \((u, v, w)\) and \((x, y, z)\) be 3-cycles at distance two, where \(u\) and \(x\) have a common neighbor \(p\). By Claim 4.24, \(v\) and \(z\) are not adjacent. Then \(E' = \{\{u, v\}, \{x, z\}\}\) is a good set, since \(|\Psi(E')| \leq 17\). See Fig. 4.26(a).
Claim 4.52. If \( G \) has a 3-cycle at distance two from a double 4-face then it contains a good set of edges.

Proof. Let \((x, y, z)\) be a 3-cycle and let \((v_1, v_2, v_3), (v_3, v_2, v_5, v_6)\) be a double 4-face at distance two, with \(x\) and \(v_1\) having a common neighbor \(p\). By Claim 4.22, \(p\) is neither adjacent to \(y\) nor \(z\). By Claim 4.28, \(p\) is not adjacent to \(v_6\). By Claim 4.26, \(v_6\) is neither adjacent to \(y\) nor \(z\). Then \(E' = \{\{y, z\}, \{p, v_1\}, \{v_3, v_6\}\}\) is a good set, since \(|\Psi(E')| \leq 26\). See Fig. 4.26(b). \(\square\)

Claim 4.53. If \( G \) has a pair of double 4-faces at distance two then it has a good set of edges.

Proof. Let \((u_1, u_2, u_3, u_4), (u_4, u_3, u_5, u_6)\) and \((v_1, v_2, v_3, v_4), (v_5, v_6, v_4)\) be two double 4-faces at distance two, such that \(u_1\) and \(v_2\) have a common neighbor \(p\). Consider the following cases, see Fig. 4.27.

1. If \(u_5\) and \(v_6\) are adjacent then by Claim 4.30, \(p\) and \(u_6\) are not adjacent. Then \(E' = \{\{u_5, u_6\}, \{u_1, p\}, \{v_3, v_4\}\}\) is a good set, since \(|\Psi(E')| \leq 26\).

2. If \(u_5\) and \(v_5\) are adjacent then by Claim 4.30, \(p\) and \(u_6\) are not adjacent. Then \(E' = \{\{u_5, u_6\}, \{u_1, p\}, \{v_3, v_4\}\}\) is a good set, since \(|\Psi(E')| \leq 26\). The case that \(u_6\) and \(v_6\) are adjacent is treated symmetrically.

3. Otherwise, the following pairs of vertices are not adjacent: \(u_2\) and \(u_5\), by Claim 4.23; \(u_2\) and \(u_6\), by Claim 4.30; \(u_2\) and \(v_1\), by Claim 4.31; \(u_2\) and \(v_5\), by Claim 4.32; \(u_2\) and \(v_6\), by Claim 4.33; \(u_5\) and \(v_1\), by Claim 4.33; \(u_5\) and \(v_6\), by (2); \(u_5\) and \(v_6\), by (1); \(u_6\) and \(v_6\), by Claim 4.32; \(u_6\) and \(v_1\), by Claim 4.33; \(u_6\) and \(v_5\), by (2); \(v_1\) and \(v_5\), by Claim 4.30; \(v_1\) and \(v_6\), by Claims 4.23. Then \(E' = \{\{u_1, u_2\}, \{u_5, u_6\}, \{v_1, v_2\}, \{v_5, v_6\}\}\) is a good set, since \(|\Psi(E')| \leq 35\). \(\square\)

To wrap up the proof of Lemma 4.8, we list the specific claims which certify the presence of a good set, given the presence of one of the structures (C1)–(C12).

(C1) Claim 4.11.
(C3) Claims 4.12 and 4.15.
(C4) Claims 4.18, 4.19, 4.20, 4.40, 4.41 and 4.42.
4.3. INDUCED MATCHINGS IN PLANAR GRAPHS

(C7) Claims 4.43, 4.44 and 4.45.
(C8) Claims 4.23, 4.29, 4.35, 4.39, 4.32, 4.33, and 4.49.
(C9) Claim 4.48.
(C10) Claim 4.50.
(C11) Claims 4.22, 4.23, 4.43, 4.44, 4.47, 4.51, 4.52 and 4.53.
(C12) Claim 4.46.

This concludes the proof of Lemma 4.8. □

Lemma 4.9. Every plane graph of maximum degree three contains one of the structures (C1)–(C12) listed in Lemma 4.8.

We now prove Lemma 4.9 by means of a discharging procedure. Suppose that \( G \) is a plane graph with maximum degree three that does not contain any of the structures (C1)-(C12).

We will obtain a contradiction by using the Discharging Method, which is commonly used in graph coloring. We assign to each vertex and face of \( G \) an initial charge, such that their total sum is negative, and then apply redistribution rules for exchanging charge between the vertices and faces. These redistribution rules leave the total sum of charges invariant; however, we will prove that if none of (C1)-(C12) occurs in \( G \), then each vertex and each face will have non-negative charge after the discharging procedure has finished. Hence we must have at least one of (C1)-(C12). We proceed to fill in the details.

**Initial charge** For every vertex \( v \in V(G) \), let its initial charge \( ch(v) \) be \( 2d(v) - 6 \), and for every face \( f \in F(G) \), let its initial charge \( ch(f) \) be \( d(f) - 6 \). We claim that this way the total sum of initial charges will be negative. To see this, note that by Euler’s formula \( 6|E(G)| - 6|V(G)| - 6|F(G)| = -12 \). It follows from \( \sum_{v \in V(G)} d(v) = 2|E(G)| = \sum_{f \in F(G)} d(f) \) that

\[
-12 = (4|E(G)| - 6|V(G)|) + (2|E(G)| - 6|F(G)|) = \sum_{v \in V(G)} (2d(v) - 6) + \sum_{f \in F(G)} (d(f) - 6),
\]

which proves the claim.

**Discharging rule** We apply the following rule to each \((\geq 7)\)-face.

Each \((\geq 7)\)-face sends a charge of

- 1/5 to each adjacent 5-face,
- 1/2 to each adjacent 4-face that is not in a double 4-face,
- 1/2 to each adjacent 4-face in a double 4-face if it is adjacent to both 4-faces,
- 1 to each adjacent 4-face in a double 4-face if it is adjacent to only one,
• 1 to each adjacent 3-face, and
• 1 to each incident 2-vertex.

When we say that an \((\geq 7)\)-face sends charge to an adjacent face or incident vertex, we mean that the charge is sent as many times as these elements are adjacent or incident to each other. The final charge of each vertex \(v\) and each face \(f\) is denoted by \(\text{ch}^*(v)\) and \(\text{ch}^*(f)\), respectively.

**Final charge of 2-vertices**  The initial charge of a 2-vertex is \(-2\). By (C2) it is adjacent to two \((\geq 8)\)-faces. Hence it receives 2, so that the final charge is \(\geq 0\).

**Final charge of 3-vertices**  A 3-vertex has initial charge 0. Since it sends no charge, its final charge is \(\geq 0\).

**Final charge of 3-faces**  A 3-face has initial charge \(-3\). By (C5) it is only adjacent to \((\geq 8)\)-faces. Hence it receives charge of 3 and the final charge is \(\geq 0\).

**Final charge of 4-faces**  Let \(f\) be a 4-face; then its initial charge is \(\text{ch}(f) = -2\). If \(f\) is not in a double 4-face then by (C5) and (C6) \(f\) is only adjacent to \((\geq 7)\)-faces, and receives a charge of at least 1/2 from each of them; thus \(\text{ch}^*(f) \geq 0\). Otherwise, if \(f\) is in a double 4-face, then \(f\) is adjacent to exactly one 4-face and three \((\geq 8)\)-faces by (C8). Thus, \(f\) receives a charge of 1 from one \((\geq 8)\)-face and charges of at least 1/2 from the other two, and so \(\text{ch}^*(f) \geq 0\).

**Final charge of 5-faces**  Let \(f\) be a 5-face; then its initial charge is \(\text{ch}(f) = 1\). By (C5), (C8), (C9) and (C10), \(f\) is adjacent to no 3-faces, no double 4-faces and at most two 4- or 5-faces. Thus, \(f\) sends a charge of at most \(2 \cdot 1/2\), and so \(\text{ch}^*(f) \geq 0\).

**Final charge of 6-faces**  The initial charge of a 6-face is 0 and it sends no charge, so its final charge is \(\geq 0\).

**Final charge of 7-faces**  Let \(f\) be a 7-face; then its initial charge is \(\text{ch}(f) = 1\). By (C5), (C8), (C9) and (C10), \(f\) is adjacent to no 3-faces, no double 4-faces and at most two 4- or 5-faces. Thus, \(f\) sends a charge of at most \(2 \cdot 1/2\), and so \(\text{ch}^*(f) \geq 0\).

**Final charge of 8-faces**  Let \(f\) be an 8-face; then its initial charge is \(\text{ch}(f) = 2\). We consider several cases.

First, suppose that \(f\) is incident to a 2-vertex. By (C3), \(f\) is incident to at most two 2-vertices. However, if \(f\) is incident to exactly two 2-vertices, then by (C2) and (C4) \(f\) is adjacent only to \((\geq 6)\)-faces; thus, \(\text{ch}^*(f) = 0\). So assume that \(f\) is incident to exactly one 2-vertex \(v\). By (C4), faces that are adjacent to \(f\) but at distance at most two from \(v\) must be \((\geq 6)\)-faces. There remain two other faces adjacent to \(f\), and these may send charge at most 1 combined and so \(\text{ch}^*(f) \geq 0\). Thus, we may hereafter assume that \(f\) is not incident to a 2-vertex.
4.3. INDUCED MATCHINGS IN PLANAR GRAPHS

Second, suppose that \( f \) is adjacent to a 3-face \( f' \). By (C5) and (C7), faces that are adjacent to \( f \) but at distance at most two from \( f' \) must be \((\geq 6)\)-faces. There remain three other faces adjacent to \( f \), call them \( f_1, f_2, f_3 \), in sequence. By (C7), (C8), (C9), (C11), and (C12), if one of these is a 3-face or part of a double 4-face, then the others are \((\geq 6)\)-faces. By (C9), at most one of these is a 4-face (that is not part of a double 4-face). By (C6), at most two of these are 5-faces. In all of these sub-cases, \( f_1, f_2, f_3 \) are sent from \( f \) a combined charge of at most 1. Since the total charge sent to the other five faces is 1, this implies that \( \text{ch'}(f) \geq 0 \). Thus, we may hereafter assume that \( f \) is not adjacent to a 3-face.

Third, suppose that \( f \) is adjacent to a 4-face \( f' \). Assume that \( f' \) is part of a double 4-face. By (C6), (C8), (C9), (C11), and (C12), faces that are adjacent to \( f \) but at distance at most two from \( f' \) must be \((\geq 6)\)-faces. There remain at most three other faces adjacent to \( f \) and we proceed as in the previous case. Thus \( f' \) is not part of a double 4-face. By (C6), (C9), of the faces that are adjacent to \( f \) but at distance at most two from \( f' \), none are 4-faces and at most two are 5-faces. Thus in total, at most 1/2 + 2/5 < 1 charge is sent to \( f' \) and these four faces. Again, there remain at most three other faces adjacent to \( f \) and we proceed as in the previous case. Thus, we may hereafter assume that \( f \) is not adjacent to an \((\leq 4)\)-face.

Finally, by (C6), \( f \) is adjacent to at most four 5-faces, and so \( f \) sends total charge of at most \( 4 \cdot 1/5 < 2 \) and \( \text{ch'}(f) > 0 \).

**Final charge of \((\geq 9)\)-faces**  Let \( f \) be an \((\geq 9)\)-face and let \((e_1, e_2, e_3, e_4)\) be a path of four edges along \( f \), where \( e_i = \{v_i, v_{i+1}\} \) for \( i = 1, \ldots, 4 \). Denote by \( f_i \) the face adjacent to \( f \) via the edge \( e_i \). We first show that the combined charge sent through these four edges (counting half of the charge contributed to the end-vertices \( v_1, v_5 \) if 2-vertices) is at most 3/2.

First, suppose that at least one of the \( v_i \) is a 2-vertex. By (C3), at most two are 2-vertices. If two are, then, without loss of generality, either \( v_1 \) and \( v_4 \) are 2-vertices, or \( v_1 \) and \( v_5 \) are 2-vertices. In the former case, \( f_2 \) is a \((\geq 6)\)-face by (C4) and the total charge sent is 3/2; in the latter case, \( f_2 \) and \( f_3 \) are \((\geq 6)\)-faces by (C4) and the total charge sent is 1. If exactly one of the \( v_i \) is a 2-vertex, then without loss of generality, either \( v_1 \) is is a 2-vertex, or one of \( v_2 \) or \( v_3 \) is. In the former case, \( f_2 \) and \( f_3 \) are \((\geq 6)\)-faces by (C4) and the total charge sent is at most 3/2 (since \( f_4 \) is sent charge at most 1); in the latter case, then all the \( f_i \) are \((\geq 6)\)-faces by (C4) and the total charge sent is 1.

Second, suppose that some \( f_i \) is a 3-face. Without loss of generality, there are two sub-cases to consider: \( i = 1 \) or \( i = 2 \). In the former sub-case, we have by (C5), (C7) and (C11) that \( f_2 \) and \( f_3 \) are both \((\geq 6)\)-faces and \( f_4 \) is forbidden from being a 3-face or part of a double 4-face, in which case the total charge sent is at most 3/2. In the latter sub-case, we have by (C5) and (C7) that \( f_1, f_3 \) and \( f_4 \) are \((\geq 6)\)-faces, in which case the total charge sent is 1.

Third, suppose that some \( f_i \) is part of a double 4-face. Without loss of generality, there are two sub-cases to consider: \( i = 1 \) or \( i = 2 \). In the former sub-case, suppose that \( f_2 \) is also part of the same double 4-face. Then by (C8) and (C11) \( f_3 \) is a \((\geq 6)\)-face and \( f_4 \) is not part of a double 4-face; thus the total charge sent is at most 3/2. Therefore, in the former case we may suppose that \( f_2 \) is not a 4-face. By (C6), (C8) and (C11), at most one of \( f_2, f_3, f_4 \) is a 4- or 5-face and none is part of a double
4. ALGORITHMS THROUGH EXTREMAL COMBINATORICS

4-face, in which case the total charge sent is at most 3/2. Next, in the latter sub-

case, we have by (C8) that \( f_1, f_3 \) are \((\geq 6)\)-faces, and by (C11) that \( f_4 \) is not part of a
double 4-face, in which case the total charge sent is at most 3/2.

We now have that none of the \( v_i \) is a 2-vertex, none of the \( f_i \) is a 3-face or part

of a double 4-face. By (C6), not every \( f_i \) is a 4- or 5-face. Thus, the total charge sent

is at most 3/2, completing our proof of the claim.

We complete the analysis of the final charge for \( f \). Let us denote the facial
cycle by \((v_1, v_2, v_3, \ldots, v_k)\) and denote by \( f_i \) the face adjacent to \( f \) via the edge
\( e_i = (v_i, v_{i+1} \mod k) \). Assume, without loss of generality, that \( d(v_1) = d(v_2) = 3 \)
and \( f_1 \) is a \((\geq 6)\)-face. By the above claim, the total charge sent through \( e_2, e_3, \ldots, e_9 \)
is at most 3.

Every face \( f_i, i > 9 \), receives a charge of at most 1 from \( f \) (taking into account
charge sent to 2-vertices). Hence, \( f \) sends total charge at most \( 3 + d(f) - 9 = d(f) - 6 = \text{ch}(f) \), and so \( \text{ch}^*(f) \geq 0 \).

We have seen that every vertex and every face of \( G \) has final charge \( \geq 0 \), which
gives the required contradiction and completes the proof of Lemma 4.9.

\[ \square \]

4.4 Concluding Remarks

The field of extremal combinatorics offers many interesting opportunities to design
fast exponential-time algorithms. In particular, non-trivial upper bounds on the
number of maximal subsets with a relevant property and of small size are interest-
ing. Once these bounds have been obtained, it must be shown the collection of such
subsets can be enumerated quickly.

It would be interesting to investigate whether non-trivial upper bounds on ex-
tremal set families can help to speed up fixed-parameter algorithms. For instance,
the fixed-parameter algorithm for \((k)\)-\textsc{Feedback Vertex Set} in tournaments by Dom
et al. [78] is based on iterative compression and it enumerates all feedback vertex
sets of a certain subgraph \( H \). If that enumeration can be restricted to the minimal
feedback vertex sets of \( H \) then the combinatorial bounds presented in this chapter
immediately yield a significant improvement in running time.

Three concrete open problems are the following: Can we close the gap between
the lower and upper bound on the maximum number of minimal feedback vertex
sets in tournaments, and is the Paley graph on 7 vertices indeed the extremal exam-
ple? Can the maximal acyclic subgraphs of a general (directed or undirected) graph
be listed with polynomial delay and polynomial space at the same time? Can we
obtain a non-trivial upper bound on the maximum number of minimal feedback
vertex sets in general directed graphs? Such an upper bound would complement
the following lower bound.

\textbf{Theorem 4.9.} The maximum number of minimal feedback vertex sets in directed graphs
on \( n \) vertices is at least \( 105^n/10 > 1.5926^n \).

\textbf{Proof.} We show a lower bound on the maximum number of minimal feedback vertex
sets in any graph by providing an infinite family of graphs with \( n \) vertices
and \( 105^n/10 > 1.5926^n \) minimal feedback vertex sets. The infinite family consists
of graphs whose strong components are copies of a single graph, the \( k \)-th member
4.4. CONCLUDING REMARKS

Figure 4.28: A graph with $n = 10$ vertices and $105^{n/10}$ minimal feedback vertex sets.

Figure 4.29: The connected 3-regular planar graph $G_0$ generating the infinite family of graphs to prove an upper bound on the size of a maximum induced matching. A maximum induced matching of $G_0$ is given by the bold edges.

having $\ell$ copies. Our infinite family is inspired by the infinite family of undirected graphs with many minimal feedback vertex sets; here we provide a suitable orientation of such graphs. The generating graph for our infinite family of graphs is given in Fig. 4.28. Let a pair denote an ordered pair $(i, i')$, for $i = 1, \ldots, 5$, where $i$ is a vertex on the outer cycle and $i'$ is the corresponding vertex on the inner cycle. The graph has $5 \cdot 2^4 = 80$ minimal feedback vertex sets containing one vertex from four of the pairs, $5 \cdot 2^2 = 20$ containing one pair and one vertex from each of the opposite pairs and five containing two pairs. In total, it has 105 minimal feedback vertex sets and yields a lower bound of $105^{n/10} > 1.5926^n$ using multiple copies. □

With respect to maximum induced matchings in 3-regular planar graphs $G$, we ask whether the lower bounds of $|E(G)|/9$ and $|V(G)|/6$ can be asymptotically improved if $G$ is connected. The following infinite family of graphs shows that the lower bounds cannot be improved beyond $|E(G)|/6$ and $3|V(G)|/10$, respectively. The generating graph $G_0$ of our infinite family is given in Fig. 4.29, and the $\ell^{th}$ member $G$ of the family consists of $\ell$ copies $G_1, \ldots, G_\ell$ of $G_0$ and edges $v_{i-1}u_i$, for $i = 1, \ldots, \ell + 1 \mod \ell$. Then $G$ has $|E(G)| = 36\ell$ edges and $|V(G)| = 20\ell$ vertices, and any maximum induced matching of $G$ has size $6\ell$. Our construction strengthens a theorem by Zito [253], who proved that in any 3-regular graph $G$ vertices the size of a maximum induced matching is at most $3|V(G)|/10$ and exhibited a family of planar graphs for which this bound is sharp. Our family of graphs shows that this bound is sharp even if $G$ is required to be connected.
Parameterizations Above Guarantee

To study the computational complexity of an NP-hard optimization problem within the framework of Parameterized Complexity, there is a potential range of what is a (mathematically or practically) “interesting” parameter for the problem. Yet some parameters that look interesting at first sight turn out to be of little practical value or small theoretical interest for parameterized complexity.

As an example, consider INDEPENDENT SET restricted to planar graphs, an NP-hard problem [121]. Any planar graph \( G \) has, by the famous Four-Color Theorem [24, 220], a proper 4-vertex-coloring, and such a coloring can be found in quadratic time [220]. The vertices of each color class form an independent set of \( G \); thus, the largest color class is an independent set of \( G \) with size at least \( |V(G)|/4 \). As this lower bound on the solution value increases monotonically and unbounded with the instance size, the parameter “solution size \( k \)” is uninteresting. From the practical point of view, \( k \) will not be small, thwarting the aim of fixed-parameter algorithms of separating the combinatorial explosion from the input size. From the theoretical point of view, \((k)\)-INDEPENDENT SET restricted to planar graphs is trivially fixed-parameter tractable: if \( k \leq |V(G)|/4 \) then \((G, k)\) is a “yes”-instance, and otherwise \((G, k)\) is a kernel with at most \( 4k \) vertices. A more natural parameterized problem is thus to decide whether a planar graph \( G \) contains an independent set of size \( |V(G)|/4 + k \), for parameter \( k \). Whether this problem is fixed-parameter tractable was first asked by Niedermeier [205], and an answer is yet to be found.

In this chapter we study the parameterized complexity of NP-hard maximization problems \( \Pi \) for which every instance \( x \) has a solution of value at least \( b(|x|) \), for some monotonically increasing and unbounded function \( b \). We call the maximum value of \( b(|x|) \) over all instances \( x \) a lower bound for \( \Pi \), and call a lower bound tight if it is best possible for an infinite family of instances \( x \). The parameterized problem that we study is then to decide whether a given pair \((x, k)\), where \( x \) is an instance for \( \Pi \) and \( k \) is a non-negative integer, has a solution of value at least \( b^*(|x|) + k \), where \( b^*(|x|) \) is the tight lower bound on the solution value. We call this problem \( \Pi \)-ABOVE TIGHT LOWER BOUND, or \( \Pi \)-ATLB, and remark that the parameter is only \( k \).

Observe that only parameterizations above tight lower bounds are interesting, because for lower bounds that are not tight the same kernelization trick as above applies. That \( |V(G)|/4 \) is a tight lower bound for INDEPENDENT SET restricted to planar graphs follows from taking \( G \) to be the disjoint union of 4-cliques.
Parameterizations of maximization problems above tight lower bound provide formidable combinatorial puzzles, for it requires understanding which instances are tight. The practical and theoretical importance of such problems was first recognized by Mahajan and Raman [188], who considered Max Sat and Max Cut parameterized above lower bounds. For Max Sat the tight lower bound is \( m/2 \), where \( m \) is the number of clauses, and the problem is to decide whether we can satisfy at least \( m/2 + k \) clauses, where \( k \) is the parameter. Mahajan and Raman proved that this parameterization of Max Sat is fixed-parameter tractable by obtaining a problem kernel with \( O(k) \) variables. Despite clear importance of parameterizations above tight lower bounds, until recently only a few sporadic non-trivial results on the topic were obtained [145, 146, 188, 241].

In this chapter we give fixed-parameter algorithms and kernels, for maximization problems parameterized above tight lower bounds on solution values that monotonically increase with the instance size. The results presented in this chapter, together with the survey of Mahajan et al. [189], greatly renewed interest in such parameterizations. Several questions of their survey were recently answered by newly-developed methods [15, 63, 64, 142, 143], using algebraic, probabilistic and harmonic analysis tools.

We start with independent sets in graphs of maximum degree three, and thereafter continue with ternary permutation constraint satisfaction problems. In particular, we present an advanced probabilistic approach that together with the Hypercontractive Inequality from harmonic analysis allows us to prove the existence of a quadratic kernel for the parameterized Betweenness Above Average problem, thus, answering an open question of Benny Chor [205] from 2006.

5.1 Independent Set in Graphs of Maximum Degree Three

We prove fixed-parameter tractability of Independent Set parameterized above tight lower bound (Independent Set-ATLB) restricted to graphs with maximum degree three, which is an NP-hard problem [164]. In particular, we derive kernels with \( O(k) \) vertices.

We start with graphs of maximum degree three and no other conditions. By letting \( G \) be the disjoint union of 4-cliques, one sees that \( |V(G)|/4 \) is a tight lower bound on the maximum size of an independent set in graphs of maximum degree three.

**Theorem 5.1.** Independent Set-ATLB is fixed-parameter tractable on graphs of maximum degree three and has a kernel with \( 12k \) vertices.

**Proof.** Let \( G \) be a graph with maximum degree three. Then any 4-clique of \( G \) corresponds to a connected component of \( G \). We exhaustively remove any 4-cliques from \( G \). The resulting graph \( G' \) of maximum degree three has an independent set of size at least \( |V(G')|/4 + k \) if and only if \( G \) has an independent set of size at least \( |V(G)|/4 + k \). By Brooks' Theorem [44], \( G' \) has a proper 3-vertex-coloring and so contains an independent set of size at least \( |V(G')|/3 \). Thus, if \( k \leq |V(G')|/12 \) then \( (G',k) \) is a “yes”-instance. Otherwise, \( (G',k) \) is a kernel with \( 12k \) vertices. \( \square \)
5.2 Ordinal Embeddings

We study ordinal embedding relaxations in the realm of parameterized complexity. Specifically, we prove the existence of a quadratic kernel for the Betweenness problem parameterized above tight lower bound, which is stated as follows. For a set \( V \) of variables and set \( B \) of betweenness constraints “\( v_j \) is between \( v_i \) and \( v_k \)”, decide whether there is a bijection from \( V \) to the set \( \{1, \ldots, |V|\} \) satisfying at least \( |B|/3 + k \) of the constraints in \( B \). Betweenness constraints arise in the context of embedding points with measured pairwise distances into a target metric space. The quality of such an embedding can be measured with various objectives; for example isometric embeddings preserve all distances while aiming at low-dimensional target spaces. Yet, for many contexts in nearest-neighbor search, visualization, clustering and compression it is the order of distances rather than the distances themselves that captures the relevant information. The study of such ordinal embeddings dates back to the 1950’s and has recently witnessed a surge in interest [14, 26, 32, 124, 169]. In an ordinal embedding the relative order between pairs of distances must be preserved as much as possible, i.e., one minimizes the relaxation of an ordinal embedding defined as the maximum ratio between two distances whose relative order is inverted by the embedding.

Figure 5.1: The only two connected triangle-free graphs \( G \) of maximum degree three for which \( \alpha(G) = 5|V(G)|/14 \).
Betweenness constraints form partial orders that specify the maximum edge for some triangles, which we aim to embed into a one-dimensional target space. This problem has been studied under the name of Betweenness (problem A12 of Garey and Johnson [121]), which takes a set \( V \) of variables and a set \( B \) of betweenness constraints of the form "\( v_i \) is between \( v_j \) and \( v_k \)" for distinct variables \( v_i, v_j, v_k \in V \). Such a constraint will be written as \((v_i, \{v_j, v_k\})\). The objective is to find a bijection \( \alpha \) from \( V \) to the set \( \{1, \ldots, |V|\} \) that "satisfies" the maximum number of constraints from \( B \), where a constraint \((v_i, \{v_j, v_k\})\) is satisfied by \( \alpha \) if either \( \alpha(v_i) < \alpha(v_j) < \alpha(v_k) \) or \( \alpha(v_k) < \alpha(v_i) < \alpha(v_j) \) holds. Observe that \( \alpha \) is a linear order of \( V \).

Such linear orders are of significant interest in molecular biology, where for example markers on a chromosome need to be linearly ordered as to satisfy the maximum number of constraints [62, 126]. More theoretical interest comes from the constraint programming framework with unbounded domains and interval graph recognition [178].

Of the computational complexity of Betweenness, the following is known. For a set \( B \) of betweenness constraints over variable set \( V \), deciding if all constraints from \( B \) can be satisfied by some linear order of \( V \) is NP-complete [207]. Hence, Betweenness is NP-hard. It was further shown that finding a linear order of \( V \) satisfying a \( 1 - \varepsilon \) fraction of the constraints in \( B \) is NP-hard for all \( \varepsilon \in (0, 1/48) \) [59]. Any uniformly-at-random permutation of \( V \) satisfies at least one-third of all constraints in \( B \), and this fraction is tight: take \( B \) to be the set of all possible 3 triangulations, and observe that a uniformly-at-random permutation of \( V \) "satisfies" \( 3(|V|)/3 \) constraints over \( V \). This yields a randomized 1/3-approximation for Betweenness, and this approximation factor is best-possible under the Unique Games Conjecture [52]. There are polynomial-time algorithms that find a linear order of \( V \) satisfying at least half of the constraints from \( B \), assuming that some linear order satisfies all constraints from \( B \) [59, 190].

Parameterizing Betweenness by the number \( k \) of satisfiable constraints in an instance \((V, B)\) is of little interest, because a uniformly-at-random permutation of the variables in \( V \) satisfies at least one-third of the constraints in \( B \) in expectation. Instead, we parameterize Betweenness above the tight lower bound of \(|B|/3\), meaning that we ask for a linear order of \( V \) that satisfies at least \(|B|/3 + k\) of the constraints. We call this problem Betweenness Above Average (Betweenness-AA), as the average number of satisfied constraints equals one-third of the total number of constraints. The parameterized complexity of Betweenness-AA was first asked by Benny Chor, and was stated as an open problem by Niedermeier [205]. Since deciding if all constraints are satisfiable is an NP-complete problem, the complementary question of whether all but \( k \) constraints are satisfiable by some linear order of \( V \) is not fixed-parameter tractable, unless \( P = NP \). For the special case of a dense set of constraints, containing a constraint for each 3-subset of variables, a subexponential fixed-parameter algorithm was recently obtained [18].

In this section we settle Benny Chor’s question [205, p. 43] about the parameterized complexity of Betweenness-AA. Our main result is that Betweenness-AA is fixed-parameter tractable and has a quadratic kernel. The kernel is obtained via a nontrivial extension of the recently introduced probabilistic Strictly Above-Below Expectation Method [143]. In fact, work on this problem was a trigger to the development of that method. By now, applications of this method solved several open problems from the literature, proving for instance fixed-parameter tractibil-
ity of Binary Linear Ordering [143], Max r-SAT [15] and special cases of Max Lin-2 [63, 64, 143] parameterized above tight lower bounds.

We describe the method briefly in Section 5.2.1 and point out how to extend it in order to obtain a quadratic kernel for Betweenness-AA. We extended the method to handle that feasible solutions for Betweenness-AA are permutations of variables; see Subsection 5.2.1 for a high-level discussion and Subsection 5.2.2 for details.

5.2.1 Strictly Above/Below Expectation Method

The Strictly Above/Below Expectation method [143] allows us to prove fixed-parameter tractability of maximization (minimization) problems \( \Pi \) parameterized above (below) tight lower (upper) bounds on the solution value. In that method, we first apply polynomial-time reductions rules to reduce the given instance \((x,k)\) of a restricted version \(\Pi'\) of \(\Pi\). Then we introduce a random variable \(X\) such that \((x',k')\) is a “yes”-instance for \(\Pi'\) if and only if \(X\) takes with positive probability a value greater than or equal to \(k\). If \(X\) happens to be a symmetric random variable then the simple inequality \(P(X \geq \sqrt{E[X^2]}) > 0\) can be useful; here \(P(\cdot)\) and \(E[\cdot]\) denote probability and expectation, respectively. Applications are Binary Linear Ordering and a special case of Max Lin-2 parameterized above tight lower bounds on their respective solution values [143]. However, it is often difficult to construct \(X\) to be symmetric. In many such cases the following result by Alon et al. [16] is of use, a weaker version of which was obtained by Håstad and Venkatesh [152].

**Proposition 5.1** ([16]). Let \(X\) be a real random variable whose first, second and fourth moments satisfy \(E[X] = 0, E[X^2] = \sigma^2 > 0\) and \(E[X^4] \leq b\sigma^4\), respectively. Then \(P(X > \frac{\sigma}{4\sqrt{b}}) \geq \frac{1}{32\pi} \).

We will combine this result with the following extension of Khinchin’s Inequality by Bourgain [43].

**Proposition 5.2** ([143]). Let \(f = f(x_1, \ldots, x_n)\) be a polynomial of degree \(r\) in \(n\) variables \(x_1, \ldots, x_n\) with domain \(\{-1, 1\}\). Choose a vector \((\epsilon_1, \ldots, \epsilon_n) \in \{-1, 1\}^n\) uniformly at random and define a random variable \(X\) by \(X = f(\epsilon_1, \ldots, \epsilon_n)\). Then \(E[X^4] \leq 26rE[X^2]^2\).

These two propositions were used by Alon et al. [15] and Gutin et al. [143] to prove fixed-parameter tractability of Max r-SAT and a special case of Max Lin-2 parameterized above tight lower bounds. Yet it appears impossible to introduce a random variable \(X\) for which \(P(X \geq k) > 0\) if and only if the given pair \((x,k)\) is a “yes”-instance for Betweenness-AA and such that \(X\) is either symmetric or satisfies the conditions of Proposition 5.2. Thus, in the next section, we introduce a random variable \(X\) for which we have a weaker property with respect to Betweenness-AA: if \(P(X \geq k) > 0\) then the given pair \((x,k)\) is a “yes”-instance for Betweenness-AA. This \(X\), however, satisfies the conditions of Proposition 5.2.

To apply Proposition 5.1, we need to evaluate \(E[X^2]\). While such evaluations for Max r-SAT-ATLB [15] and a special case of Max Lin-2-ATLB [143] are rather straightforward, our evaluation of \(E[X^2]\) is quite involved and requires assistance of a computer. Besides, with \(X\) only satisfying the weaker property, for some “yes”-instances of Betweenness-AA it may hold that \(P(X \geq k) = 0\). This prevents us
from algebraically simplifying $X$ as was done by Alon et al. [15] for Max $r$-SAT-ATLB, in order to ease the computation of $\mathbb{E}[X^2]$.

### 5.2.2 Quadratic Kernel for Betweenness Above Tight Lower Bound

We will now show fixed-parameter tractability of Betweenness-AA. In fact, we will prove the stronger statement that this problem has a quadratic kernel.

For a constraint $B$ of $\mathcal{B}$ let $vars(B)$ denote the set of variables in $B$. Call a triple $B, B', B''$ of distinct betweenness constraints complete if $vars(B) = vars(B') = vars(B'')$. Since every linear order of $V$ satisfies exactly one constraint in each complete triple, we have the following reduction rule.

**Lemma 5.1.** Let $\mathcal{B}$ be a set of betweenness constraints over variables $V$ and let $(V', B')$ be obtained by removing all complete triples from $\mathcal{B}$ and all variables from $V$ appearing in no constraint of $B'$. Then $(V, B, k)$ is a "yes"-instance for Betweenness-AA if and only if $(V', B', k')$ is, for any integer $k' \geq 0$.

Call an instance $(V, B, k)$ of Betweenness-AA reduced if it does not contain a complete triple. Using Lemma 5.1, any instance can be transformed into a reduced one in time $O(|B|^3)$.

Let $(V, B, k)$ be a reduced instance of Betweenness-AA. For a random function $\varphi : V \rightarrow \{0, 1, 2, 3\}$, let $\ell_i(\varphi)$ be the number of variables in $V$ that are mapped by $\varphi$ to $i$, for $i \in \{0, \ldots, 3\}$. Now obtain a bijection $\alpha : V \rightarrow \{1, \ldots, |V|\}$ by uniformly-at-random assigning values $1, \ldots, \ell_0(\varphi)$ to all $\alpha(v)$ for which $\varphi(v) = 0$, and values $\sum_{\ell=0}^{j-1} \ell_i(\varphi) + 1, \ldots, \sum_{\ell=0}^{|V|} \ell_i(\varphi)$ to all $\alpha(v)$ for which $\varphi(v) = j$ for every $j = 1, 2, 3$. A linear order $\alpha$ of $V$ obtained in this way is called a $\varphi$-compatible. It is easy to see that $\alpha$ obtained in this two stage process is, in fact, a random linear order of $V$, but this fact is not going to be used here.

Fix a function $\varphi : V \rightarrow \{0, 1, 2, 3\}$ and let $\alpha$ be a $\varphi$-compatible linear order of $V$. For each constraint $B \in \mathcal{B}$, define a binary variable $x_B(\alpha)$ taking value 1 if $B$ is satisfied by $\alpha$ and value 0 otherwise. Let $X_B = \mathbb{E}[x_B(\alpha)] - 1/3$ for all $B \in \mathcal{B}$ and let $X = \sum_{B \in \mathcal{B}} X_B$. Observe that if $\varphi$ is a random function from $V$ to $\{0, 1, 2, 3\}$ then $X$ and $X_B$, for all $B \in \mathcal{B}$, are random variables.

**Lemma 5.2.** If $X \geq k$ then $(V, B, k)$ is a "yes"-instance of Betweenness-AA.

**Proof.** By linearity of expectation, $X \geq k$ implies $\mathbb{E}[\sum_{B \in \mathcal{B}} x_B(\alpha)] \geq |\mathcal{B}|/3 + k$. Thus, if $X \geq k$ then there is a $\varphi$-compatible linear order $\alpha$ that satisfies at least $|\mathcal{B}|/3 + k$ constraints.

**Lemma 5.3.** $\mathbb{E}[X] = 0$.

**Proof.** Let $B = (v, \{v_j, v_k\}) \in \mathcal{B}$; then $X_B$ obeys the probability distribution given in Table 5.1. It is easy to see that $\mathbb{E}[X_B] = 0$ and, thus, $\mathbb{E}[X] = \sum_{B \in \mathcal{B}} \mathbb{E}[X_B] = 0$.

**Lemma 5.4.** The random variable $X$ can be expressed as a polynomial of degree 6 in independent uniformly distributed random variables with values $-1$ and 1.
5.2. ORDINAL EMBEDDINGS

| \{ \varphi(v_i), \varphi(v_j), \varphi(v_k) \} | \begin{array}{lll}
| 0 | 0 & 1/16 \\
| 1 | \varphi(v_i) = \varphi(v_j) = \varphi(v_k) & 0 & 1/16 \\
| 2 | \varphi(v_i) \neq \varphi(v_j) = \varphi(v_k) & -1/3 & 3/16 \\
| 2 | \varphi(v_i) \in \{ \varphi(v_j), \varphi(v_k) \} & 1/6 & 6/16 \\
| 3 | \varphi(v_i) \text{ between } \varphi(v_j) \text{ and } \varphi(v_k) & 2/3 & 2/16 \\
| 3 | \varphi(v_i) \text{ not between } \varphi(v_j) \text{ and } \varphi(v_k) & -1/3 & 4/16
\end{array} |

Table 5.1: Probability distribution of the random variable $X_B$ that corresponds to constraint $B = (v_i, \{v_j, v_k\})$.

Proof. For each constraint $B = (v_i, \{v_j, v_k\}) \in \mathcal{B}$ define variables

$$
\epsilon_i^1 = \begin{cases} -1, & \text{if } \varphi(v_i) \in \{0, 1\}, \\ +1, & \text{if } \varphi(v_i) \in \{2, 3\}, \end{cases} \quad \epsilon_i^2 = \begin{cases} -1, & \text{if } \varphi(v_i) \in \{0, 2\}, \\ +1, & \text{if } \varphi(v_i) \in \{1, 3\}. \end{cases}
$$

Similarly, define variables $\epsilon_1^1, \epsilon_2^1, \epsilon_1^3, \epsilon_2^3$. Then $\epsilon_i^1 \epsilon_i^2$ can be seen as a binary representation of a number from the set $\{0, 1, 2, 3\}$ and $\epsilon_1^1 \epsilon_1^2 \epsilon_2^1 \epsilon_2^2 \epsilon_2^3 \epsilon_2^3$ can be seen as a binary representation of a number from the set $\{0, 1, \ldots, 63\}$, where $-1$ plays the role of 0. Then $X_B$ can be written as the polynomial

$$
\frac{1}{64} \sum_{q=0}^{63} (-1)^{s_q} w_q \cdot (\epsilon_1^1 + c_1^q)(\epsilon_1^2 + c_2^q)(\epsilon_1^3 + c_1^q)(\epsilon_2^1 + c_2^q)(\epsilon_2^2 + c_1^q)(\epsilon_2^3 + c_2^q),
$$

where $c_1^q, c_2^q, c_1^q, c_2^q, c_1^q, c_2^q$ is the binary representation of $q$, $s_q$ is the number of digits equal $-1$ in this representation, and $w_q$ equals the value of $X_B$ for the case when the binary representations of $\varphi(v_i), \varphi(v_j)$ and $\varphi(v_k)$ are $c_1^q, c_2^q, c_1^q$ and $c_1^q, c_2^q$, respectively. The actual values for $X_B$ for each case are given in the proof Lemma 5.3. The above polynomial is of degree 6. The proof is completed by recalling that $X = \sum_{B \in \mathcal{B}} X_B$. \hfill \square

Lemma 5.5. Reduced instances $(\mathcal{V}, \mathcal{B}, k)$ of BETWEENNESS-AA satisfy $\mathbb{E}[X^2] \geq \frac{11}{768} |\mathcal{B}|$.

Proof. First, observe that $\mathbb{E}[X^2] = \sum_{B \in \mathcal{B}} \mathbb{E}[X_B^2] + \sum_{B, B' \in \mathcal{B}, B \neq B'} \mathbb{E}[X_B X_{B'}]$. We compute $\mathbb{E}[X_B^2]$ and $\mathbb{E}[X_B X_{B'}]$ separately.

Using the distribution of $X_B$ give in Table 5.1, it is easy to see that $\mathbb{E}[X_B^2] = 11/96 = 88/768$. It remains to show that

$$
\sum_{B, B' \in \mathcal{B}, B \neq B'} \mathbb{E}[X_B X_{B'}] \geq -\frac{77}{768} |\mathcal{B}|. \quad (5.1)
$$
Indeed, (5.1) and \(E[X_B^2] = 88/768\) imply that

\[
E[X^2] = \sum_{B \in \mathcal{B}} E[X_B^2] + \sum_{B, B' \in \mathcal{B}, B \neq B'} E[X_B X_{B'}] \geq \frac{88}{768} |\mathcal{B}| - \frac{77}{768} |\mathcal{B}| = \frac{11}{768} |\mathcal{B}|
\]

In the reminder of this proof we show that (5.1) holds. Let \(B, B'\) be a pair of distinct constraints of \(\mathcal{B}\). To evaluate \(E[X_B X_{B'}]\), we consider several cases. A simple case is when the sets \(\text{vars}(B)\) and \(\text{vars}(B')\) are disjoint: then \(X_B\) and \(X_{B'}\) are independent random variables and, thus, \(E[X_B X_{B'}] = E[X_B]E[X_{B'}] = 0\). Let \(U = \{(B, B') \mid B, B' \in \mathcal{B}, B \neq B'\}\) be the collection of ordered pairs of distinct constraints in \(\mathcal{B}\), and let

\[
S_1(u) = \{(B, B') \in U \mid B = (u, \{a, b\}), B' = (u, \{c, d\}), a, b, c, d \in V\}
\]

\[
S_2(u) = \{(B, B') \in U \mid B = (a, \{u, b\}), B' = (c, \{u, d\}), a, b, c, d \in V\}
\]

\[
S_3(u) = \{(B, B'), (B', B) \in U \mid B = (u, \{a, b\}), B' = (c, \{u, d\}), a, b, c, d \in V\}
\]

\[
S_4(u, v) = \{(B, B') \in U \mid B = (u, \{v, a\}), B' = (u, \{v, b\}), a, b \in V\}
\]

\[
\cup \{(B, B') \in U \mid B = (v, \{u, a\}), B' = (v, \{u, b\}), a, b \in V\}
\]

\[
S_5(u, v) = \{(B, B') \in U \mid B = (a, \{u, v\}), B' = (b, \{u, v\}), a, b \in V\}
\]

\[
S_6(u, v) = \{(B, B') \in U \mid B = (u, \{v, a\}), B' = (b, \{v, u\}), a, b \in V\}
\]

\[
\cup \{(B, B'), (B', B) \in U \mid B = (v, \{u, a\}), B' = (v, \{u, b\}), a, b \in V\}
\]

\[
S_7(u, v) = \{(B, B') \in U \mid B = (u, \{v, a\}), B' = (v, \{u, b\}), a, b \in V\}
\]

\[
S_8(u, v, w) = \{(B, B') \in U \mid \text{vars}(B) = \text{vars}(B') = \{u, v, w\}\}
\]

Let \(u, v, w \in V\) be distinct variables; then

\[
S_4(u, v) = (S_1(u) \cap S_2(v)) \cup (S_1(v) \cap S_2(u))
\]

\[
S_5(u, v) = S_2(u) \cap S_2(v)
\]

\[
S_6(u, v) = (S_3(u) \cap S_2(v)) \cup (S_3(v) \cap S_2(u))
\]

\[
S_7(u, v) = S_3(u) \cap S_3(v)
\]

\[
S_8(u, v, w) = (S_3(u) \cap S_3(v) \cap S_3(w)) \cup (S_3(v) \cap S_3(w) \cap S_2(u)) \cup (S_3(w) \cap S_3(u) \cap S_2(v))
\]

For each variable \(u \in V\), define \(b(u) = |\{B = (u, \{a, b\}) \mid a, b \in V\}|\) and \(e(u) = |\{B = (a, \{u, b\}) \mid a, b \in V\}|\). Then \(|S_1(u)| = b(u)(b(u) - 1), |S_2(u)| = e(u)(e(u) - 1)\) and \(|S_3(u)| = 2b(u)e(u)|\).

For each pair \(\{u, v\} \subseteq V\) of distinct variables, define \(c_{uv}^a = |\{B = (u, \{v, a\}) \mid a \in V\}|\).
5.2. ORDINAL EMBEDDINGS

We will briefly describe how our program computes 768E[X_B X_{B'}], 768w'.

Table 5.2: Data for sets {u, v} for i = 1, 2, ..., 8.

| Set       | Union/intersection | | Set | 768E[X_B X_{B'}] | 768w' |
|-----------|-------------------|---|----------------|-----------------|
| S_1(u)   | -                 | |    b(u)(b(u) - 1) | 12 = w_1       | 12               |
| S_2(u)   | -                 | |    e(u)(e(u) - 1) | 3 = w_2        | 3                |
| S_3(u)   | (S_1(u) ∩ S_2(v)) | |    b(u)e(u) + e(u)b(u) | -6 = w_3     | -6               |
| S_4(u, v)| (S_4(u, v) ∩ S_2(v)) | |    c^u_v(c^u_v - 1) + c^u_w(c^u_w - 1) | 24 = w_4     | 9                |
| S_5(u, v)| S_2(u) ∩ S_2(v)   | |    c_{uw}c_{uv} - 1 | 36 = w_5      | 30               |
| S_6(u, v)| (S_3(u) ∩ S_2(v)) | |    2(c^u_v + c^w_u) · c_{uw} | -18 = w_6    | -15              |
| S_7(u, v)| S_3(u) ∩ S_3(v)   | |    2c^w_{uv}c^w_{uw} | -6 = w_7      | 6                |
| S_8(u, v, w)| see (5.2) | |    ≤ 2 | -44 = w_8 | -11             |

V} and c_{uv} = |{B = (a, {u, v}) | a ∈ V}. Then

|S_4(u, v)| = c^u_v(c^u_v - 1) + c^u_w(c^u_w - 1),
|S_5(u, v)| = c_{uw}(c_{uv} - 1),
|S_6(u, v)| = 2(c^u_v + c^w_u) · c_{uw},
|S_7(u, v)| = 2c^w_{uv}c^w_{uw}.

For each tuple {u, v, w} ⊆ V of distinct variables, the number of ordered pairs (B, B') for which vars(B) = vars(B') = {u, v, w} is at most 2, that is, |S_8(u, v, w)| ≤ 2, since (V, B, k) is reduced.

We summarize the data obtained so far in Table 5.2. We list the sets S_i(·), their union/intersection forms (for i = 4, 5, 6, 7) and their sizes in Table 5.2. If (B, B') belongs to some S_i but to no S_j for j > i, then Table 5.2 also contains the value 768E[X_B X_{B'}], in the row corresponding to S_i. These values cannot be easily calculated analytically as there are many cases to consider and we have calculated them using a computer. The Python program written for that purpose is given in Appendix A. We will briefly describe how our program computes 768E[X_B X_{B'}] using as an example the case (B, B') ∈ S_i(u), that is, B = (u, {a, b}), B' = (u, {c, d}). For each (q_1, q_2, q_3, q_4, q_5) ∈ {0, 1, 2, 3}^5 the probability of (u, a, b, c, d) = (q_1, q_2, q_3, q_4, q_5) is 4^{-5} and the corresponding value of X_B X_{B'} can be found in Table 5.2.

We are ready to compute a lower bound on the term ∑_{B, B' ∈ B, B' ≠ B} E[X_B X_{B'}]. To that end, we define the values w_i' for i = 1, 2, ..., 8 as it is done in Table 5.2, and show the following to hold (note that the sets we sum over have to contain distinct
elements):

\[
\sum_{B,B' \in B; B \neq B'} \mathbb{E}[X_B X_{B'}] = \sum_{u \in V} \frac{3}{4} |S_i(u)| w'_i + \sum_{\{u,v\} \subseteq V} 7 |S_i(u,v)| w'_i + \sum_{\{u,v,w\} \subseteq V} |S_8(u,v,w)| w'_8 \tag{5.3}
\]

To show (5.3) we consider the possible cases for \((B, B') \in U\).

**Case 1:** \(|\text{vars}(B) \cap \text{vars}(B')| = 0\). In this case \(\mathbb{E}[X_B X_{B'}] = 0\) and the pair \((B, B')\) does not belong to any \(S_i\) and therefore contributes zero to the right-hand side of (5.3).

**Case 2:** \(|\text{vars}(B) \cap \text{vars}(B')| = 1\). Each pair \((B, B') \in S_1(u)\) contributes \(\frac{12}{768}\) to both sides of (5.3), as in this case \((B, B')\) does not belong to any \(S_i\) with \(j > 1\). Analogously if \((B, B') \in S_2(u)\) then it contributes \(\frac{3}{768}\) to both sides of (5.3). Furthermore if \((B, B') \in S_3(u)\) then it contributes \(-\frac{6}{768}\).

**Case 3:** \(|\text{vars}(B) \cap \text{vars}(B')| = 2\). Consider a pair \((B, B') \in S_4(u,v)\) and assume, without loss of generality, that \((B, B') \subseteq S_1(u) \cap S_2(v)\). Note that \((B, B')\) contributes \(\frac{24}{768}\) to the left-hand side of (5.3) and it contributes \(w'_i + w'_j + w'_k = \frac{24}{768}\) to the right-hand side, as \((B, B') \subseteq S_1(u) \cap S_2(v) \cap S_4(u,v)\). Analogously if \((B, B') \in S_5(u,v)\) we get a contribution of \(w_5 = \frac{36}{768}\) to both sides of (5.3). If \((B, B') \in S_6(u,v)\) we get a contribution of \(w_6 = -\frac{18}{768}\) to both sides of (5.3). If \((B, B') \in S_7(u,v)\) we get a contribution of \(w_7 = -\frac{6}{768}\) to both sides of (5.3).

**Case 4:** \(|\text{vars}(B) \cap \text{vars}(B')| = 3\). Assume, without loss of generality, that \((B, B') \subseteq S_3(u) \cap S_3(v) \cap S_2(w)\) and note that \((B, B') \in S_7(u,v) \cap S_6(u,w) \cap S_6(v,w)\). Therefore we get a contribution of \(w_8 = -\frac{44}{768}\) to both sides of (5.3).

Therefore, (5.3) holds. We conclude that

\[
\sum_{B,B' \in B; B \neq B'} \mathbb{E}[X_B X_{B'}]
= \sum_{u \in V} \frac{3}{4} |S_i(u)| w'_i + \sum_{\{u,v\} \subseteq V} 7 |S_i(u,v)| w'_i + \sum_{\{u,v,w\} \subseteq V} |S_8(u,v,w)| w'_8
= \frac{1}{2 \cdot 768} \sum_{u \in V} (6(2b(u) - e(u))^2 - 24b(u) - 6e(u))
+ \frac{1}{2 \cdot 768} \sum_{\{u,v\} \subseteq V} \left(15(c^v_u + c^v_w - 2c_{uw})^2 + 12 \left(\frac{c^v_u - c^v_w}{2}\right)^2 - 18(c^v_u + c^v_w) - 60c_{uw}\right)
+ \sum_{\{u,v,w\} \subseteq V} |S_8(u,v,w)| w'_8
\]

To complete the proof of the lemma it remains to translate this sum into a function on the number of constraints. In that respect, notice that \(\sum_{u \in V} b(u) = |B|\) and \(\sum_{u \in V} e(u) = 2|B|\). Further, each constraint \(\{u, \{v, w\}\}\) contributes exactly one unit
to each of $c_u^h$ and $c_{uv}$ as well as exactly one unit to $c_{vu}$. Hence $\sum_{\{u,v\}\subset V} (c_u^h + c_{uv}) = 2|B|$ and $\sum_{\{u,v\}\subset V} c_{uv} = |B|$. Since $(V, B, k)$ is reduced, the number of ordered pairs $(B, B')$ for which $\text{vars}(B) = \text{vars}(B')$ is at most $|B|/2$ and, thus,

$$\sum_{\{u,v,w\}\subset V} |S_8(u,v,w)|w_8^h \leq |B| \cdot w_8^h.$$  

Together these bounds imply that

$$\sum_{B, B' \in B, B \neq B'} E[X_B X_B'] \geq -\frac{36}{2 \cdot 768} |B| - \frac{96}{2 \cdot 768} |B| - \frac{11}{768} |B| = -\frac{77}{768} |B|$$

and (5.1) holds. \qed

We are ready to prove the main result of this section.

**Theorem 5.3.** Betweenness-AA is fixed-parameter tractable and has a kernel of size $O(k^2)$.

**Proof.** Let $(V, B, k)$ be an instance of Betweenness-AA. By Lemma 5.1, in time $O(|B|^3)$ we obtain a reduced instance $(V', B', k)$ such that $(V, B, k)$ is a “yes”-instance for Betweenness-AA if and only if $(V', B', k)$ is. For the instance $(V', B', k)$, let $X$ be the random variable defined as above. Then $X$ is expressible as a polynomial of degree 6 by Lemma 5.4; hence it follows from Proposition 5.2 that $E[X^4] \leq 2^{36}E[X^2]^2$. Consequently, $X$ satisfies the conditions of Lemma 5.1, from which we conclude in combination with Lemma 5.5 that $P \left( X > \frac{1}{4 \cdot 768} \sqrt{\frac{11}{768}|B'|} \right) > 0$. By Lemma 5.2 if $\frac{1}{4 \cdot 768} \sqrt{\frac{11}{768}|B'|} \geq k$ then $(V', B')$ is a “yes”-instance for Betweenness-AA. Otherwise, $|B'| = O(k^2)$ and $(V', |B'|, k)$ is a kernel of size $O(k^2)$. \qed

The result of Theorem 5.3 can be made constructive: we sketch how to find a linear order $\alpha'$ of a set $V'$ in a reduced instance $(V', B', k)$ such that $\alpha'$ satisfies at least $|B'|/3 + \frac{1}{4 \cdot 768} \sqrt{\frac{11}{768}|B'|}$ constraints from $B'$. First, find a function $\varphi : V \to \{0,1,2,3\}$ such that there is a $\varphi$-compatible linear order $\alpha'$ of $V'$ with $X = X(\alpha') \geq c \sqrt{|C|}$, for appropriate constant $c$. This can be done similar to our estimate of $E[X^2]$ by trying $\varphi(u) = 0, \ldots, \varphi(u) = 3$ for each $u \in V$ one by one and checking which one is better on average. Second, fix the found function $\varphi$ and find an optimal $\varphi$-compatible linear order $\alpha'$ of $V'$ by checking whether $u < v$ or $v < u$ is better for $X$ on average.

We finish this section by answering the following natural question: why have we considered functions $\varphi : V \to \{0,1,2,3\}$ rather than functions $\varphi : V \to \{0,1\}$? The latter would involve less computations and give a smaller degree of the polynomial representing $X$. The reason is that our proof of Lemma 5.5 does not work for functions $\varphi : V \to \{0,1\}$ (we would only be able to prove that $E[X^2] \geq \sum_{\{u,v\}\subset V} [c_u^h + c_{uv} - 2c_{uv}]^2$, which is not enough).
5.3 Generalization to Ternary Permutation Constraint Satisfaction Problems

We extend the results of the preceding section to all ternary permutation constraint satisfaction (CSP) problems. A ternary Permutation-CSP is specified by a subset $\Pi$ of the symmetric group $S_3$. An instance of such a problem consists of a set of variables $V$ and a multiset of constraints, which are totally ordered triples of distinct variables of $V$. The objective is to find a linear order $\alpha$ of $V$ that maximizes the number of triples whose order under $\alpha$ follows a permutation in $\Pi$. Important special cases are Betweenness [52, 124, 142, 207] and Circular Ordering [120, 125], which find applications in circuit design and computational biology [59, 207], and in qualitative spatial reasoning [161], respectively.

Our main result in this section is that all ternary Permutation-CSPs parameterized above average (AA) have kernels with a quadratic number of variables. This result is obtained by first reducing all the problems to just one, Linear Ordering-AA, then showing that Linear Ordering-AA has a quadratic kernel and, thus, concluding that there is a quadratic bikernel from each of the problems AA to Linear Ordering-AA. Finally, using a result on bikernels by Alon et al. [15] (or, alternatively, polynomial parameter transformations [34]), we obtain that all the problems AA have kernels with a quadratic number of variables.

By far the most difficult part of this chain of arguments is the proof that Linear Ordering-AA has a quadratic kernel. It is impossible to show this result similarly to that for Betweenness-AA: to eliminate all instances of Linear Ordering-AA whose optimal solution coincides with the lower bound, we need an infinite number of “normal” reduction rules, whereas for Betweenness-AA, only one reduction rule is required—see Section 5.2. So, determining fixed-parameter tractability of Linear Ordering-AA turns out to be much harder than that of Betweenness-AA. Fortunately, we found a non-trivial way of reducing Linear Ordering-AA to a combination of Betweenness-AA and Acyclic Subdigraph-AA, a binary Permutation-CSP. Using further probabilistic and deterministic arguments for the mixed problem, we prove that Linear Ordering-AA has a quadratic kernel.

5.3.1 Ternary Permutation-CSPs

Let $V$ be a set of variables. The symmetric group on three elements is $S_3 = \{(123), (132), (213), (231), (312), (321)\}$. A constraint set over $V$ is a multiset $C$ of constraints, which are permutations of three distinct elements of $V$. A constraint $(a, b, c) \in C$ is satisfied by a linear order $\alpha$ of $V$ if $\alpha(a) < \alpha(b) < \alpha(c)$. For each subset $\Pi \subseteq S_3$ and a linear order $\alpha$ of $V$, a constraint $(v_1, v_2, v_3) \in C$ is $\Pi$-satisfied by $\alpha$ if there is a permutation $\pi \in \Pi$ such that $(v_{\pi(1)}, v_{\pi(2)}, v_{\pi(3)})$ is satisfied by $\alpha$.

For each subset $\Pi \subseteq S_3$, the problem $\Pi$-CSP is to decide whether for a given pair $(V, C)$ of variables and constraints there is a linear order $\alpha$ of $V$ that $\Pi$-satisfies all constraints in $C$. A complete dichotomy of problems $\Pi$-CSP with respect to their computational complexity was given by Guttmann and Maucher [148]. For that, they reduced $2^{|S_3|} = 64$ problems by two types of symmetry. First, two problems differing just by a consistent renaming of the elements of their permutations are of the same complexity. Second, two problems differing just by reversing their
5.3. TERNARY PERMUTATION-CSPS

and Maicher [148], see Table 5.3.

Theorem 5.4. For $i = 0, 1, 2, 3$, problem $\text{Max } \Pi_i$-CSP from Table 5.3 is NP-hard.

Proof. We will consider the four cases one by one.

$i = 0$: Theorem 5.5 will imply, in particular, that $\text{Max-Betweenness}$ can be reduced to Max-$\Pi_0$-CSP. Thus, Max-$\Pi_0$-CSP is NP-hard.

$i = 1$: Denote constraints of $\text{Max-} \Pi_1$-CSP by $(u < v, w)$. This constraint is $\Pi_1$-satisfied by a linear order $a$ of $\{u, v, w\}$ if and only if $a(u) < \min\{a(v), a(w)\}$.

From an instance $(D, k)$ of $(k)$-ACYCLIC SUBDAGRAPH, construct an instance $(V, C, k)$ of (a decision version of) $\text{Max-} \Pi_1$-CSP by setting $V = V(D) \cup \{z\}$ and, for each arc $(u, v) \in A(D)$, adding $(u < v, z)$ to $C$. Observe that, without loss of generality, an optimal linear order of $(V, C, k)$ has $z$ at the end as if it does not then moving $z$ to the end does not falsify any constraints. Therefore $(u, v)$ is satisfied in $D$ if and only if $(u < v, z)$ is $\Pi_1$-satisfied in $(V, C, k)$.

$i = 2$: Denote constraints of $\text{Max-} \Pi_2$-CSP by $(w; u, v)$. Such a constraint is $\Pi_2$-satisfied by a linear order $a$ of $\{u, v, w\}$ if and only if $a(u) < a(v)$.

From an instance $(D, k)$ of $(k)$-ACYCLIC SUBDAGRAPH, construct an instance $(V, C, k)$ of (a decision

<table>
<thead>
<tr>
<th>$\Pi \subseteq S_3$</th>
<th>Problem name</th>
<th>Complexity to satisfy all constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Pi_0 = {(123)}$</td>
<td>LINEAR ORDERING</td>
<td>polynomial</td>
</tr>
<tr>
<td>$\Pi_1 = {(123), (132)}$</td>
<td>polynomial</td>
<td></td>
</tr>
<tr>
<td>$\Pi_2 = {(123), (213), (231)}$</td>
<td>polynomial</td>
<td></td>
</tr>
<tr>
<td>$\Pi_3 = {(132), (231), (312), (321)}$</td>
<td>polynomial</td>
<td></td>
</tr>
<tr>
<td>$\Pi_4 = {(123), (231)}$</td>
<td>$\text{NP-complete}$</td>
<td></td>
</tr>
<tr>
<td>$\Pi_5 = {(123), (321)}$</td>
<td>BETWEENNESS</td>
<td>$\text{NP-complete}$</td>
</tr>
<tr>
<td>$\Pi_6 = {(123), (132), (231)}$</td>
<td>$\text{NP-complete}$</td>
<td></td>
</tr>
<tr>
<td>$\Pi_7 = {(123), (231), (312)}$</td>
<td>$\text{NP-complete}$</td>
<td></td>
</tr>
<tr>
<td>$\Pi_8 = S_3 \setminus {(123), (231)}$</td>
<td>$\text{NP-complete}$</td>
<td></td>
</tr>
<tr>
<td>$\Pi_9 = S_3 \setminus {(123), (321)}$</td>
<td>$\text{NP-complete}$</td>
<td></td>
</tr>
<tr>
<td>$\Pi_{10} = S_3 \setminus {(123)}$</td>
<td>INTERMEZZO</td>
<td>$\text{NP-complete}$</td>
</tr>
</tbody>
</table>

Table 5.3: Ternary Permutation-CSPs $\text{Max-} \Pi_i$-CSP, after symmetry considerations, and the computational complexity of $\Pi_i$-CSP.
version of) \textsc{Max-$\Pi_2$-CSP} by setting \( V = V(D) \cup \{ z \} \) and, for each arc \((u,v) \in A(D)\), adding constraint \((z;u,v)\) to the constraint set \( C \). Observe that \( D \) has a set of \( k \) arcs that form an acyclic directed subgraph if and only if there are \( k \) constraints in \( C \) that can be \( \Pi_2 \)-satisfied by a linear order of \( V \). Thus, we have reduced \((k)\)-\textsc{Acyclic Subdigraph} to \textsc{Max-$\Pi_2$-CSP}, implying that \textsc{Max-$\Pi_2$-CSP} is \( \mathsf{NP} \)-hard.

\( i = 3 \): Denote constraints of \textsc{Max-$\Pi_3$-CSP} by \((u,v \not< w)\). This constraint is \( \Pi_3 \)-satisfied by a linear order \( \alpha \) if and only if \( w \) is not the last element among \( u,v,w \) in \( \alpha \). Now consider an instance \((V,C_3,k)\) of \textsc{Max-$\Pi_3$-CSP}, which we have shown to be \( \mathsf{NP} \)-hard. For each constraint \((u < v,w)\) in \( C_3 \) add \((u,v \not< w)\) and \((u,w \not< v)\) to \( C_3 \). We will show that \((V,C_3,k)\) is equivalent to \((V,C_3,|C_3| + k)\), which is an instance of \textsc{Max-$\Pi_3$-CSP}. Let \( \alpha \) be any linear order of \( V \) and let \( \alpha' \) be the reverse linear order. Note that \((u < v,w)\) is \( \Pi_3 \)-satisfied by \( \alpha \) if and only if both \((u,v \not< w)\) and \((u,w \not< v)\) are \( \Pi_3 \)-satisfied by \( \alpha' \). Furthermore one of \((u,v \not< w)\) and \((u,w \not< v)\) is always \( \Pi_3 \)-satisfied. Therefore, at least \( k \) constraints of \( C_3 \) are \( \Pi_3 \)-satisfied if and only if at least \( 2k + (|C_3| - k) \) constraints of \( C_3 \) are \( \Pi_3 \)-satisfied in \( \alpha' \). So, we have reduced \textsc{Max-$\Pi_3$-CSP} to \textsc{Max-$\Pi_3$-CSP}, completing the proof. \( \square \)

Now observe that given a variable set \( V \) and a constraint multiset \( C \) over \( V \), for a random linear order \( \alpha \) of \( V \) the probability of a constraint in \( C \) being \( \Pi \)-satisfied by \( \alpha \) equals \( \frac{|C|}{|V|} \). Hence, the expected number of satisfied constraints from \( C \) is \( \frac{|C|}{|V|} \cdot |C| \), and thus there is a linear order \( \alpha \) of \( V \) satisfying at least \( \frac{|C|}{|V|} \cdot |C| \) constraints (and this bound is tight). A derandomization argument leads to \( \frac{|C|}{|V|} \)-approximation algorithms for the problems \textsc{Max-$\Pi_i$-CSP} \([52]\). No better constant factor approximation is possible assuming the Unique Games Conjecture \([52]\).

We study the parameterization of \textsc{Max-$\Pi_i$-CSP} above tight lower bound:

**\( \Pi \)-Above Average (\( \Pi \)-AA)**

- **Input:** A finite set \( V \) of variables, a multiset \( C \) of totally ordered triples \((a,b,c)\) of distinct variables from \( V \) and an integer \( k \geq 0 \).
- **Parameter:** \( k \).
- **Question:** Is there a linear order \( \alpha \) of \( V \) such that at least \( \frac{|C|}{|V|} |C| + k \) constraints of \( C \) are \Pi-satisfied by \( \alpha \)?

For example, choose \( \Pi = \{(1,2,3), (3,2,1)\} \) for Betweenness-\( \text{AA} \). \( \Pi_0 \)-\text{AA} is called the **Linear Ordering-\( \text{AA} \)** problem.

We reduce the number of problems \( \Pi \)-\text{AA} to be studied to just one. Let \( \Pi \) be a subset of \( S_3 \). Clearly, if \( \Pi \) is the empty set or equal to \( S_3 \) then the corresponding problem \( \Pi \)-\text{AA} can be solved in polynomial time.

**Theorem 5.5.** Let \( \Pi \) be a subset of \( S_3 \) such that \( \Pi \notin \{ \emptyset, S_3 \} \). There is a polynomial time transformation \( f \) from \( \Pi \)-\text{AA} to \( \Pi_0 \)-\text{AA} such that an instance \((V,C,k)\) of \( \Pi \)-\text{AA} is a “yes”-instance if and only if \((V,C_0,k) = f(V,C,k)\) is a “yes”-instance of \( \Pi_0 \)-\text{AA}.

**Proof.** From an instance \((V,C,k)\) of \( \Pi \)-\text{AA}, construct an instance \((V,C_0,k)\) of \( \Pi_0 \)-\text{AA} as follows. For each triple \((v_1,v_2,v_3)\) in \( C \), add \( |\Pi| \) triples \((v_{\pi(1)},v_{\pi(2)},v_{\pi(3)})\), one for each \( \pi \in \Pi \), to \( C_0 \).

Observe that a triple \((v_1,v_2,v_3)\) in \( C \) is \( \Pi \)-satisfied if and only if exactly one of the triples \((v_{\pi(1)},v_{\pi(2)},v_{\pi(3)})\), \( \pi \in \Pi \), is \( \Pi_0 \)-satisfied. Thus, \( \frac{|C|}{|V|} |C| + k \) constraints...
from $C$ are $\Pi$-satisfied if and only if the same number of constraints from $C_0$ are $\Pi_0$-satisfied. It remains to observe that $\frac{|\Pi|}{6}|C| + k = \frac{1}{6}|C_0| + k$ as $|C_0| = |\Pi| \cdot |C|$. $\square$

For a variable set $V$, a constraint multiset $C$ over $V$ and a linear order $\alpha$ of $V$, the $\alpha$-deviation of $(V, C)$ is the number $\text{dev}(V, C, \alpha)$ of constraints of $C$ that are $\Pi$-satisfied by $\alpha$ minus $\frac{|\Pi|}{6}|C|$. The maximum deviation of $(V, C)$ is the maximum of $\text{dev}(V, C, \alpha)$ over all linear orders $\alpha$ of $V$. Let $\text{dev}(V, C)$ denote the maximum deviation of $(V, C)$. Then problem $\Pi$-$\text{AA}$ can be reformulated as the problem of deciding whether $\text{dev}(V, C) \geq k$.

As before, we build on the probabilistic Strictly Above Expectation method. We use the following improvement of Proposition 5.1 by Alon et al. [15].

**Proposition 5.3** ([15]). Let $X$ be a real random variable and suppose that its first, second and fourth moments satisfy $\mathbb{E}[X] = 0$, $\mathbb{E}[X^2] = \sigma^2 > 0$ and $\mathbb{E}[X^4] \leq c\sigma^4$, respectively, for some constant $c$. Then $\mathbb{P}(X > \frac{\sigma^2}{\mathbb{E}^2[X^2]}) > 0$.

We combine this result with the following result from harmonic analysis, improving upon the constant in Proposition 5.2.

**Proposition 5.4** (Hypercontractive Inequality [42, 131]). Let $f = f(x_1, \ldots, x_n)$ be a polynomial of degree $r$ in $n$ variables $x_1, \ldots, x_n$ with domain $\{-1, 1\}$. Define a random variable $X$ by choosing a vector $(\varepsilon_1, \ldots, \varepsilon_n) \in \{-1, 1\}^n$ uniformly at random and setting $X = f(\varepsilon_1, \ldots, \varepsilon_n)$. Then $\mathbb{E}[X^4] \leq 9^r \mathbb{E}[X^2]^2$.

### 5.3.2 Facts on the Betweenness and Acyclic Subdigraph Problems

We look at a generalization of BETWEENNESS-$\text{AA}$ by allowing multisets of constraints. That is, instances consists of a set $V$ of variables and a multiset $B$ of betweenness constraints. For each betweenness constraint $B \in B$, let $[B]$ denote the multiset of constraints in $B$ that are equivalent to $B$. For distinct constraints $B, B' \in B$, write $B \equiv B'$ if $|B| = |B'|$. As before, call instances of BETWEENNESS-$\text{AA}$ without complete subsets reduced.

Let $(V, B)$ be an instance of BETWEENNESS-$\text{AA}$ and let $\varphi$ be a fixed function from $V$ to $\{0, 1, 2, 3\}$. A linear order $\alpha : V \to \{1, \ldots, n\}$ of $V$ is called $\varphi$-compatible if for each pair $u, v \in V$ with $\alpha(u) < \alpha(v)$ it holds $\varphi(u) \leq \varphi(v)$. For a random $\varphi$-compatible linear order $\pi$ of $V$, define a binary random variable $X_\pi$ that takes value 1 if $B \in B$ is satisfied by $\pi$ and value 0 otherwise. Let $X_B = \mathbb{E}[X_B] - 1/3$ for each $B \in B$, and let $X = \sum_{B \in B} X_B$.

Let $\varphi$ be a random function from $V$ to $\{0, 1, 2, 3\}$. Then $X$ and $X_B, B \in B$, are random variables. For a constraint $B = \{v_u, \{v_j, v_k\}\}$, the distribution of $X_B$ as given in Table 5.1 implies that $\mathbb{E}[X_B] = 0$. Thus, by linearity of expectation, $\mathbb{E}[X] = 0$.

We now generalize Lemma 5.5 for BETWEENNESS in which $B$ is a set, not a multiset. A simple modification of its proof gives us the following.

**Lemma 5.6.** Reduced instances $(V, B, k)$ of BETWEENNESS-$\text{AA}$ satisfy $\mathbb{E}[X^2] \geq \frac{11}{768}|B|$.

**Scheme.** Observe that $\mathbb{E}[X^2] = \sum_{B \in B} \mathbb{E}[X_B^2] + \sum_{B, B' \in B, B \neq B'} \mathbb{E}[X_B X_{B'}]$. Using Table 5.1, it is easy to see that $\sum_{B \in B} \mathbb{E}[X_B^2] = \frac{88}{768}|B|$. 


Similarly to Section 5.2, we obtain that

\[ \sum_{\pi} \phi(u) = \phi(v) \]

in combination with \( \sum_{\pi} \phi(u) < \phi(v) \quad \phi(u) > \phi(v) \). Thus, there is a linear order \( \pi \) of \( A \) such that \( \phi(u) = \phi(v) \) is satisfied by \( \phi(u) < \phi(v) \), and \( \phi(u) > \phi(v) \) is satisfied by \( \phi(u) < \phi(v) \). We therefore define, for a directed multigraph \( D \) and an arc \( a \),

\[
\begin{array}{|c|c|}
\hline
\text{Event } E & \text{Value of } Y_a & \mathbb{P}(E) \\
\hline
\phi(u) = \phi(v) & 0 & 1/4 \\
\phi(u) < \phi(v) & 1/2 & 3/8 \\
\phi(u) > \phi(v) & -1/2 & 3/8 \\
\hline
\end{array}
\]

Table 5.4: Probability distribution of the random variable \( Y_a \) corresponding to the arc \( a = (u, v) \).

Let \( U = \{(B, B') \mid B, B' \in B, B \neq B' \} \) be the set of all ordered pairs of distinct constraints in \( B \). Let \( U^s = \{(B, B') \in U \mid \text{vars}(B) = \text{vars}(B'), B \neq B' \} \) and \( U^{**} = \{(B, B') \in U \mid B \equiv B' \} \). Taking into consideration that \( |U^s| \leq |B| \) and \( |U^{**}| \geq 0 \), similarly to Section 5.2, we obtain that

\[
\sum_{(B, B') \in U} \mathbb{E}[X_B X_{B'}] \geq -\frac{66}{768} |B| - \frac{11}{768} |U^s| + \frac{22}{768} |U^{**}| \geq -\frac{66}{768} |B| - \frac{11}{768} |B| = -\frac{77}{768} |B|. 
\]

In combination with \( \sum_{B \in B} \mathbb{E}[X^2_B] = \frac{88}{768} |B| \) it follows that \( \mathbb{E}[X^2_B] \geq \frac{11}{768} |B| \).

The problem \((k)\)-acyclic subdigraph can be reformulated as asking for a linear order \( \pi \) of \( V(A) \) in a given directed multigraph \( D \) that maximizes the number of satisfied arcs, where an arc \((u, v) \in A(D)\) is satisfied by \( \pi \) if \( \pi(u) < \pi(v) \). If \( \pi \) is a uniformly-at-random linear order of \( V(D) \) then the probability of an arc of \( D \) to be satisfied is 1/2. Thus, there is a linear order \( \pi \) of \( V(D) \) in which the number of satisfied arcs is at least \(|A|/2\). We therefore define, for a directed multigraph \( D \) and a linear order \( \pi \) of \( V(D) \), the \( \pi \)-deviation of \( D \) as the number of arcs satisfied by \( \pi \) minus \(|A(D)|/2\), and denote it by \( \text{dev}(V(D), A(D), \pi) \).

As every linear order of \( V(D) \) satisfies exactly one of two mutually opposite arcs \((u, v)\) and \((v, u)\), the following lemma naturally codifies a reduction rule.

**Lemma 5.7.** Let \( D \) be a directed multigraph and let \( \pi \) be a linear order of \( V(D) \). Let \( A'(D) \) be the set of arcs obtained from \( A(D) \) by deleting all pairs of mutually opposite arcs. Then \( \text{dev}(V(D), A(D), \pi) = \text{dev}(V(D), A'(D), \pi) \).

A directed multigraph without mutually opposite arcs is called reduced.

Let \( D \) be a directed multigraph and let \( \phi \) be a fixed function from \( V(D) \) to \( \{0, 1, 2, 3\} \). For a random \( \phi \)-compatible linear order \( \pi \) of \( V(D) \), define a binary random variable \( y_a \) that takes value 1 if \( a \) is satisfied by \( \pi \) and value 0 otherwise. Let \( Y_a = \mathbb{E}[y_a] - 1/2 \) for each \( a \in A(D) \) and let \( Y = \sum_{a \in A} Y_a \).

Now let \( \phi \) be a random function from \( V(D) \) to \( \{0, 1, 2, 3\} \). Then \( Y \) and \( Y_a \) are random variables. For an arc \( a = (u, v) \), the distribution of \( Y_a \) as given in Table 5.4 implies that \( \mathbb{E}[Y_a] = 0 \). Thus, by linearity of expectation, \( \mathbb{E}[Y] = 0 \).

We have the following analogue of Lemma 5.6.

**Lemma 5.8.** For a reduced directed multigraph \( D \) it holds that \( \mathbb{E}[Y^2] \geq \frac{1}{576} |A(D)| \).
5.3. TERNARY PERMUTATION-CSPS

Proof. We write $\mathbb{E}[Y^2]$ as the sum

$$
\mathbb{E}[Y^2] = \sum_{a \in A} \mathbb{E}[Y_a^2] + \sum_{a,a' \in A(D), a \neq a'} \mathbb{E}[Y_a Y_{a'}].
$$

From Table 5.4 it follows that $\mathbb{E}[Y_a^2] = \frac{3}{16}$, and hence it remains to bound the second sum in (5.4). Consider any ordered pair $(a, a')$ of distinct arcs in $D$. If $a$ and $a'$ are vertex-disjoint, then clearly $\mathbb{E}[Y_a Y_{a'}] = 0$. If $a$ and $a'$ have vertices in common, we define

$$
S_1(u) = \{(a, a') \mid a = (u, x), a' = (u, y), x, y \in V \}
$$

$$
\cup \{(a, a') \mid a = (x, u), a' = (y, u), x, y \in V \},
$$

$$
S_2(u) = \{(a, a') \mid a = (u, x), a' = (y, u), x, y \in V \}
$$

$$
\cup \{(a, a') \mid a = (x, u), a' = (u, y), x, y \in V \},
$$

$$
S_3(u, v) = \{(a, a') \mid a = (u, v), a' = \}.
$$

By setting $l(u) = |\{a \in A(D) \mid a = (u, y), y \in V(D)\}|$ and $r(u) = |\{a \in A(D) \mid a = (x, u), x \in V(D)\}|$ it follows that

$$
|S_1(u)| = l(u)(l(u) - 1) + r(u)(r(u) - 1),
$$

$$
|S_2(u)| = 2l(u)r(u).
$$

Consider a pair $(a, a') \in S_1(u)$, with say $a = (u, x)$ and $a' = (u, y)$. It is easy to calculate that out of the 64 functions $\varphi : \{u, x, y\} \rightarrow \{0, 1, 2, 3\}$, there are 14 in which $\varphi(u) < \varphi(x)$ and $\varphi(u) < \varphi(y)$. Symmetrically, there are 14 functions $\varphi$ in which $\varphi(u) > \varphi(x)$ and $\varphi(u) > \varphi(y)$. In both cases, $Y_a Y_{a'} = \frac{1}{2}$, by Table 5.1. Similarly, there are 4 functions $\varphi$ in which $\varphi(u) < \varphi(x)$ and $\varphi(u) > \varphi(y)$, and 14 function $\varphi$ in which $\varphi(u) > \varphi(x)$ and $\varphi(u) < \varphi(y)$; in both cases $Y_a Y_{a'} = \frac{1}{2}$. For all other functions $\varphi$ we have that $Y_a Y_{a'} = 0$, and thus it follows that $\mathbb{E}[Y_a Y_{a'}] = \frac{5}{64}$ for each pair of arcs $(a, a')$ in $S_1(u)$.

Similarly, for each pair $(a, a') \in S_2(u)$ it holds that $\mathbb{E}[Y_a Y_{a'}] = -\frac{5}{64}$, and for each pair $(a, a') \in S_3(u, v)$ it holds that $\mathbb{E}[Y_a Y_{a'}] = \mathbb{E}[Y_a^2] = \frac{3}{16}$.

Since $D$ is reduced, any ordered pair $(a, a')$ of distinct arcs in $D$ sharing a vertex $u$ belongs to exactly one of $S_1(u), S_2(u), S_3(u)$. Hence,

$$
\sum_{a,a' \in A, a \neq a'} \mathbb{E}[Y_a Y_{a'}] = \sum_{u \in V(D)} \frac{5}{64}|S_1(u)| - \frac{5}{64}|S_2(u)| + \sum_{u, v \in V(D)} w'|S_3(u, v)|,
$$
with \( \frac{5}{64} + \frac{5}{64} + w' = \frac{3}{16} \), because \( S_3(u, v) = S_1(u) \cap S_1(v) \). Thus, \( w' = \frac{1}{32} \), and

\[
\sum_{a,a' \in A, a \neq a'} \mathbb{E}[Y_a Y_{a'}] = \frac{5}{64} \sum_{u \in V(D)} l(u)(l(u) - 1) + r(u)\max(0, r(u) - 1) - 2l(u)r(u) + \sum_{u,v \in V(D)} \frac{1}{32} |S_3(u, v)|
\]

\[
= \frac{5}{64} \sum_{u \in V(D)} (l(u) - r(u))^2 - l(u) - r(u) + \sum_{u,v \in V} \frac{1}{32} |S_3(u, v)| \geq - \frac{5}{64} \sum_{u \in V(D)} l(u) + r(u) = \frac{10}{64} |A|
\]

because each arc contributes exactly one to \( \sum_{u \in V(D)} l(u) \) and one to \( \sum_{u \in V(D)} r(u) \).

We conclude that \( \mathbb{E}[Y^2] \geq \frac{9}{16} |A(D)| - \frac{5}{32} |A(D)| = \frac{1}{16} |A(D)| \). \( \square \)

The following theorem was proved by Gutin et al. [143].

**Theorem 5.6.** **Acyclic Subdigraph-AA** has a kernel with a quadratic number of vertices and arcs.

### 5.3.3 Kernels for \( \Pi \)-AA Problems

Our goal is now to construct a polynomial kernel for Linear Ordering-AA. We start from the following key construction. With an instance \((V, C)\) of Linear Ordering, we associate an instance \((V, B)\) of Betweenness and two instances \((V, A')\) and \((V, A'')\) of Acyclic Subdigraph as follows: If \( C = (u, v, w) \subseteq C \) then \( B_C = (v, \{u, w\}) \subseteq B \) and \( a'_C = (u, v) \subseteq A' \) and \( a''_C = (v, w) \subseteq A'' \).

**Lemma 5.9.** Let \((V, C, k)\) be an instance of Linear Ordering-AA and let \( \alpha \) be a linear order of \( V \). Then

\[
dev(V, C, \alpha) = \frac{1}{2} \left[ dev(V, A', \alpha) + dev(V, A'', \alpha) + dev(V, B, \alpha) \right].
\]

**Proof.** For each constraint \( C = (u, v, w) \subseteq C \), define a binary variable \( y'_C \) that takes value 1 if \( a'_C \) is satisfied by \( \alpha \) and value 0 otherwise. Similarly, define binary variables \( y''_C \) for arc \( a''_C \), \( z_C \) for constraint \( B_C \) and \( z_C \) for constraint \( C \). To show the lemma it suffices to prove that for each constraint \( C \subseteq C \) and every linear order \( \pi \) of \( \{u, v, w\} \) it holds that

\[
dev(V, \{C\}, \pi) = \frac{1}{2} \left[ dev(V, \{a'_C\}, \pi) + dev(V, \{a''_C\}, \pi) + dev(V, \{B_C\}, \pi) \right],
\]

where \( dev(V, \{C\}, \pi) = z_C - 1/6 \), \( dev(V, \{a'_C\}, \pi) = y'_C - 1/2 \), \( dev(V, \{a''_C\}, \pi) = y''_C - 1/2 \) and \( dev(V, \{B_C\}, \pi) = z_C - 1/3 \). Thus, it suffices to prove that \( z_C = (y'_C + y''_C + z_C - 1) / 2 \). But this expression holds, as can be seen from Table 5.5: if \( C \) is satisfied by \( \pi \) then all three constraints \( a'_C, a''_C, B_C \) are satisfied by \( \pi \), whereas if \( C \) is not satisfied by \( \pi \) then exactly one of the three constraints \( a'_C, a''_C, B_C \) is satisfied by \( \pi \). \( \square \)
Let \((V, C, k)\) be an instance of Linear Ordering-AA, and let \(\varphi\) be a function from \(V\) to \(\{0, 1, 2, 3\}\). For a random \(\varphi\)-compatible linear order \(\pi\) of \(V\), define a binary random variable \(z_C\) that takes value 1 if \(C\) is satisfied by \(\pi\) and value 0 otherwise. Let \(Z_C = E[z_C] - 1/6\) for each \(C \in C\), and let \(Z = \sum_{C \in C} Z_C\).

**Lemma 5.10.** If \(Z \geq k\) then \((V, C, k)\) is a “yes”-instance of Linear Ordering-AA.

**Proof.** By linearity of expectation, \(Z \geq k\) implies \(E[\sum_{C \in C} z_C] \geq |C|/6 + k\). Thus, if \(Z \geq k\) then there is a \(\varphi\)-compatible linear order \(\pi\) that satisfies at least \(|C|/6 + k\) constraints. \(\square\)

Fix a function \(\varphi: V \rightarrow \{0, 1, 2, 3\}\) and assign variables \(X_C, Y'_C, Y''_C\), respectively, to the three instances of Betweenness and Acyclic Subdigraph above.

**Lemma 5.11.** For each \(C \in C\) it holds \(Z_C = \frac{1}{2}[Y'_C + Y''_C + X_C]\).

**Proof.** Let \(C = (u, v, w) \in C\). Table 5.6 shows the values of \(X_C, Y'_C, Y''_C, Z_C\) for some relations between \(\varphi(u), \varphi(v), \varphi(w)\). The values of \(Y'_C, Y''_C, X_C\) can be computed using Tables 5.4 and 5.1, respectively. In all cases of Table 5.6 it holds \(Z_C = \frac{1}{2}[Y'_C + Y''_C + X_C]\). Thus, \(Z_C = \frac{1}{2}|Y'_C + Y''_C + X_C|\) for each possible relation between \(\varphi(u), \varphi(v), \varphi(w)\). \(\square\)

Let \(Y = \sum_{C \in C} Y'_C + Y''_C\), let \(X = \sum_{C \in C} X_C\) and let \(\varphi\) be a random function from \(V\) to \(\{0, 1, 2, 3\}\). Then \(X, Y, Z\) and \(Y'_C, Y''_C, X_C, Z_C\), for each \(C \in C\), are random variables. From \(E[Y'] = E[Y''] = E[X] = 0\) it follows that \(E[Z] = 0\). We will be able to use Proposition 5.4 in the proof of Lemma 5.14 due to the following result.

**Lemma 5.12.** The random variable \(Z\) can be expressed as a polynomial of degree 6 in independent uniformly distributed random variables with values \(-1\) and 1.

**Proof.** Consider \(C = (u, v, w) \in C\). Define variables

\[
\varepsilon_1^u = \begin{cases} 
-1, & \text{if } \varphi(u) \in \{0, 1\}, \\
+1, & \text{if } \varphi(u) \in \{2, 3\},
\end{cases}
\quad\varepsilon_2^u = \begin{cases} 
-1, & \text{if } \varphi(u) \in \{0, 2\}, \\
+1, & \text{if } \varphi(u) \in \{1, 3\}.
\end{cases}
\]

<table>
<thead>
<tr>
<th>linear order (\pi) of ({u, v, w})</th>
<th>constraints satisfied by (\pi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(uvw)</td>
<td>((u, v), (v, w), (v, {u, w}))</td>
</tr>
<tr>
<td>(uwv)</td>
<td>((u, w))</td>
</tr>
<tr>
<td>(wuv)</td>
<td>((w, u))</td>
</tr>
<tr>
<td>(vuw)</td>
<td>((v, w))</td>
</tr>
<tr>
<td>(vuw)</td>
<td>((v, w))</td>
</tr>
<tr>
<td>(wvu)</td>
<td>((v, {u, w}))</td>
</tr>
</tbody>
</table>

Table 5.5: Constraints satisfied by \(\pi\).
Lemma 5.13. Let $\epsilon$ be the number of complete triples in the directed multigraph of $B$.

The above polynomial is of degree 6. It remains to recall that $\phi$ cyclic- $\epsilon$-AA and $A$ cyclic- $\epsilon$-AA, where $-\epsilon$ plays the role of 0.

Then we can write $Z_C$ as the polynomial

$$Z_C = \sum_{q=0}^{63} (-1)^{s_q} W_q \cdot (e_q^1 + c_1^q)(e_q^2 + c_2^q)(e_q^3 + c_3^q)(e_q^4 + c_4^q)(e_q^5 + c_5^q)(e_q^6 + c_6^q),$$

where $c_1^q, c_2^q, c_1, c_2$ is the binary representation of $q$, $s_q$ is the number of digits equal $-1$ in this representation, and $W_q$ equals the value of $Z_C$ for the case when the binary representations of $\phi(u), \phi(v)$ and $\phi(w)$ are $c_1^q, c_2^q, c_1, c_2$ and $c_1^q, c_2^q$, respectively. The actual values for $Z_\phi$ for each case are given in the proof of Lemma 5.11. The above polynomial is of degree 6. It remains to recall that $Z = \sum_{C \in \mathcal{C}} Z_C$.

Consider the following natural transformation of our key construction introduced in the beginning of this subsection. Let $(V, C, k)$ be an instance of LINEAR ORDERING-AA and $(V, B, k)$, $(V, A', k)$ and $(V, A'', k)$ be the associated instances of BETWEENNESS-AA and ACYCLIC SUBDIAGRAM-AA. Let $b$ be the number of pairs of mutually opposite arcs in the directed multigraph $D = (V, A' \cup A'')$ and let $r = 2(|C| - b)$. Let $t$ be the number of complete triples in $B$ and let $s = |C| - 3t$.

Lemma 5.13. $\mathbb{E}[Z^2] \geq \frac{11}{3072} (r + s)$.

Proof. Let $A = A' \cup A'' = \{a_1, \ldots, a_2|C|\}$ and $D = (V, A)$. Fix a function $\phi : V \to \{0, 1, 2, 3\}$. For a random $\phi$-compatible linear order $\pi$ of $V$, define a binary random
variable \( y_a \) that takes value one if and only if \( a \) is satisfied by \( \pi \). Analogously, define a binary random variable \( y_B \) that takes value one if and only if \( B \) is satisfied by \( \pi \). Let \( Y_a = \mathbb{E}[y_a] - 1/2 \) for all \( a \in A \), let \( X_B = \mathbb{E}[x_B] - 1/3 \) for all \( B \in B \) and let \( Y = \sum_{a \in A} Y_a \) and \( X = \sum_{B \in B} X_B \). Recall that \( b \) was the number of deleted pairs of mutually opposite arcs from \( D \), and \( t \) was the number of complete 3-sets deleted from \( B \). Let \( A^\ast \) be the set of remaining arcs and let \( B^\ast \) be the set of remaining betweenness constraints. Then \( Y = \sum_{a \in A^\ast} Y_a = \sum_{a \in A^\ast} Y_a^c \), \( X = \sum_{B \in B^\ast} X_B = \sum_{B \in B^\ast} X_B^c \), and, by Lemma 5.11, \( Z = Y + X/2 \). Therefore,

\[
\mathbb{E}[Z^2] = \mathbb{E}[Y^2 + XY + X^2/4]
\]

\[
= \mathbb{E}[Y^2] + \mathbb{E}[X^2]/4 + \mathbb{E} \left[ \left( \sum_{a \in A^\ast} Y_a \right) \left( \sum_{B \in B^\ast} X_B \right) \right]
\]

\[
= \mathbb{E}[Y^2] + \mathbb{E}[X^2]/4 + \sum_{a \in A^\ast} \sum_{B \in B^\ast} \mathbb{E}[Y_a X_B].
\]

We want to show that \( \mathbb{E}[Y_a X_B] = 0 \) for any pair \( (a, B) \). Let \( \varphi^\prime : V \to \{0, 1, 2, 3\} \) be defined as \( \varphi^\prime(x) = 3 - \varphi(x) \) for all \( x \). Let \( Y_a(\varphi) \) be the value of \( Y_a \) when considering \( \varphi \)-compatible orders and define \( Y_a(\varphi^\prime), X_B(\varphi) \) and \( X_B(\varphi^\prime) \) analogously. From Table 5.1 we note that \( X_B(\varphi) = X_a(\varphi^\prime) \), and from Table 5.4 we note that \( Y_B(\varphi) = -Y_a(\varphi^\prime) \). From \( \mathbb{E}[Y_a X_B] = \frac{1}{4|V|} \sum_{\varphi} Y_a(\varphi) X_B(\varphi) \) it follows that

\[
2\mathbb{E}[Y_a X_B] = 2 \left[ \frac{1}{4|V|} \sum_{\varphi} Y_a(\varphi) X_B(\varphi) \right]
\]

\[
= \frac{1}{4|V|} \sum_{\varphi} \left[ Y_a(\varphi) X_B(\varphi) + Y_a(\varphi^\prime) X_B(\varphi^\prime) \right]
\]

\[
= \frac{1}{4|V|} \sum_{\varphi} X_B(\varphi) [Y_a(\varphi) + Y_a(\varphi^\prime)] = 0.
\]

Therefore, \( \mathbb{E}[Z^2] = \mathbb{E}[Y^2] + \mathbb{E}[X^2]/4 \). It follows from Lemmas 5.8 and 5.6 that \( \mathbb{E}[Y^2] \geq |A^\ast|/32 \) and \( \mathbb{E}[X^2] \geq \frac{11}{3072} |B^\ast| \). We conclude that \( \mathbb{E}[Z^2] \geq \frac{11}{3072} (|A^\ast| + |B^\ast|) \). \( \square \)

**Lemma 5.14.** There is a constant \( c > 0 \) such that if \( |A^\ast| + |B^\ast| \geq ck^2 \), then \((V, C, k)\) is a “yes”-instance of Linear Ordering-AA.

**Proof.** By Lemma 5.12 and Proposition 5.4, we have \( \mathbb{E}[Z^4] \leq 9^6 \mathbb{E}[Z^2]^2 \). As \( \mathbb{E}[Z] = 0 \), it follows from Proposition 5.3 that \( \mathbb{P}(Z > \frac{\sqrt{\mathbb{E}[Z^2]}}{2.97}) > 0 \). By Lemma 5.13, \( \mathbb{E}[Z^2] \geq \frac{11}{3072} (|A^\ast| + |B^\ast|) \). Hence, \( \mathbb{P} \left( Z > \frac{1}{2.97} \sqrt{\frac{11}{3072}} (|A^\ast| + |B^\ast|) \right) > 0 \). Therefore if \( |A^\ast| + |B^\ast| > ck^2 \), where \( c = 4 \cdot 9^6 \cdot 3072/11 \), then by Lemma 5.10 \((V, C, k)\) is a “yes”-instance of Linear Ordering-AA. \( \square \)

After we have deleted mutually opposite arcs from \( D \) and complete triples from \( B \) we may assume, by Lemma 5.14, that \( D \) has an arc multiset \( A^\ast \) left, with \( |A^\ast| = O(k^2) \), and \( B \) is reduced to a set \( B^\ast \), with \( |B^\ast| = O(k^3) \). By Lemma 5.9,
\( \text{dev}(V, C) = \max_{\pi} \left( \left[ \text{dev}(V, A, \pi) + \text{dev}(V, B, \pi) \right] / 2 \right) \), where the maximum is taken over all linear orders \( \pi \) of \( V \).

We now create a new instance \((V', C', k)\) of Linear Ordering-AA as follows. Let \( \mu \) be a new variable not in \( V \). For every arc \( a = (u, v) \in A^* \) add the constraints \((\mu, u, v)\), \((u, \mu, v)\) and \((u, v, \mu)\) to \( C' \). For every constraint \( B = (a, [b, c]) \in B^* \) add the constraints \((b, a, c)\) and \((c, a, b)\) to \( C' \). Let \( V' \) be the set of variables that appear in some constraint in \( C' \). Then \((V', C', k)\) is an instance of Linear Ordering with \( O(k^2) \) variables. Now the number of constraints in \( C' \) satisfied by any linear order \( \alpha \) of \( V' \) equals the number of constraints in \( D \) satisfied by \( \alpha \) plus the number of constraints in \( B \) satisfied by \( \alpha \). As the average number of constraints satisfied in \((V', C')\) equals \((3|A^*| + 2|B^*|)/6 = |A^*|/2 + |B^*|/3\), it follows that \( \text{dev}(V, C) = \max_{\pi} \left( \left[ \text{dev}(V, A, \pi) + \text{dev}(V, B, \pi) \right] / 2 \right) = \text{dev}(V', C') / 2 \). Hence, \((V', C', k)\) is a kernel of Linear Ordering-AA with \( O(k^2) \) variables and constraints. We have established the following theorem.

**Theorem 5.7.** Linear Ordering-AA has a quadratic kernel.

This theorem in combination with Theorem 5.5 enables us to prove the following.

**Theorem 5.8.** There is a bikernel with \( O(k^2) \) variables from \( \Pi_I\text{-}AA \) to \( \Pi_I\text{-}AA \) for each pair \((i, j)\) such that \( 0 \leq i \leq 10 \) and \( 0 \leq j \leq 10 \) but \( j \not\in \{2, 7\} \).

Proof. By Theorem 5.5, it suffices to prove the statement for \( i = 0 \) and \( 0 \leq j \leq 10 \) but \( j \not\in \{2, 7\} \). As the case \( j = 0 \) follows from Theorem 5.7, it remains to analyze the following cases.

**Part 1:** \( j = 5 \). By the proof of Theorem 5.7, any instance \((V, C, k)\) of Linear Ordering-AA can be reduced in polynomial time to a mixed instance consisting of an instance \( D = (V, A^*) \) of Acyclic Subdigraph-AA, with \(|A^*| = O(k^2)\), and an instance \((V, B^*)\) of Betweenness-AA, with \(|B^*| = O(k^2)\), such that the answer to \((V, C, k)\) is “yes” if and only if there is a linear order of \( V \) satisfying at least \(|A^*|/2 + |B^*|/3 + k\) arcs and constraints of the mixed instance. Let \( V' \) be the set of all variables and vertices in constraints of \( B^* \) and arcs of \( A^* \); then \(|V'| = O(k^2)\).

Construct an instance \((V', B')\) of Betweenness as follows. Set \( V' = V^* \cup \{y, z\} \) and initialize \( B' \) by setting \( B' = B^* \). Then for each \( x \in V' \), add \(|A^*| + |B^*| + 1\) copies of the constraint \((x, \{y, z\})\) to \( B' \), and for each arc \((u, v) \in A^* \), add one copy of the constraint \((u, \{v, z\})\). This completes the construction of \((V', B')\). Observe that \(|V'| = O(k^2)\), and the total number of constraints in the multiset \( B' \) is equal to \( p = (|V'| + 1)(|A^*| + |B^*| + 1) - 1 \). Recall that the average number of constraints satisfied in an instance of Betweenness with \( p \) constraints equals \( p/3 \). We may assume that \( p \) is divisible by 3, as otherwise we can add one or two more constraints of the type \((x, \{y, z\})\) to \( B' \). Let \( d = (|A^*| + |B^*|) - (|A^*|/2 + |B^*|/3 + k) \) and let \( k' = 2p/3 - d \). Then \((V', B', k')\) is a “yes”-instance for Betweenness-AA if and only if there is a linear order of \( V' \) that falsifies at most \( d \) constraints of \( B' \). Since \( d \leq |A^*| + |B^*| \), to falsify at most \( d \) constraints of \( B' \), a linear order \( \alpha \) of \( V' \) must satisfy all constraints of the form \((x, \{y, z\})\) and at least \(|A^*|/2 + |B^*|/3 + k \) other constraints. Since \( \alpha \) must satisfy all constraints of the form \((x, \{y, z\})\), we have \( \{\alpha^{-1}(1), \alpha^{-1}(|V'|)\} = \{y, z\} \). Without loss of generality, we may assume that
5.3. TERNARY PERMUTATION-CSPS

Recall that $\alpha^{-1}(|V'|) = z$. Then $\alpha$ satisfies at least $|A^*|/2 + |B^*|/3 + k$ other constraints if and only if it satisfies at least $|A^*|/2 + |B^*|/3 + k$ arcs and constraints of the mixed instance. Thus, $(V', B', k')$ is equivalent to $(V, C, k)$, and since $k'$ is bounded by a function of $k$, we are done.

**Part 2:** $j = 1$. Denote constraints of $\Pi_1$-AA by $(u < \min\{v, w\})$. Such a constraint is satisfied by a linear order $\alpha$ of $\{u, v, w\}$ if and only if $\alpha(u) < \min\{\alpha(v), \alpha(w)\}$. Consider the instance $(V', B', k')$ built in Part 1. Construct an instance $(V''', C_1, k_1)$ of $\Pi_1$-AA as follows. Let $V'' = V' \cup \{z'\}$, where $z'' \notin V'$. For each constraint $(v, \{u, w\})$ of $B'$, let $C_1$ have two copies of $(u < \min\{v, w\})$, two copies of $(w < \min\{u, v\})$ and one copy of $(v < \min\{w, z'\})$ and one copy of $(v < \min\{u, z'\})$. Thus, $C_1$ has $6p$ constraints and note that the average number of constraints satisfied in an instance of $\Pi_1$-AA with $6p$ constraints is $2p$. Let $k_1 = p - d$, where $p$ and $d$ are defined in Part 1.

Let $\alpha$ be a linear order of $V''$ and assume that $\alpha$ satisfies the maximum number of constraints in $C_1$ and this number is at least $2p + k_1 = 3p - d$. We may assume that $\alpha(z') = |V''|$ as moving $z'$ to the last position in the linear order will not falsify any constraint of $C_1$. Observe now that if $\alpha$ satisfies $(v, \{u, w\})$, then it satisfies exactly $3$ constraints of $C_1$ from the six constraints generated by $(v, \{u, w\})$ and if $\alpha$ falsifies $(v, \{u, w\})$, it satisfies exactly two constraints of $C_1$ from the six constraints generated by $(v, \{u, w\})$. Therefore, $\alpha$ satisfies exactly $3t + 2(p - t)$ constraints of $C_1$, where $t$ is the number of constraints in $B'$ satisfied by $\alpha$. Hence, $t \geq p - d$.

Now assume that a linear order $\alpha$ of $V'$ satisfies at least $p - d$ constraints of $B'$. We extend $\alpha$ to $V''$ by setting $\alpha(z') = |V''|$. Similarly to the above we can show that $\alpha$ satisfies at least $2p + k_1 = 3p - d$ constraints in $C_1$. Thus, $(V', C_1, k_1)$ is equivalent to $(V', B', k')$ and, therefore by Part 1, to $(V, C, k)$, an instance of Linear Ordering-AA. Clearly, $|V''| = O(k^2)$ and $k_1$ is bounded by a function of $k$.

**Part 3:** $j = 3$. In Part 2, we have proved that for any instance $(V, C, k)$ of Linear Ordering-AA there is an equivalent instance $(V', C_1, k_1)$ of $\Pi_1$-AA with $O(k^2)$ variables and distinct constraints (and $k_1$ is bounded by a function of $k$). Recall that $(V', C_1, k_1)$ has $6p$ constraints. Let $\alpha$ be a linear order of $V'$ and let $\alpha'$ be the reverse linear order. As in the proof of Case $i = 3$ of Theorem 5.4, construct from $(V', C_1, k_1)$ an instance $(V'', C_3, k_3)$ of $\Pi_3$-AA such that $C_3$ has $12p$ constraints and at least $q$ constraints of $C_1$ are satisfied by $\alpha$ if and only if at least $2q + (|C_1| - q)$ constraints of $C_3$ are satisfied in $\alpha'$. Let $q = 2p + k_1$ and $k_3 = k_1$. Assume that $(V', C_1, k_1)$ is a "yes"-instance certified by $\alpha$. Then $\alpha'$ satisfies at least $8p + k_3$ constraints of $(V', C_3, k_3)$ and $(V', C_3, k_3)$ is a "yes"-instance. Similarly, if $(V', C_3, k_3)$ is a "yes"-instance, then $(V', C_1, k_1)$ is a "yes"-instance, too.

**Part 4:** $j = 4, 8, 9, 10$. For each $j = 4, 8, 9, 10$ the proof is similar to Part 2 and, thus, we will only describe how to transform the instance $(V', B', k')$ built in Part 1 into an instance $(V', C_1, k')$ of $\Pi_1$-AA for every $i = 4, 8, 9, 10$, and observe how the fact that a constraint $B$ of $(V', B', k')$ is satisfied or falsified corresponds to the number of satisfied constraints in the instance of $\Pi_1$-AA generated by $B$. Then it is not hard to check that $(V', B', k')$ and $(V', C_1, k')$ are equivalent.
5. Parameterizations above Guarantee

Case $j = 4$. Denote constraints of $\Pi_4$-AA by $(u \parallel \{v \prec w\})$. Such a constraint is $\Pi_4$-satisfied by a linear order $\alpha$ of $\{u, v, w\}$ if and only if $a(v) < a(w)$ and $a(u)$ is not between $a(v)$ and $a(w)$. Construct an instance $(V', C_4, k_4)$ of $\Pi_4$-AA as follows. For each constraint $(v, \{u, w\})$ of $B'$, let $C_4$ have four constraints: $(u \parallel \{v \prec w\}),(u \parallel \{w \prec v\}),(w \parallel \{u \prec v\})$ and $(w \parallel \{v \prec u\})$. It is easy to check that if $(v, \{u, w\})$ is satisfied by a linear order $\alpha$ of $V'$, then two of the four constraints are satisfied by $\alpha$ and if $(v, \{u, w\})$ is falsified by $\alpha$, then only one of the four constraints is satisfied by $\alpha$.

Case $j = 8$. Denote constraints of $\Pi_8$-AA by $(v \parallel \{u, w\})$. Such a constraint is satisfied by a linear order $\alpha$ of $\{u, v, w\}$ if and only if either $a(v) < a(u) < a(w)$ or $a(w) < a(v)$. For each constraint $(v, \{u, w\})$ of $B'$, let $C_8$ have two constraints: $(w \prec v < u \prec w)$ and $(u \prec v < w \prec u)$. It is easy to check that if $(v, \{u, w\})$ is satisfied by a linear order $\alpha$ of $V'$, then both constraints generated by $(v, \{u, w\})$ are satisfied by $\alpha$ and if $(v, \{u, w\})$ is falsified by $\alpha$, then only one of two constraints is satisfied by $\alpha$.

Case $j = 9$. Denote constraints of $\Pi_9$-AA by $(v \parallel \{u, w\})$. Such a constraint is satisfied by a linear order $\alpha$ of $\{u, v, w\}$ if and only if $a(v)$ is not between $a(u)$ and $a(w)$. Construct an instance $(V', C_9, k_9)$ of $\Pi_9$-AA as follows. For each constraint $(v, \{u, w\})$ of $B'$, let $C_9$ have two constraints: $(u \parallel \{v \prec w\})$ and $(w \parallel \{u \prec v\})$. It is easy to check that if $(v, \{u, w\})$ is satisfied by a linear order $\alpha$ of $V'$, then both constraints generated by $(v, \{u, w\})$ are satisfied by $\alpha$ and if $(v, \{u, w\})$ is falsified by $\alpha$, then only one of two constraints is satisfied by $\alpha$.

Case $j = 10$. Denote constraints of $\Pi_{10}$-AA by $(u \parallel \{v \prec w\})$. Such a constraint is satisfied by a linear order $\alpha$ of $\{u, v, w\}$ if and only if we do not have $a(u) < a(v) < a(w)$. For each constraint $(v, \{u, w\})$ of $B'$, let $C_{10}$ have four constraints: $(v \prec u < w)$, $(v \prec w < u)$, $(u \prec w < v)$ and $(w \prec u < v)$. It is easy to check that if $(v, \{u, w\})$ is satisfied by a linear order $\alpha$ of $V'$, then all four constraints generated by $(v, \{u, w\})$ are satisfied by $\alpha$ and if $(v, \{u, w\})$ is falsified by $\alpha$, then only three of the four constraints are satisfied by $\alpha$.

Part 5: $j = 6$. Denote constraints of $\Pi_6$-AA by $(u \parallel \{v \prec w\} \parallel \{w \prec v\})$. Such a constraint is satisfied by a linear order $\alpha$ of $\{u, v, w\}$ if and only if either $a(u) < a(v) < a(w)$ or $a(w) < a(v)$. Consider the instance $(V', B', k')$ built in Part 1. Construct an instance $(V_6, C_6, k_6)$ of $\Pi_6$-AA as follows.

Let $V_6 = V' \cup \{a, b\}$, where $\{a, b\} \cap V' = \emptyset$. Initiate $C_6$ by adding to it, for each $x \in V'$, $6p + 1$ copies of $(x < b < a \lor a, \{x, b\})$ and $6p + 1$ copies of $(x < a < b \lor b, \{x, a\})$. For each $(v, \{u, w\}) \in B'$, add to $C_6$ the following constraints: two copies of $(u \prec w < v \lor v, \{u, w\})$, two copies of $(w \prec u < v \lor v, \{u, w\})$, a copy of $(b \prec u < w \lor u, \{v, b\})$, and a copy of $(b \prec v < w \lor w, \{v, b\})$. Recall that $B'$ has $p$ constraints and note that $C_6$ has $6p + 2(6p + 1)|V'|$ constraints. Observe that the average number of satisfied constraints, in an instance of $\Pi_6$-AA with $6p + 2(6p + 1)|V'|$ constraints, is $3p + (6p + 1)|V'|$. Let $k_6 = (6p + 1)|V'| + (5p - 3d)$, where $d$ is defined in Part 1.

Then $(V_6, C_6, k_6)$ is a "yes"-instance if and only if there is a linear order $\alpha$ of $V_6$
that satisfies at least $2(6p + 1)|V'| + (5p - 3d)$ constraints. For $a$ to satisfy so many constraints, it must satisfy all constraints of the forms $(x < b < a)$ or $a, (x, b)$ and $(x < a < b)$ or $b, (x, a)$, implying that $a$ and $b$ must be the last two variables in $a$, and at least $5p - 3d$ constraints generated by $B'$. Observe that if $a$ satisfies $(v, \{u, w\}) \in B'$ then exactly five constraints of $C_6$ generated by $(v, \{u, w\})$ are satisfied by $a$ and if $a$ falsifies $(v, \{u, w\}) \in B'$ then exactly two constraints of $C_6$ generated by $(v, \{u, w\})$ are satisfied by $a$. Thus, $a$ satisfies at least $5p - 3d$ constraints generated by $B'$ if and only if $a$ satisfies at least $p - d$ constraints of $B'$. Therefore, $(V', B', k')$ and $(V, C_6, k_6)$ are equivalent.

To prove the main result of this section, we need the following.

**Proposition 5.5** ([15]). Let $\Pi, \Pi'$ be parameterized problems such that $\Pi'$ is in NP, and $\Pi$ is NP-complete. If there is a birelationalization from $\Pi$ to $\Pi'$ producing a birelational size, then $\Pi$ has a polynomial-size kernel.

**Theorem 5.9.** All ternary Permutation-CSPs parameterized above average have kernels with a quadratic number of variables.

**Proof.** By Theorem 5.8, it suffices to prove that the problems $\Pi_j$-AA, $j = 2, 7$, have kernels with quadratic number of variables.

**Case** $j = 2$. Denote constraints of $\Pi_2$-AA by $(u, v < w)$. Such a constraint is satisfied by a linear order $\alpha$ of $\{u, v, w\}$ if and only if $\alpha(v) < \alpha(w)$ and $\alpha(w) < \alpha(u)$. Consider the instance $(V, C, k)$ of $\Pi_2$-AA and construct an instance $(V, A, k)$ of ACYCLIC SUBDIGRAPH-AA as follows: if $(u, v < w) \in C$ then $(v, w)$ is added to $A$. Clearly, $(V, C, k)$ and $(V, A, k)$ are equivalent. By Theorem 5.6, in polynomial time, $(V, A, k)$ can be transformed into an equivalent instance $(V', A', k')$ of ACYCLIC SUBDIGRAPH-AA such that $|V'| = O(k^2)$ and $k'$ is bounded by a function of $k$ (in fact, $k' = k$). As in the proof of Case $j = 3$ of Theorem 5.4, from $(V', A', k')$ we can construct an equivalent instance $(V^*, C^*, k')$ of $\Pi_2$-AA such that $|V^*| = |V'| + 1 = O(k^2)$. Observe that $(V^*, C^*, k')$ is the required kernel.

**Case** $j = 7$. Denote constraints of $\Pi_7$-AA by $(u, v, w)$. Such a constraint is satisfied by a linear order $\alpha$ of $\{u, v, w\}$ if and only if either $\alpha(u) < \alpha(v) < \alpha(w)$ or $\alpha(v) < \alpha(w) < \alpha(u)$. Consider the instance $(V, C, k)$ of $\Pi_7$-AA and construct an instance $(V, A, k)$ of ACYCLIC SUBDIGRAPH-AA as follows: if $(u, v, w) \in C$ then $(u, v), (v, w)$ and $(w, u)$ are added to $A$. Let $\alpha$ be a linear order of $V$ and observe that if $(u, v, w)$ is satisfied by $\alpha$ then exactly two of the three arcs of $A$ generated by $(u, v, w)$ are satisfied by $\alpha$ and if $(u, v, w)$ is falsified by $\alpha$ then exactly one of the three arcs of $A$ generated by $(u, v, w)$ is satisfied by $\alpha$. Thus, $\alpha$ satisfies at least $|C|/2 + k$ constraints of $C$ if and only if $\alpha$ satisfies at least $2(|C|/2 + k) + (|C|/2 - k) = 3|C|/2 + k = |A|/2 + k$ arcs of $A$. By Theorem 5.6, in polynomial time, $(V, A, k)$ can be transformed into an equivalent instance $(V', A', k')$ of ACYCLIC SUBDIGRAPH-AA such that $|V'| = O(k^2)$ and $k' = k$ is bounded by a function of $k$.

Now construct an instance $(V'', C'', k')$ of $\Pi_7$-AA by setting $V'' = V' \cup \{z\}$, where $z \notin V'$, and $C'' = \{ (u, v, z) : (u, v) \in A' \}$. Let $\alpha$ be a linear order of $V''$ satisfying at least $|C''|/2 + k'$ constraints of $C''$. We may assume that $\alpha(z) = |V''|$ as moving the last element of a linear order to the front of the order does not falsify any constraint,
and so by repeatedly doing this we will move $z'$ to the last position in our order. Thus, $a$ satisfies at least $|A'|/2 + k'$ arcs of $A'$. Let $a$ be a linear order of $V'$ satisfying at least $|A'|/2 + k'$ arcs of $A'$. Extend $a$ to $V''$ by setting $a(z) = |V''|$ and observe that $a$ satisfies at least $|C'|/2 + k'$ constraints in $C'$. Hence, $(V'', C', k')$ is equivalent to $(V', A', k')$ and, thus, to $(V, C, k)$ implying that $(V'', C', k')$ is a kernel of III-AA.

$\square$

Theorem 5.9 can be made constructive in a similar way as Theorem 5.3.

Let us close this section by explaining why we did not solve Linear Ordering-AA “directly”, by showing that no finite sequence of “normal” reduction rules can provide a polynomial kernel for this problem. We call a reduction rule normal if it removes a set of constraints which will always have the average number of constraints satisfied no matter what order is used. Note that all reduction rules for Betweenness-AA and Acyclic Subdigraph-AA are normal.

We describe a directed graph $G_i$ with vertex set $V_i$ and a decomposition $C_i$ of the arc set of $G_i$ into 3-cycles. When $i = 0$ we have $V_0 = \{x_1, x_2, x_3\}$ and $C_0 = \{(x_1, x_2, x_3), (x_3, x_2, x_1)\}$. Note that the arc set of $G_i$ is the set of arc used in $C_i$.

When $i > 0$ we will construct $G_i$, $V_i$ and $C_i$ recursively. So assume that $G_{i-1}$, $V_{i-1}$ and $C_{i-1}$ have been constructed and let $G'_{i-1}$ be another copy of $G_{i-1}$ on vertex set $V'_{i-1}$ and with decomposition $C'_{i-1}$. Let $V_i = V_{i-1} \cup V'_{i-1}$ and note that $|V_i| = 2|V_{i-1}|$. Let $C = (x_0, x_1, x_i)$ be any 3-cycle in $C_{i-1}$ and let $C' = (x'_{i'}, x'_{i}, x'_{i'})$ be any 3-cycle in $C'_{i-1}$. Let $C_i$ be the set of 3-cycles in $C_{i-1} \setminus \{c\}, C'_{i-1} \setminus \{c'\}$ together with the following six 3-cycles:

\[
C_1 = (x_0, x_{i'}, x_{i'}), \quad C_2 = (x_{i'}, x_0, x_{i}), \quad C_3 = (x_{i'}, x_{i}, x_0), \\
C_4 = (x_{i'}, x_{i}, x_{i'}), \quad C_5 = (x_0, x_{i'}, x_{i'}), \quad C_6 = (x_{i'}, x_{i}, x_0).
\]

Call a directed graph $D$ symmetric if $(u, v) \in A(D)$ implies $(v, u) \in A(D)$.

**Lemma 5.15.** It holds that $|V_i| = 3 \cdot 2^i$ and $G_i$ is a symmetric directed graph with no parallel arcs for all $i \geq 0$. Furthermore, if $C_i \subset C_{i-1}$ is a proper nonempty subset of $C_i$ then the arcs of $C_i^*$ do not form a symmetric directed graph.

**Proof.** Since $|V_0| = 3$ and $|V_i| = 2|V_{i-1}|$ we have $|V_i| = 3 \cdot 2^i$ for all $i \geq 0$. Clearly, $G_0$ is symmetric with no parallel arcs. Assume that $G_i$ is symmetric with no parallel arcs for each $0 \leq j < i$ and consider $G_i$, $i > 0$. It is not difficult to see that by deleting the arcs in $c$ and $c'$ and adding the arcs in $C_1, \ldots, C_6$ we obtain a symmetric directed graph with no parallel arcs, which completes the proof of the first part of the lemma.

The second part of the lemma clearly holds when $i = 0$, so assume that $i > 0$ and that the second part holds for each $0 \leq j < i$. If $C_i^* \cap \{C_1^*, \ldots, C_6^*\} = \emptyset$ then we are done by induction, as either $C_i^* \cap C_{i-1}$ or $C_i^* \cap C_{i-1}^*$ is non-empty and therefore induces a non-symmetric directed subgraph.

So we may assume that $C_i^* \cap \{C_1^*, \ldots, C_6^*\} \neq \emptyset$. Since the arcs of of $C_i^*$ form a symmetric directed graph, we have the following. Due to the connection between $x_a$ and $x_a'$ we note that $C_1^* \subset C_i^*$ if and only if $C_5^* \subset C_i^*$. Analogously,

- $C_1^* \subset C_i^*$ if and only if $C_5^* \subset C_i^*$, due to $(x_{a'}, x_b)$,
- $C_2^* \subset C_i^*$ if and only if $C_4^* \subset C_i^*$, due to $(x_{a'}, x_c)$,
\[C^2 \in C^*_i \text{ if and only if } C^5 \in C^*_i, \text{ due to } (x', x),\]

- \[C^3 \in C^*_i \text{ if and only if } C^6 \in C^*_i, \text{ due to } (x', y),\]

- \[C^3 \in C^*_i \text{ if and only if } C^4 \in C^*_i, \text{ due to } (x', z).\]

Thus, if \(C^*_i \cap \{C^1, \ldots, C^6\} \neq \emptyset\) and the arcs of \(C^*_i\) form a symmetric directed graph then we must always have \(C^1, \ldots, C^6 \in C^*_i\).

As \(C^*_i\) is a proper subset of \(C_i\) we may assume, without loss of generality, that there is a 3-cycle in \(C_{i-1} \setminus \{C_i\}\) (otherwise it is in \(C_{i-1} \setminus \{C_i\}\) which does not belong to \(C^*_i\). By induction, the arc set of \(\{C_i \cup C^*_i\} \cap C_{i-1}\) does not form a symmetric directed graph. So the arcs of \(C^*_i\) do not form a symmetric directed graph either. \(\square\)

For each \(i \geq 0\) we construct an instance \((V_i, K_i)\) of LINEAR ORDERING as follows. For every 3-cycle in \(C_i\), say \((u, v, w)\), add the three constraints \((u, v, w), (v, w, u), (w, u, v)\) to \(K_i\). Let \((V_i, B_i)\) be the instance of BETWEENNESS which we associate with \((V_i, K_i)\) in Subsection 5.3.3 and let \((V_i, A'_i)\) and \((V_i, A''_i)\) be the two instances of ACYCLIC SUBDIAGRAM which we also associate with \((V_i, K_i)\) there. By Lemma 5.9, all linear orders \(\alpha\) of \(V_i\) satisfy

\[
de(v, K_i, \alpha) = \frac{1}{2} \left[\text{dev}(V_i, A'_i, \alpha) + \text{dev}(V_i, A''_i, \alpha) + \text{dev}(V_i, B_i, \alpha)\right]. \tag{5.5}\]

**Lemma 5.16.** It holds \(\text{dev}(V_i, K_i) = 0\), and if \(K^*_i\) is a nonempty proper subset of \(K_i\) then there exists a linear order of \(V_i\) that satisfies more than \(|K^*_i|/6\) constraints of \(K^*_i\).

**Proof.** As a 3-cycle \((u, v, w)\) in \(C_i\) gives rise to the betweenness constraints \((v, \{u, w\}), (w, \{v, u\})\) and \((u, \{w, v\})\) in \(B_i\) we can only satisfy \(|C_i|\) constraints in \(B_i\). Furthermore, a 3-cycle \((u, v, w)\) in \(C_i\) gives rise to two copies of the constraints \((u, v), (v, w)\) and \((w, u)\) in \(A'_i \cup A''_i\). Thus, we can think of an arc \((u, v)\) in \(G_i\) as giving rise to two copies of the acyclic directed subgraph constraint \((u, v)\). As \(G_i\) is symmetric this means that every constraint \((u, v)\) can be paired with a constraint \((v, u)\) so we can only satisfy half the constraints in \(A'_i \cup A''_i\). As we can only satisfy the average number of constraints in both \(A'_i \cup A''_i\) and \(B_i\), (5.5) implies that \(\text{dev}(V_i, K_i) = 0\), which proves the first part of the lemma.

For the sake of contradiction, assume that \(K^*_i\) is a nonempty proper subset of \(K_i\) for which \(\text{dev}(V_i, K^*_i) = 0\). Let \((V_i, B^*_i)\) be the instance of BETWEENNESS which we associate with \((V_i, K^*_i)\) in Subsection 5.3.3 and let \((V_i, A'_i)\) and \((V_i, A''_i)\) be the two instances of ACYCLIC SUBDIAGRAM which are also associated with \((V_i, K^*_i)\) Let \(Z, X, Y\) be the random variables associated with \((V_i, K^*_i), (V_i, B^*_i)\) and \((V_i, A'_i \cup A''_i)\), respectively. Note that \(\text{dev}(V_i, K^*_i) = 0\) is equivalent to \(E[Z^2] = 0\), which by the proof of Lemma 5.13 implies that \(E[X^2] = 0\) and \(E[Y^2] = 0\). Observe that by Lemma 5.6, this implies that if \((u, \{v, w\}) \in B^*_i\) then \((w, \{u, v\}), (v, \{u, w\}) \in B^*_i\). So, if \((u, v, w) \in K^*_i\), then \((v, u, w), (w, u, v) \in K^*_i\). Therefore, \(K^*_i\) can be thought of as being obtained from a proper subset, \(C^*_i\), of the 3-cycles \(C_i\). Observe that by Lemma 5.15 some arc \((u, v)\) belongs to a 3-cycle in \(C^*_i\), but the arc \((v, u)\) does not belong to such a 3-cycle. However, this implies that \((u, v) \in A'_i \cup A''_i\), but \((v, u) \notin A'_i \cup A''_i\). Thus, \(E[Y^2] > 0\) by Lemma 5.8, the desired contradiction. \(\square\)
Lemma 5.16 implies that each instance \((V_i, K_i)\) of LINEAR ORDERING-AA cannot be reduced by any normal reduction rule, except the one that removes all constraints in the instance. Therefore, no finite set of normal reduction rules can guarantee that one always gets either the empty instance or an instance where one can do better than the average. For both BETWEENNESS-AA and ACYCLIC SUBDIGRAPH-AA we only needed one normal reduction rule to get such a guarantee. This is another indication that LINEAR ORDERING-AA is a more difficult problem.

5.4 Concluding Remarks

In this chapter we presented combinatorial approaches to determine the parameterized complexity of independent set and permutation constraint satisfaction problems parameterized above tight lower bounds on their solution value. The ideas presented in this chapter also apply to problems where one aims to construct more complex structures such as trees. For example, we can prove the following.

**Theorem 5.10.** Given a triplet set \(T\), the problem of deciding whether some tree contains at least \(|T|/3 + k\) triplets from \(T\) as homeomorphic subtree is fixed-parameter tractable and admits a kernel with \(O(k^2)\) triplets.

For quartets, it was asked by Niedermeier [205] whether MINIMUM QUARTET INCONSISTENCY is fixed-parameter tractable when parameterized as follows: decide whether removing at most \((n - 3)/2 + k\) quartets from \(Q\) yields a set \(Q'\) that is consistent with a tree, for parameter \(k\). We can answer this question in the affirmative by referring to the results of Wu et al. [252].

The area of parameterizations above tight lower bound is still in its infancy, with many interesting open research directions.

First, several questions from the survey of Mahajan et al. [189] remain open. For example, the parameterized complexity of MAX LIN-2 parameterized above tight lower bound remains unknown, despite recent efforts [63, 64].

Second, we re-Pose the question of Niedermeier [205] whether INDEPENDENT SET restricted to planar graphs parameterized above tight lower bound is fixed-parameter tractable. We further pose the question whether the INDUCED MATCHING problem, restricted to planar graphs \(G\) and parameterized above tight lower bound \(|V(G)|/28 [89]\), is fixed-parameter tractable.

Third, due to their practical importance, it would be interesting to see whether the quadratic kernel for (say) BETWEENNESS-AA can be improved to a linear kernel, or whether subquadratic kernels can be ruled out modulo a collapse of the polynomial hierarchy. Further, we would like to know whether approximation ratios better than 1/3 are achievable in polynomial time for reduced instances of BETWEENNESS. From the combinatorial perspective, upper bounds on the maximum number of satisfiable constraints in reduced instances are worth investigating.

In this chapter we considered ternary Permutation-CSPs. The only binary Permutation-CSP is ACYCLIC SUBDIGRAPH, the problem whose above average parameterization was studied by Gutin et al. [143]. It would be interesting to investigate above average parameterizations for Permutation-CSPs of higher arities.
In this chapter we study fixed-parameter algorithms for parameterized problems \( \Pi \) that run in time that is subexponential in the parameter. That is, algorithms solving instances \((x, k)\) of \( \Pi \) in time \( 2^{o(k)} \cdot |x|^{O(1)} \); we refer to such algorithms as \textit{subexponential fixed-parameter algorithms} or \textit{subexponential FPT-algorithms}. Algorithms of this kind provide an exponential speed-up over fixed-parameter algorithms running in time \( 2^{\Omega(k)} \cdot |x|^{O(1)} \), which often result from bounded search tree approaches and algorithms based on iterative compression. Compared with the large body of literature on fixed-parameter algorithms, much less is known about the existence of subexponential fixed-parameter algorithms. Most of the results of this kind are centered around graph subset problems, on graph classes which are either very sparse—such as planar graphs, or very dense—such as tournaments. For sparse graph classes, graph separator theorems play an important role in obtaining subexponential fixed-parameter algorithms, whereas for dense graph classes, randomized coloring procedures have been the method of choice.

Our results in this chapter concern algorithms for both sparse and dense instances. We first give a subexponential fixed-parameter algorithm for an important problem from phylogenetics, namely the problem of removing at most \( k \) triplets from a dense triplet set \( T \) such that some rooted binary tree contains all other triplets as homeomorphic subtree. Second, we obtain subexponential fixed-parameter algorithms for almost all bidimensional problems on graphs excluding a fixed minor, and these algorithms require only polynomial space. That is, for inputs of size \( n \), the algorithm requires space \( n^c \) for some constant \( c \). These algorithms significantly improve the space demand of previously known subexponential fixed-parameter algorithms that require exponential space, while preserving a running time that is subexponential in the parameter.

Our results are motivated by the general research question which fixed-parameter tractable problems admit polynomial-space FPT-algorithms. For many fixed-parameter tractable problems \( \Pi \) it is a non-straightforward task to devise an FPT-algorithm that uses only polynomial space, whereas it is often easy to show that \( \Pi \) belongs to XP by enumerating all size-\( k \) vertex subsets or edge subsets and answering yes if and only if one such subset has the desired property \( \pi \). If the property \( \pi \) can be tested in polynomial time then this brute-force procedure runs in polynomial space. The following is a sufficient condition for a parameterized problem to
be polynomial-space fixed-parameter tractable.

**Theorem 6.1** ([185]). *Any fixed-parameter tractable problem with a polynomial-space XP-algorithm admits a polynomial-space fixed-parameter algorithm.*

**Proof.** Let \((x, k)\) be an instance of a fixed-parameter tractable problem \(\Pi\). Let \(A_1\) be an algorithm solving \(\Pi\) on input \((x, k)\) in time \(f(k) \cdot |(x, k)|^c\) and let \(A_2\) be an algorithm solving \(\Pi\) on input \((x, k)\) in time \(|(x, k)|^{g(k)}\) and polynomial space, for computable functions \(f, g\) and constant \(c\).

Now if \(|(x, k)| \leq f(k)\) then execute \(A_2\), which requires time at most \(f(k)^{g(k)}\) and polynomial space. And if \(f(k) \leq |(x, k)|\) then execute \(A_1\), which requires time at most \(|x| \cdot |x|^c = |x|^{c+1}\) and hence polynomial space. \(\square\)

A more appropriate question, to which we shed light on in this chapter, is thus to ask which parameterized problems admit polynomial-space subexponential FPT-algorithms.

### 6.1 Minimum Triplet Inconsistency

We study the parameterized complexity of inferring supertrees from sets of rooted triplets, an important problem in phylogenetics. For a set \(L\) of labels and a dense set \(T\) of rooted (unrooted) binary trees distinctly leaf-labeled by 3-subsets of \(L\) we seek a tree distinctly leaf-labeled by \(L\) and containing all but at most \(k\) triplets from \(T\) as homeomorphic subtree. Our results are the first polynomial kernel, with \(O(k^2)\) labels, and a subexponential fixed-parameter algorithm running in time \(2^{O(k^{1/3} \log k)} + O(|L|^4)\), for this problem.

In phylogenetics, distinctly leaf-labeled trees represent the evolutionary history of a set of species, each species corresponding to a label. *Supertree methods* are widely used in this field, in order to construct a large tree from smaller trees on overlapping subsets of species. As the inference of the small trees can be computationally expensive, it is desirable to keep their size as low as possible. The simplest approach in this setting consists in inferring the smallest possible informative trees. Such trees are either *triplets*, rooted binary trees on three labels, or *quartets*, unrooted binary trees on four labels, in the rooted respectively unrooted setting. Quartets methods received prominent attention over the last decade, whereas triplets methods were somewhat overlooked though they enjoy similar interesting properties and may be less computationally expensive.

Let \(L\) be a set of labels. A set \(T\) of rooted (unrooted) binary trees distinctly leaf-labeled by subsets of \(L\) is *consistent* if there exists a rooted (unrooted) binary tree distinctly leaf-labeled by \(L\) and containing every element of \(T\) as homeomorphic subtree; and *inconsistent* otherwise. Deciding consistency for a set of triplets is polynomial-time solvable [4]; in contrast, this problem is NP-hard for quartets [231]. For an inconsistent triplet set \(T\), two approaches have been studied to obtain a consistent triplet set from it.

The first approach is to find a smallest subset \(L' \subseteq L\) such that removing all leaves labeled by elements from \(L'\) from triplets in \(T\) yields a consistent set of trees.
The problem of finding \( L' \) is the dual of the maximum agreement supertree problem \([134, 163]\), and we call this problem MINIMUM LABEL INCONSISTENCY (MLI). The parameterization of MLI by \(|L'|\) is denoted as \((k)\)-MLI; this problem is NP-hard and fixed-parameter intractable. The restriction of \((k)\)-MLI to instances that are dense, that is \( T \) contains at least one triplet for each 3-subset of \( L \), is fixed-parameter tractable; call this problem \((k)\)-DENSE MLI. Fixed-parameter tractability of \((k)\)-DENSE MLI follows from that for dense triplet sets, consistency has a characterization in terms of obstructions involving at most four labels and three triplets. More precisely, for a dense triplet set \( T \) on \( n \) labels, we can build in time \( O(n^k) \) the set of obstructions, and in time \( O(n^3) \) either find an obstruction or decide if \( T \) is consistent \([134]\). These results lead to an \( O(4^k n^3) \)-time algorithm for \((k)\)-DENSE MLI \([134]\).

The second approach is to remove a smallest subset \( T' \) of triplets from \( T \) such that the set \( T \setminus T' \) is consistent. The problem of finding \( T' \) is the ROOTED TRIPLET INCONSISTENCY (RTI) problem \([45, 47, 251]\). We denote the restriction of RTI to dense instances, and parameterized by the size of \( T' \), by \((k)\)-DENSE RTI. This restriction of RTI is still NP-hard. Our first result is a simple \( O(4^k n^3) \)-time algorithm for \((k)\)-DENSE RTI, based on a characterization of consistency in terms of obstructions and given in Section 6.1.2. For general instances, the parameterization of RTI by \(|T'|\) is not fixed-parameter tractable unless some unlikely collapse of complexity classes occurs.

Our second result is the first polynomial kernel for \((k)\)-DENSE RTI: in Section 6.1.3, we describe a kernelization that produces in time \( O(n^k) \) a kernel with \( O(k^2) \) labels.

While interesting by itself, our polynomial kernel serves as the basis for our third result. In Section 6.1.4, we present a subexponential fixed-parameter algorithm for \((k)\)-DENSE RTI. The algorithm applies the method of chromatic coding, which was recently introduced by Alon et al. \([18]\) to solve the FEEDBACK ARC SET problem in tournaments in time \( 2^{O(k^{1/2} \log k)} + O(n^3) \). Similarly, chromatic coding provides a subexponential fixed-parameter algorithm for BETWEENNESS on dense instances when parameterized by the number \( k \) of constraints to be removed, running in time \( 2^{O(k^{1/2} \log k)} + O(n^4) \) \([225]\). Let us remark that the trees we are constructing possess a more complex structure than the linear orderings encountered in ranking problems such as FEEDBACK ARC SET in tournaments and BETWEENNESS.

Chromatic coding is a variant of the color coding technique by Alon et al. \([20]\). It requires several ingredients: (i) a kernel of quadratic size, (ii) an algorithm solving instances of size \( n \) and colored with \( \ell \) colors in time \( n^{O(\ell)} \), (iii) a coloring lemma stating for a given solution of size \( k \), a “proper coloring” of the solution with \( \ell = O(k^{1/3}) \) colors exists and can be found in subexponential time. We obtain (i) and (ii) for \((k)\)-DENSE RTI, while (iii) follows from results of Alon et al. \([18, 225]\). Overall, we obtain an algorithm for \((k)\)-DENSE RTI running in time \( O(n^k) + 2^{O(k^{1/3} \log k)} \). This algorithm provides an exponential speed-up over our search-tree based \( O(4^k n^3) \)-time algorithm. Note that \( k \) can be as large as \( 3 \binom{n}{3} - \frac{n}{2} \), and for this value our algorithm yields a running time of \( 2^{O(n \log n)} \) which is asymptotically not better than the brute-force algorithm which enumerates every tree with \( n \) labels and checks consistency of \( T \) with each such tree. However, for practical instances, \( k \) can be
expected to be much smaller than $3(n)^n - n$, which makes our result valuable in real-life applications.

6.1.1 Trees and Consistency

Let $L = \{1, \ldots, n\}$ be a set of labels. A tree over $L$ is a rooted binary tree $T$ whose leaves are injectively labeled by the elements of $L$; denote by $L(T)$ the set of labels appearing at the leaves of $T$. We will use a parenthesized notation for trees: if $T_1, \ldots, T_m$ are trees over $L$ with disjoint label sets, we denote by $(T_1, \ldots, T_m)$ the tree over $L$ whose root has children $T_1, \ldots, T_m$ as subtrees. For subsets $L' \subseteq L$ the restriction of $T$ to $L'$ is the homeomorphic subtree of $T$ containing leaves labeled by elements of $L'$. A triplet over $L$ is a tree over $L$ with three leaves; we usually denote the triplet $((a, b), c)$ by $(\{a, b\}, c)$ to indicate the symmetry between $a$ and $b$. A triplet set over $L$ is a set of triplets over $L$. A triplet set $T$ over $L$ is dense (minimally dense) if for each 3-set $\{x, y, z\} \subseteq L$ there is at least (exactly) one triplet in $T$ over $\{x, y, z\}$. We sometimes simply say that $T$ is a (dense, minimally dense) triplet set. A triplet set is simple if it does not contain two triplets of the form $(\{a, b\}, c), (\{a, c\}, b)$. Observe that a triplet set $T$ is minimally dense if it is both simple and dense. For subsets $L' \subseteq L$ the restriction of $T$ to $L'$ is the subset of $T$ that consists of the triplets $t$ with $L(t) \subseteq L'$.

A tree $T$ over $L$ is consistent with a triplet $t = (\{a, b\}, c)$ (or symmetrically, $t$ is consistent with $T$) if $T|\{a, b, c\} = t$. We denote by $rt(T)$ the set of triplets over $L$ that are consistent with $T$. A triplet set $T$ is consistent with a tree $T$ if $T \subseteq rt(T)$. Note that if $T$ is minimally dense then this must be an equality. A conflict in $T$ is a set $C \subseteq L$ such that $T|C$ is not consistent. A t-conflict in $T$ is a non-consistent set $S \subseteq T$, and an st-conflict in $T$ is a t-conflict isomorphic to either $\{(\{a, b\}, c), (\{c, d\}, b), (\{b, d\}, a)\}$ or $\{(\{a, b\}, c), (\{c, d\}, b), (\{a, d\}, b)\}$. Each triplet $(\{a, b\}, c)$ has three editions: $(\{a, b\}, c), (\{a, b\}, c)$ and $(\{a, b\}, c)$. By editing triplet $t$ into $t'$ in a triplet set $T$ we mean replacing $t$ by its edition $t'$. Observe that for any triplet $t$, any tree is consistent with at most one edition of $t$. For a triplet set $T$, a peacemaker of $T$ is a triplet set $T' \subseteq T$ such that $T \setminus T'$ is consistent. Hence, an instance $(T, k)$ of $(k)$-Dense RTI is a “yes”-instance if and only if $T$ has a peacemaker of size at most $k$.

For a triplet set $T$ over $L$ and a set $L' \subseteq L$, the Aho graph [4] is the undirected edge-labeled graph $G(T, L')$ with vertex set $L'$, and edges $\{(u, v), w\}$ labeled by the set of elements $w \in L'$ such that $((u, v), w) \in T$, if such $w$ exist.

**Proposition 6.1 ([4]).** A triplet set $T$ is consistent if and only if for each set $L' \subseteq L$ of size at least three the Aho graph $G(T, L')$ is disconnected.

6.1.2 A Simple Fixed-Parameter Algorithm

We prove a local characterization of consistency for minimally dense triplet sets and derive a simple fixed-parameter algorithm for $(k)$-Dense RTI from it.

**Lemma 6.1.** Let $T$ be a simple triplet set.

1. If $|L(T)| \leq 3$ or $|T| \leq 2$ then $T$ is consistent.
Lemma 6.3. For a minimally dense triplet set $T$, the following are equivalent:

1. $|T| = 3$ and $|L(T)| > 4$ then $T$ is consistent.
2. $|L(T)| = 3$ then $T$ is the singleton $\{t\}$, since $T$ is simple. Hence $T$ is consistent with the tree $t$. If $|T| \leq 2$ we show that $T$ is consistent by Proposition 6.1. Let $L' \subseteq L(T)$. If $|L'| \leq 3$ then $G(T, L')$ is clearly disconnected; and if $|L'| \geq 4$ then $G(T, L')$ is disconnected since $G(T, L')$ has at most two edges.

Proof. (1) If $|L(T)| = 3$ then $T$ is the singleton $\{t\}$, since $T$ is simple. Hence $T$ is consistent with the tree $t$. If $|T| \leq 2$ we show that $T$ is consistent by Proposition 6.1. Let $L' \subseteq L(T)$. If $|L'| \leq 3$ then $G(T, L')$ is clearly disconnected; and if $|L'| \geq 4$ then $G(T, L')$ is disconnected since $G(T, L')$ has at most two edges.

(2) Observe that $G(T, L)$ has at least 5 vertices and at most three edges, so it is disconnected. Now, if $L' \subseteq L$ then $T|L'$ contains at most two triplets and hence, $G(T, L')$ is disconnected by (i). We conclude that $T$ is consistent, by Proposition 6.1.

Lemma 6.2. For minimally dense triplet sets $T$, the following are equivalent:

1. $T$ has no conflict of size 4.
2. For each 4-set $\{a, b, c, d\} \subseteq L(T)$, if $\{(a, b), (b, c), (c, d)\} \in T$ then $\{(a, b), d\}, (\{a, c\}, d) \in T$.
3. For each 4-set $\{a, b, c, d\} \subseteq L(T)$, if $\{(a, b), (b, c), (c, d)\} \in T$ then $\{(a, b), d\}, (\{a, c\}, d) \in T$.

Proof. (1) $\Rightarrow$ (2): Let $C = \{a, b, c, d\} \subseteq L$ be such that $\{(a, b), (b, c), (c, d)\} \in T$. As $C$ is not a conflict, $T|C$ is consistent with some tree $T$. Then $T = ((a, b), c, d)$, which implies that $\{(a, b), (b, c), d\}, (\{a, c\}, d) \in T$.

(2) $\Rightarrow$ (3): Let $C = \{a, b, c, d\} \subseteq L$ be such that $\{(a, b), (b, c), (c, d)\} \in T$. By symmetry of $b$ and $c$, it suffices to show that $\{(a, b), d\} \in T$. Suppose not; then either $\{(a, d), b\} \in T$ or $\{(b, d), a\} \in T$. If $\{(a, d), b\} \in T$ then $\{(a, b), (b, c), (c, d)\} \in T$ and $\{(d, b), c\} \notin T$, contradicting (2). If $\{(b, d), a\} \in T$ then $\{(d, b), a\}, (\{a, b\}, c) \in T$ and $\{(d, b), c\} \notin T$, contradicting (2).

(3) $\Rightarrow$ (1): Let $C = \{a, b, c, d\} \subseteq L$; we show that $T|C$ is consistent. Suppose that $G(T, C)$ has two matching edges $\{a, b\}$ and $\{c, d\}$. Assume, without loss of generality, that $T$ contains triplets $\{(a, b), c\}, (\{c, d\}, b\)$. From (3) it follows that $T$ also contains triplets $\{(a, b), d\}, (\{c, d\}, a)$, hence $T|C$ is consistent with the tree $T = ((a, b), (c, d))$.

Suppose now that $G(T, C)$ does not have two matching edges. If $G(T, C)$ has at most two edges then by Proposition 6.1 and Lemma 6.1(1) the set $T|C$ is consistent. If $G(T, C)$ has exactly three edges $\{a, b\}, (c, d)$ incident to the same vertex $a$ then one of these edges, $\{a, b\}$ say, has two labels, while the other two edges $\{a, c\}$ and $\{a, d\}$ have one label. Edge $\{a, b\}$ must then be labeled by $\{c, d\}$, hence $T$ contains triplets $\{(a, b), c\}, (\{a, b\}, d)$. Edge $\{a, c\}$ cannot be labeled by $b$, since $\{(a, b), c\} \in T$. Thus, it must be labeled by $d$ and so $\{(a, c), d\} \in T$. We obtain a contradiction by observing that edge $\{a, d\}$ can neither be labeled by $b$ (since $\{(a, b), d\} \in T$) nor by $c$ (since $\{(a, c), d\} \in T$).

The next lemma is analogous to a result for quartets by Bandelt and Dress [28, 127].

Lemma 6.3. For a minimally dense triplet set $T$, the following are equivalent:

(1) $T$ has no conflict of size 4.
(2) For each 4-set $\{a, b, c, d\} \subseteq L(T)$, if $\{(a, b), (b, c), (c, d)\} \in T$ then $\{(a, b), d\}, (\{a, c\}, d) \in T$.
(3) For each 4-set $\{a, b, c, d\} \subseteq L(T)$, if $\{(a, b), (b, c), (c, d)\} \in T$ then $\{(a, b), d\}, (\{a, c\}, d) \in T$.
6. SUBEXPONENTIAL PARAMETERIZED ALGORITHMS

(1) \( T \) is consistent,

(2) \( T \) contains no conflict of size 4,

(3) \( T \) contains no t-conflict of size 3,

(4) \( T \) contains no st-conflict.

Proof. (1) \( \Rightarrow \) (2): Clearly, if \( T \) is consistent, then for each set \( C \) of size 4, \( T|C \) is consistent, and thus \( C \) is not a conflict.

(2) \( \Rightarrow \) (1): This follows from a result by Guillemot and Berry [134, Theorem 5].

(2) \( \Rightarrow \) (3): Suppose that \( T \) contains a t-conflict \( S \) of size 3. Let \( C = L(S) \); then \( |C| = 4 \) by Lemma 6.1(2). Hence \( T|C \) is inconsistent, since it contains an inconsistent subset \( S \). We conclude that \( C \) is a conflict of size 4.

(3) \( \Rightarrow \) (4): Follows from that an st-conflict is a t-conflict of size 3.

(4) \( \Rightarrow \) (2): Follows from the implication “(3) \( \Rightarrow \) (1)” of Lemma 6.2.

Let \( T \) be a triplet set over \( L \), and let \( T \) be a tree over \( L \). We denote by \( i(T, T) := (T|L(T)) \setminus rt(T) \) the set of triplets in \( T \) that are not consistent with \( T \), and define \( I(T, T) = |i(T, T)| \).

Problem RTI takes as input a triplet set \( T \) over \( L \), and seeks a tree \( T \) over \( L \) with \( I(T, T) \) minimum. The value of the optimum is denoted by \( MRTI(T) \). Note that determining \( MRTI(T) \) is equivalent to determining (i) the minimum number of triplets in \( T \) to be removed to obtain a consistent triplet set, and (ii) the minimum number of triplets to be edited in \( T \) to obtain a consistent triplet set.

For the needs of the algorithm, we consider an annotated version of the problem where certain triplets are locked, meaning that they cannot be edited. We will use a bounded-search approach: at each step, we identify an st-conflict, edit one triplet and lock it. We will see that for a given st-conflict, four different editions will cover all possible cases. Before stating the algorithm, we need a preparatory observation.

Lemma 6.4. Any tree on \( \{a, b, c, d\} \) is consistent with one of the triplets \( (\{b, c\}, a), (\{a, c\}, b), (\{b, d\}, c), (\{a, b\}, d) \).

A simple fixed-parameter algorithm for \((k)\)-Dense RTI now works as follows. Given a minimally dense triplet set \( T \), using an algorithm by Guillemot and Berry [134, Theorem 5], in \( O(n^3) \) time we either find an st-conflict or conclude that \( T \) is consistent. If we find an st-conflict \( \{t_1, t_2, t_3\} \) with \( t_1 = (\{a, b\}, c), t_2 = (\{c, d\}, b) \) and \( t_3 \in \{(\{b, d\}, a), (\{a, d\}, b)\} \), then we branch on the four following cases:

(1) edit \( t_1 \) into \( (\{b, c\}, a) \),

(2) edit \( t_1 \) into \( (\{a, c\}, b) \),

(3) edit \( t_2 \) into \( (\{b, d\}, c) \),

(4) edit \( t_3 \) into \( (\{a, b\}, d) \).
In each case we lock the new triplet obtained from editing, and repeat with \( k := k - 1 \). Correctness of the branching step follows from Lemma 6.4, and this leads to an algorithm running in time \( O(4^4n^3) \).

We conclude with the following result.

**Theorem 6.2.** \((k)\)-DENSE RTI can be solved in time \( O(4^k n^3) \) for triplet sets over sets of \( n \) labels.

The algorithm can be modified to run in time \( O(4^k n + n^4) \) time, using ideas similar to those of Gramm and Niedermeier [127] for quartet inconsistency: build in time \( O(n^4) \) the set \( C \) of st-conflicts, then at each branching step identify an st-conflict, edit a triplet and update \( C \) in \( O(n) \) time.

### 6.1.3 A Polynomial Kernel

We give two reduction rules for \((k)\)-DENSE RTI that together will lead to a kernel with a quadratic number of labels. As before, we allow some triplets to be locked.

Let \( L \) be a set of labels and let \( T \) be a set of triplets over \( L \). Let \( L_1(T) \subseteq L(T) \) be the set of labels that appear in some st-conflict of \( T \) and let \( L_2(T) = L(T) \setminus L_1(T) \). Let \( t \) be a triplet in \( T \). A sunflower with center \( t \) is a family \( \{C_1, \ldots, C_m\} \) of st-conflicts of \( T \) such that every \( C_i \) contains \( t \) and distinct st-conflicts \( C_i, C_j \) have distinct fourth labels, that is, \( L(C_i) \cap L(C_j) = L(t) \) whenever \( i \neq j \). The next observation allows us to prove correctness of the reduction rules.

**Observation 6.1.** For a sunflower \( \{C_1, \ldots, C_m\} \) with center \( t \), the sets \( C_1 \setminus \{t\}, \ldots, C_m \setminus \{t\} \) are disjoint.

**Observation 6.2.** If a locked triplet is the center of a sunflower of size strictly larger than \( k \) then \((T, k)\) is a “no”-instance.

The first reduction rule handles unlocked triplets which are involved in a large number of conflicts. It relies on the following lemma.

**Lemma 6.5.** Let \( t_1, t_2 \) be two triplets such that \( L(t_1) = \{a, b, c\} \) and \( L(t_2) = \{b, c, d\} \). Then either

1. there is a unique tree \( T \) on \( \{a, b, c, d\} \) consistent with \( t_1 \) and \( t_2 \); or
2. for each triplet \( t_3 \) on \( \{a, b, d\} \) the set \( T_{123} := \{t_1, t_2, t_3\} \) is consistent.

**Proof.** By symmetry, it suffices to consider the following cases:

- \( t_1 = (\{b, c\}, a), t_2 = (\{b, c\}, d) \). Then (2) holds, since for
  - \( t_3 = (\{a, b\}, d) \), the tree \( T = ((b, c), a, d) \) is consistent with \( T_{123} \);
  - \( t_3 = (\{a, d\}, b) \), the tree \( T = ((b, c), (a, d)) \) is consistent with \( T_{123} \);
  - \( t_3 = (\{b, d\}, a) \), the tree \( T = ((b, c), d, a) \) is consistent with \( T_{123} \);

- \( t_1 = (\{b, c\}, a), t_2 = (\{b, c\}, d) \). Then (1) holds, since only the tree \( T = ((b, d), c, a) \) is consistent with \( t_1, t_2 \).

- \( t_1 = (\{b, a\}, c), t_2 = (\{b, d\}, c) \). Then (2) holds, since for
\[ t_3 = \{ \{ a, b \}, d \}, \] the tree \( T = (((a, b), d), c) \) is consistent with \( T_{123} \);

\[ t_3 = \{ \{ a, d \}, b \}, \] the tree \( T = (((a, d), b), c) \) is consistent with \( T_{123} \);

\[ t_3 = \{ \{ b, d \}, a \}, \] the tree \( T = ((a, b), d), c) \) is consistent with \( T_{123} \).

- \[ t_1 = \{ \{ b, a \}, c \}, t_2 = \{ \{ c, d \}, b \}. \] Then (1) holds, since only the tree \( T = ((a, b), (c, d)) \) is consistent with \( t_1, t_2. \)

**Corollary 6.1.** For an st-conflict \( C = \{ t_1, t_2, t_3 \} \) of \( T \) there exists a unique way to edit \( t_3 \) to a triplet \( t'_3 \) such that \( C' = \{ t_1, t_2, t'_3 \} \) is consistent.

**Proof.** The set \( C \) must involve the four labels \( a, b, c, d. \) We assume, without loss of generality, that \( L(t_1) = \{a, b, c\}, L(t_2) = \{b, c, d\}. \) Then we are in the conditions of Proposition 6.5, and we must be in (1) there since \( C \) is an st-conflict. It follows that there is a unique tree \( T' \) on \( \{a, b, c, d\} \) that is consistent with \( t_1 \) and \( t_2. \) Hence the triplet \( T'(L(t_3)) \) is the only way to edit \( t_3 \) to achieve consistency.

By Corollary 6.1, if a triplet \( t \) belongs to an st-conflict \( C, \) then in order to achieve consistency there is a unique way to edit \( t \) into a triplet \( t' \) without editing the other triplets of \( C. \) Let \( S(t, C) \) denote this alternative triplet \( t'. \) We are ready to formulate the reduction rules and prove their correctness.

**Reduction Rule 6.1.** If an unlocked triplet \( t \) is the center of a sunflower \( \{C_1, \ldots, C_m\} \) with \( m > 2k \) then edit \( t \) into the majority element \( t' \) among the triplets \( S(t, C), \) lock \( t' \) and set \( k' = k - 1. \)

**Lemma 6.6.** **Reduction Rule 6.1** is correct.

**Proof.** Suppose there is a solution editing at most \( k \) triplets. Let \( C_1, \ldots, C_i \) be the family of st-conflicts containing a triplet distinct from \( t \) which was edited. By Observation 6.1, it follows that \( i \leq k. \) For the st-conflicts \( C_{i+1}, \ldots, C_m, \) they must all yield the same triplet \( t' = S(t, C_p). \) Then \( t \) has been edited into this triplet \( t', \) which must be the majority element among the edited triplets.

The second reduction rule removes labels not appearing in any st-conflict of \( T. \) Note that the situation is similar for the Feedback Arc Set problem in tournaments, where we can safely remove vertices not involved in a 3-cycle without changing the optimum. As pointed out by Guo [136], for edge-modification problems in general it is not safe to remove vertices outside of an obstruction (unlike the case of vertex-deletion problems). However, it is interesting to observe that this holds in most known kernelizations for edge-modification problems.

**Reduction Rule 6.2.** Remove the labels of \( L_2(T). \)

**Lemma 6.7.** **Reduction Rule 6.2** is correct.

**Proof.** Say that an \( L \)-tree is a rooted binary tree with each leaf \( x \) labeled by a subset \( L_x \subseteq L(T) \) such that the sets \( L_x \) partition \( L(T). \) An \( L \)-tree \( T \) induces a set of triplets \( r(T), \) where \( \{\{u, v\}, w\} \in r(T) \) if and only if there exist leaf labels \( x, y, z \) of \( T \) with \( u \in L_x, v \in L_y, w \in L_z \) and \( \{\{x, y\}, z\} \) is a triplet of \( T. \) We say that \( T \) comlies with \( T \) if \( r(T) \subseteq T. \) For a label set \( S \subseteq L(T), \) say that \( T \) resolves \( S \) if each label \( u \in S \) corresponds to a leaf of \( T \) labeled by \( \{u\}. \)
Claim. There exists a binary L-tree $T$ complying with $T$ and resolving $L_2(T)$.

Proof of the claim. We prove that if $u \in L_2(T)$ then there exists an L-tree $T$ that complies with $T$ and resolves $\{u\}$.

Define relations $<$ and $\equiv$ on $L \setminus \{u\}$ by
\[
\begin{align*}
    x < y & : \iff (\{u, x\}, y) \in T, \\
    x \equiv y & : \iff (\{x, y\}, u) \in T.
\end{align*}
\]

Since $u$ does not belong to a conflict of $T$, the relations $<$ and $\equiv$ define a total preorder on $L(T) \setminus \{u\}$. For instance, $\equiv$ is transitive: if $a \equiv b$ and $b \equiv c$ then $(\{a, b\}, u) \in T$ and $(\{b, c\}, u) \in T$, which implies that $(\{a, c\}, u) \in T$ since otherwise $u$ would appear in a t-conflict of $T$. It follows that we can order the equivalence classes of $\equiv$ as $C_1, \ldots, C_m$, such that $x < y$ holds whenever $x \in C_i, y \in C_j$ with $i < j$. Besides, if $x \in C_i, y \in C_j, z \in C_k$, with $i, j < k$, then $(\{x, y\}, z) \in T$ if $i, j < k$. We conclude that $T$ complies with $T$, and also resolves $\{u\}$.

The proof of the claim is completed by a straightforward induction on $|T|$.

To prove correctness of Reduction Rule 6.2, we show that if $T' = T|((L(T) \setminus L_2(T))$ then $\text{MRTI}(T') = \text{MRTI}(T)$. The inequality $\text{MRTI}(T') \leq \text{MRTI}(T)$ follows from the inclusion $T' \subseteq T$. To prove the reverse inequality $\text{MRTI}(T) \leq \text{MRTI}(T')$, let $T$ be an L-tree given by the claim and let $S$ be an optimum solution for $T'$. Let $S'$ be the tree obtained from $T$ by substituting each leaf labeled by $L' \subseteq L_1(T)$ with the tree $S|L'$. Then $I(T, S') = I(T \setminus rt(T), S') \leq I(T', S)$ since for each triplet $t = (\{u, v\}, w) \in T \setminus rt(T)$ it holds that $L(t) \subseteq L_1(T)$, and if $t$ is consistent with $S$ if and only if it is consistent with $S'$. It follows that $\text{MRTI}(T) \leq \text{MRTI}(T')$.

An instance $(T, k)$ to which neither Reduction Rule 6.1 nor 6.2 applies is called reduced. For reduced "yes"-instances $(T, k)$ we can bound the sizes of $L(T) = L_1(T)$ in terms of $k$.

Lemma 6.8. If $(T, k)$ is a reduced "yes"-instance then $|L_1(T)| \leq 2k^2 + 3k$.

Proof. Suppose that $T$ can be made consistent by editing triplets $t_1, \ldots, t_i$ with $i \leq k$. Then $L_1(T) = L'_1(T) \cup L''_1(T)$, where $L'_1(T)$ is the set of labels appearing in some $t_j$ and $L''_1(T) = L_1(T) \setminus L'_1(T)$. Clearly, $|L'_1(T)| \leq 3k$.

We show that $|L''_1(T)| \leq 2k^2$. For each $x \in L''_1(T)$ let $C_x$ be an st-conflict containing $x$. Then $C_x$ must contain some triplet $t_j$ for $j \in \{1, \ldots, i\}$, let $j := j(x)$. Now, since $(T, k)$ is reduced, a sunflower with center $t_j$ has size at most $2k$. It follows that for each $1 \leq j \leq i$, there are at most $2k$ elements $x \in L''_1(T)$ such that $j(x) = j$. We conclude that $|L''_1(T)| \leq 2ki \leq 2k^2$.

Putting all together we obtain the following kernelization for $(k)$-Dense RTI.
Theorem 6.3. For \((k)\)-Dense RTI a kernel with at most \(2k^2 + 3k\) labels exists and can be constructed in time \(O(|L|^4)\).

Proof. Let \((T, k)\) be a reduced instance. By Lemma 6.8, if \(L_1(T)\) contains more than \(2k^2 + 3k\) labels then \((T, k)\) is a “no”-instance. Otherwise, \((T, k)\) is a problem kernel for \((k)\)-Dense RTI with at most \(2k^2 + 3k\) labels.

We justifiy that the kernel can be constructed in \(O(n^4)\) time. The idea is to maintain

- the set \(C\) of st-conflicts,
- for each triplet \(t \in T\), the size \(n(t)\) of a largest sunflower with center \(t\),
- for every \(i \in \{0, \ldots, n\}\) the set \(T_i = \{t \in T : n(t) = i\}\).

These information are collected at the beginning of the algorithm in time \(O(n^4)\). Now, assuming that we have these information, thanks to the sets \(T_i\) we can find a triplet \(t\) to which some reduction rules applies in time \(O(n)\). Applying Reduction Rule 6.1 takes time \(O(n)\), since after editing the chosen triplet \(t = \{(a, b), c\}\), we have to update the set \(C\) by considering for each \(d \in L\) the new st-conflicts which may appear in the set \(\{a, b, c, d\}\), and for each \(d \in L\) and each triplet \(t\) with \(|L(t) \cap \{a, b, c, d\}| = 3\) to update the value \(n(t)\) and the set \(T_i\) accordingly. As each application of Reduction Rule 6.1 decrements \(k\), there are at most \(O(kn^4) = O(n^4)\) applications of this rule since \(k = O(n^3)\). Finally, applying Reduction Rule 6.2 takes time \(O(n^4)\), which is the time required to build the set \(L_2(T)\) by traversing \(C\). □

6.1.4 A Subexponential Fixed-Parameter Algorithm

We give a subexponential-time fixed-parameter algorithm for \((k)\)-Dense RTI.

Let \(T\) be a set of triplets and let \(\varphi: L(T) \rightarrow \{1, \ldots, \ell\}\) be a coloring of the labels of \(T\). A triplet \(t = \{(a, b), c\} \in T\) is monochromatic under \(\varphi\) if \(\varphi(a) = \varphi(b) = \varphi(c)\). A peacemaker \(S\) of \(T\) is \(\varphi\)-colorful if no triplet of \(S\) is monochromatic under \(\varphi\). We give an algorithm that finds a \(\varphi\)-colorful peacemaker of \(T\) of minimum size.

Lemma 6.9. Let \(L\) be a set of \(n\) labels, let \(T\) be a minimally dense triplet set over \(L\) and let \(\varphi: L(T) \rightarrow \{1, \ldots, \ell\}\) be a coloring of \(L\). A \(\varphi\)-colorful peacemaker of \(T\) of minimum size can be computed in time \(O(\ell^3(6n)^4)\) and space \(O((2n)^4)\).

Proof. Let \(L_1, \ldots, L_\ell\) be the color classes under \(\varphi\). If there exists a colorful peacemaker of \(T\) then each set \(T\mid L_i\) is consistent with some tree \(T_i\) for \(i = 1, \ldots, \ell\). We have to compute the minimum of \(I(T, T_i)\) for each tree \(T\) over \(L\) containing \(T_i, \ldots, T_\ell\) as subtrees.

A position is a tuple \(\pi = (\pi_1, \ldots, \pi_\ell)\), where each \(\pi_i\) is either a node of \(T_i\) or the special value \(\perp\). For each \(i = 1, \ldots, \ell\), let \(L(\pi_i)\) be the set of labels by which the leaves of the subtree rooted at \(\pi_i\) are labeled, with \(L(\perp) = \emptyset\); let \(L(\pi) = \cup_i L(\pi_i)\). The root position is the position \(\pi_\pi = (r_1, \ldots, r_\ell)\), where \(r_i\) is the root of \(T_i\). A position \(\pi\) is called terminal if for all \(i = 1, \ldots, \ell\), either \(\pi_i\) is a leaf node of \(T_i\) or \(\pi_i = \perp\), and nonterminal otherwise. Define a partial binary relation \(\preceq\) on the set of all positions by

\[
\pi \preceq \pi' :\Leftrightarrow \text{for all } i = 1, \ldots, \ell, \begin{cases} \text{either } \pi_i = \perp, \text{ or in } T_i \text{ there is a path from } \pi_i' \text{ to } \pi_i. \end{cases}
\]
Let $\prec$ be the strict part of $\preceq$. For a position $\pi$, let $I(\pi)$ denote the minimum value of $I(T, T)$ over all trees $T$ over $L(\pi)$ such that $L(T) = L(\pi)$ and $T|L_i = T_i|L(\pi_i)$ for all $i = 1, \ldots, \ell$. Then our goal is to compute $I(\pi)$. Let $\pi$ be a nonterminal position. A decomposition of $\pi$ is an ordered pair $(\pi^1, \pi^2)$ of positions such that for each $i \in \{1, \ldots, \ell\}$,

- if $\pi_i = \bot$ then $\pi_i^1 = \pi_i^2 = \bot$;
- if $\pi_i$ is a leaf node then $\{\pi_i^1, \pi_i^2\} = \{\pi_i, \bot\}$;
- if $\pi_i$ is an internal node with children $v_1, v_2$ then $\{\pi_i^1, \pi_i^2\}$ is either $\{\pi_i, \bot\}$ or $\{v_1, v_2\}$.

A decomposition $(\pi^1, \pi^2)$ of $\pi$ is called proper if both $\pi^1$, $\pi^2$ are distinct from $\pi$; observe that in this case, $\pi^1 \prec \pi$ and $\pi^2 \prec \pi$. For a proper decomposition $(\pi^1, \pi^2)$ of $\pi$, define triplet sets $Q_{\pi, (\pi^1, \pi^2)}$ and $T_{\pi, (\pi^1, \pi^2)}$ by

$$Q_{\pi, (\pi^1, \pi^2)} = \{(u, v), w) \in T \mid \exists h \in \{1, 2\} \exists j_1, j_2, j_3 \text{ such that } \pi_h^{j_i} = u, \pi_h^{j_2} = v, \pi_h^{(h+1 \mod 2)} = w\},$$

$$T_{\pi, (\pi^1, \pi^2)} = \{(x, y), z) \in T \mid \exists (u, v), w) \in Q_{\pi, (\pi^1, \pi^2)} \text{ such that } x \in L(u), y \in L(v), z \in L(w)\}.$$

It follows that $Q_{\pi, (\pi^1, \pi^2)}$ is over a subset of $L(\pi_1) \cup L(\pi_2)$ and that $T_{\pi, (\pi^1, \pi^2)}$ is over $L(\pi)$. Let $I((\pi, (\pi^1, \pi^2)) = |T_{\pi, (\pi^1, \pi^2)} \setminus T| + I(\pi_1) + I(\pi_2)$.

We describe a procedure computing $I(\pi)$ by dynamic programming over values $I(\pi)$, for positions $\pi$ increasing under $\preceq$. Clearly, if $\pi$ is a terminal position then $I(\pi) = 0$. For nonterminal positions $\pi$, the value $I(\pi)$ can be computed thanks to the following claim.

**Claim.** If $\pi$ is nonterminal then $I(\pi)$ is the minimum of $I(\pi, (\pi^1, \pi^2))$ over the proper decompositions $(\pi^1, \pi^2)$ of $\pi$.

**Proof of the claim.** Let $D(\pi)$ be the set of proper decompositions of $\pi$. First observe that for any $(\pi^1, \pi^2) \in D(\pi)$, it holds that

$$T_{\pi, (\pi^1, \pi^2)} = \{(x, y), z) \in T \mid \exists h \in \{1, 2\} : x, y \in L(\pi^h), z \in L(\pi^{(h+1 \mod 2)})\}.$$

Let $T(\pi)$ be the set of trees $T$ such that $L(T) = L(\pi)$ and $T|L_i = T_i|L(\pi_i)$ for all $i = 1, \ldots, \ell$. Further, let

$$M = \min_{T \in T(\pi)} I(T, T), \quad \text{ and } \quad M' = \min_{(\pi^1, \pi^2) \in D(\pi)} I(\pi, (\pi^1, \pi^2)).$$

We have to show that $M = M'$.

For $M \leq M'$, let $(\pi^1, \pi^2) \in D(\pi)$ be such that $I(\pi, (\pi^1, \pi^2)$ is minimum. For $h = 1, 2$, let $S_h \in T(\pi^h)$ be such that $I(T, S_h) = I(\pi^h)$. With $\pi = \bigcup_h L(\pi_i)$ and by definition of a decomposition, it holds that $L(\pi) = L(\pi^1) \cup L(\pi^2)$. We thus consider the tree $S = (S_1, S_2)$ over $L(\pi)$. Then $i(T, S) = i(T, S_1) \cup i(T, S_2) \cup J.$
where $J$ is the set of triplets in $T$ that are not consistent with $S$ and that intersect both $L(\pi^1)$ and $L(\pi^2)$. Thus, by the initial observation, $J = T_{\pi_1(\pi^1,\pi^2)} \setminus T$. It follows that $M \leq I(T,S) = I(\pi^1) + I(\pi^2) + |T_{\pi_1(\pi^1,\pi^2)} \setminus T| = I(\pi_1, \pi^2)) = M'$. 

For $M' \leq M$, let $S \in \mathcal{T}(\pi)$ be such that $I(T,S)$ is minimum. Since $\pi$ is nonterminal, we can write $S = (S_1,S_2)$. For $i = 1, \ldots, \ell$, let $p_i(S)$ be the least common ancestor in $\mathcal{T}_i$ of the leaf labels in $S$, or $\perp$ if $\mathcal{T}_i$ contains no leaf label from $S$. We define positions $\pi^1$ and $\pi^2$ by setting, for $h = 1, 2$ and $i = 1, \ldots, \ell$, the values $\pi^h_i = p_i(S)$. It follows that $(\pi^1, \pi^2)$ is a proper decomposition of $\pi$, and $S^h \in \mathcal{S}(\pi^h)$ for $h = 1, 2$. Besides, $i(T,S) = i(T,S_1) \cup i(T,S_2) \cup J$, where $J = T_{\pi_1(\pi^1,\pi^2)} \setminus T$. We conclude that $M' \leq I(\pi_1, \pi^2)) = I(\pi^1) + I(\pi^2) + |T_{\pi_1(\pi^1,\pi^2)} \setminus T| \leq I(T,S_1) + I(T,S_2) + |T_{\pi_1(\pi^1,\pi^2)} \setminus T| = I(T,S) = M$. 

To complete the proof of the lemma, it remains to justify the time and space requirements. Let $P_i$ be the set of positions with exactly $i$ components which are internal nodes. Clearly, $|P_i| = 2^i \ell^{-i}$. A position $\pi \in P_i$ has $4^i 2^{\ell-i}$ decompositions, and for each decomposition $D$ the value $I(\pi,D)$ can be computed in $O(n^3)$ time.

This time can be reduced to $O(\ell^3)$ by precomputing some quantities, as follows. For nodes $u, v, w$ of trees $T_i, T_j, T_k$, respectively, and $i, j, k$ not all equal, let $n_{\{(u,v),w\}}$ be the number of triplets of the form $(\{y,z\},x)$ or $(\{x,z\},y)$ with $x \in L(u), y \in L(v)$ and $z \in L(w)$. The values $n_{\{(u,v),w\}}$ can be precomputed in $O(\ell^3)$ time at the beginning of the algorithm, and for a given position $\pi$ with decomposition $(\pi^1, \pi^2)$, we then have

$$|T_{\pi_1(\pi^1,\pi^2)} \setminus T| = \sum_{(u,v,\omega) \in Q_{\pi_1(\pi^1,\pi^2)}} n_{\{(u,v),w\}}$$

Thus, the algorithm uses space $O(\sum_{i=0}^\ell \binom{\ell}{i} n^i) = O((2n)^\ell)$, and time $O(\sum_{i=0}^\ell 4^i 2^{\ell-i}(\binom{\ell}{i})^n_i) = O(\ell^3(2n)^\ell)$. 

Hence, to decide an instance $(T,k)$, we need to find proper colorings of an hypothetical peacemaker of $T$ that has size at most $k$. Such colorings can be found either probabilistically—by generating sufficiently many random colorings, or deterministically—by explicitly constructing sufficiently large families of colorings. To this end, we use hypergraphs as modeling tool. Let $H = (V,E)$ be an $r$-uniform hypergraph. Let $\varphi : V \to \{1, \ldots, \ell\}$ be a coloring of $V$. A hyperedge $e \in E$ is monochromatic under $\varphi$ if $\varphi(v) = \varphi(v')$ for all vertices $v, v' \in e$. A coloring $\varphi$ a proper coloring of $H$ if $H$ has no monochromatic edges under $\varphi$.

To probabilistically find a proper coloring of $H$, we would need to generalize a lemma by Alon et al. [18] on random colorings of graphs to random colorings of $r$-uniform hypergraphs. Such a generalization was used by the same authors to give a subexponential fixed-parameter algorithm for BETWEENNESS on dense instances parameterized by the number of constraints to be removed [225].

**Conjecture 6.1.** Let $H = (V,E)$ be an $r$-uniform hypergraph and let $m = |E|$. There exist constants $\alpha, \beta$, such that a uniformly-at-random coloring of $V$ with $(\alpha r m)^{1/r}$ colors is, with probability at least $e^{-(\beta r m)^{1/r}}$, a proper vertex-coloring of $V$. 


As we cannot prove the conjecture, we deterministically show how to find a proper coloring of $H$. For integers $n, m, \ell, r$, a family $F$ of functions from $\{1, \ldots, n\}$ to $\{1, \ldots, \ell\}$ is called a universal $(n, m, \ell, r)$-coloring family if for any $r$-uniform hypergraph $H$ on the set of vertices $\{1, \ldots, n\}$ with at most $m$ edges, there exists an $f \in F$ which is a proper coloring of $H$. The deterministic coloring procedure by Alon et al. [18] can be readily adapted to $r$-uniform hypergraphs, as follows.

The following lemma was announced to appear by Alon et al. [18]; we include its proof for completeness.

**Lemma 6.10** ([225]). There exists an explicit universal $(am^2, m, O(m^{1/r}), r)$-coloring family $F$ of size $|F| \leq 2^{O(m^{1/r})}$.

**Proof.** Let $n = am^2$ and let $\ell = m^{1/r}$. Let $G$ be an explicit family of functions $g$ from $\{1, \ldots, n\}$ to $\{1, \ldots, k\}$ with $r$-wise independent coordinates, that is, for each $i \leq r$ and each $x_1, \ldots, x_i \in \{1, \ldots, n\}$ distinct it holds $\Pr(g(x_1) = v_1, \ldots, g(x_i) = v_i) = \ell^{-i}$. Alon et al. [13] provide a construction of such a family of size roughly $n^{r/2} = m^{O(1)}$.

Now, each function $f \in F$ is described by a function $g \in G$ and a subset $T_g \subset \{1, \ldots, am^2\}$ of size $s = m^{1/r}$ as follows. Suppose that $T_g = \{i_1, \ldots, i_s\}$ with $i_1 < \cdots < i_s$. Define $f$ by

$$f(i) = \begin{cases} m^{1/r} + j, & \text{if } i = i_j \in T, \\ g(i), & \text{otherwise}. \end{cases}$$

We claim that $F$ has the desired properties. First, each function $f \in F$ has range $2m^{1/r}$. Second, the size of $F$ is at most $(\frac{am^2}{m^{1/r}})|G| \leq 2^{O(m^{1/r} \log m)}$. It remains to show that for every $r$-uniform hypergraph $H$ on with vertex set $\{1, \ldots, am^2\}$ and at most $m$ edges, there is an element $f \in F$ which is a proper coloring $H$. Fix such a hypergraph $H$.

**Claim.** If the vertices of $H$ are colored by a uniformly-at-random chosen function $g \in G$ then the expected number of edges in $H$ that are monochromatic under $g$ is $m^{1/r}$.

**Proof of the claim.** By the $r$-wise independence of the coordinates of each function $g \in G$, the probability that some edge is monochromatic under $g$ is exactly $\frac{m^{1/r}}{m}$. Hence, by linearity of expectation, the expected number of edges in $H$ that are monochromatic under $g$ is equal to $m^{1/r}$. \hfill \Box

Let $g \in G$ be a function for which, with positive probability, the number of edges in $H$ that are monochromatic under $g$ is less than $m^{1/r}$. Let $T_g = \{i_1, \ldots, i_s\} \subseteq V(H)$ be a set of vertices containing at least one element of each edge in $H$ that is monochromatic under $g$. Let $f$ be defined by $T_g$ and $g$ and observe that

- edges containing no element of $T_g$ are properly colored by $g$, and thus by $f$;
- edges containing one element of $T_g$ and one element of $\{1, \ldots, am^2\} \setminus T_g$ are properly colored by $f$, since $f$ maps elements of $\{1, \ldots, am^2\} \setminus T_g$ to $\{0, \ldots, m^{1/r} - 1\}$ and maps elements of $T_g$ to $\{m^{1/r}, \ldots, 2m^{1/r} - 1\}$;
• edges included in $T_k$ are properly colored by $f$, since their elements have distinct images by $g$ and thus by $f$. □

By combining the above results, we obtain a subexponential time fixed-parameter algorithms for $(\hat{k})$-Dense RTI.

**Theorem 6.4.** $(k)$-Dense RTI can be solved by a deterministic algorithm using $O(n^4 + 2^{O(k^{1/3} \log k)})$ time and $2^{O(k^{1/3} \log k)}$ space. Moreover, assuming Conjecture 6.1, $(k)$-Dense RTI can be solved by a randomized algorithm in time $O(n^4 + 2^{O(k^{1/3} \log k)})$ time and space $2^{O(k^{1/3} \log k)}$.

**Proof.** We first run the kernelization to build in $O(n^4)$ time a kernel $(T', k')$ with $s = O(k^2)$ labels.

The deterministic algorithm is obtained by constructing a family $\mathcal{F}$ as stated in Lemma 6.10. Then for each function $f \in \mathcal{F}$, we find an $f$-colorful peacemaker of $T'$ of minimum size. We accept $(T', k')$ as a “yes”-instance if some function $f$ yields a peacemaker of size at most $k'$, and reject $(T', k')$ as a “no”-instance otherwise.

The randomized algorithm is obtained by repeating $r$ times the following: (i) choose a random coloring of $L$ with $k$ colors, and (ii) find a minimum colorful peacemaker using Lemma 6.9. We accept $(T', k')$ as a “yes”-instance if one execution finds a colorful peacemaker of size at most $k'$, and reject $(T', k')$ as a “no”-instance otherwise. Conjecture 6.1 implies that we can choose $r = 2^{O(k^{1/3})}$ and $\ell = O(k^{1/3})$.

Both the deterministic and the randomized algorithm applied to the kernel $(T', k')$ have time and space requirements $2^{O(k^{1/3} \log k)}$. □

### 6.2 Bidimensionality in Polynomial Space

Parameterized subexponential time algorithms for NP-hard problems on sparse graphs have thrived enormously in the past decade through the theory of graph separators and the meta-algorithmic theory of bidimensionality created by Demaine, Fomin, Hajiaghayi and Thilikos. This meta-algorithmic theory provides simple criteria for checking whether a parameterized problem is solvable in subexponential time on sparse graphs. The meta-algorithm consists of two steps: For an instance of the problem consisting of a graph $G$ and a positive integer $k$, one shows that the treewidth of the input graph is $o(k)$—sublinear in the parameter, and then uses a fast dynamic programming algorithm over graphs of bounded treewidth to solve the problem. The use of dynamic programming algorithms over graphs of bounded treewidth makes bidimensionality-based algorithms suffer from non-polynomial space complexity. Until now, the only bidimensional problems for which subexponential time, polynomial space algorithms were known are the bidimensional problems admitting a linear kernel. Thus, it is natural to inquire which bidimensional problems that do not have linear kernels still admit parameterized subexponential time algorithms using polynomial space. In this section we show that most bidimensional problems admit polynomial space subexponential time algorithms on sparse graphs. Our result unifies all known results on polynomial space subexponential time algorithms for bidimensional problems, and handles a few problems for which such algorithms were not previously known. Most notably, we provide...
6.2. BIDIMENSIONALITY IN POLYNOMIAL SPACE

the first space-efficient subexponential time parameterized algorithm for the notoriously hard problem of finding a path of length \( k \) on graphs excluding a fixed graph \( H \) as a minor.

In 2000, Alber et al. [7] obtained the first parameterized subexponential algorithm on undirected planar graphs by showing that \((k)\)-Dominating Set is solvable in time \( 2^{O(\sqrt{k})}n^{O(1)} \). This result triggered an extensive study of parameterized problems on planar and more general classes of sparse graphs like graphs of bounded Euler genus, apex minor free graphs and \( H \)-minor free graphs. All this work led to subexponential time algorithms for several fundamental problems like \((k)\)-Feedback Vertex Set, \((k)\)-Edge Dominating Set, \((k)\)-Max Leaf Spanning Tree, \((k)\)-Path, \((k)\)-r-Dominating Set, \((k)\)-Vertex Cover to name a few on planar graphs [7, 8, 11, 54, 69, 101, 115, 144, 168], bounded Euler genus graphs [69, 114], graphs excluding some single-crossing graph as a minor [74], apex-minor-free graphs [68] and \( H \)-minor-free graphs [69, 72, 73]. These algorithms are obtained by showing a combinatorial relation between the parameter and the structure of the input graph and proofs require strong graph theoretic arguments. This graph-theoretic and combinatorial component in the design of subexponential time parameterized algorithms makes it of an independent interest.

Demaine et al. [69] abstracted out the “common theme” among the parameterized subexponential time algorithms on sparse graphs and created the meta-algorithmic theory of Bidimensionality. The bidimensionality theory unifies and improves almost all known previous subexponential algorithms on sparse graphs. The theory is based on algorithmic and combinatorial extensions to various parts of the Graph Minors Theory of Robertson and Seymour [221] and provides a simple criterion for checking whether a parameterized problem is solvable in subexponential time on sparse graphs. The theory applies to graph problems that are bidimensional in that the value of the solution for the problem in question on \( k \times k \) grid or grid-like graph is at least \( \Omega(k^2) \) and the value of solution decreases while contracting (or for some problems deleting) edges. Problems that are bidimensional include \((k)\)-Feedback Vertex Set, \((k)\)-Edge Dominating Set, \((k)\)-Max Leaf Spanning Tree, \((k)\)-Path, \((k)\)-r-Dominating Set, \((k)\)-Vertex Cover and many others.

In most cases we obtain subexponential time algorithms for a problem using bidimensionality theory in following steps. Given an instance \((G,k)\) to a bidimensional problem \( \Pi \), in polynomial time we either decide that it is a “yes”-instance to \( \Pi \) or the treewidth of \( G \) is \( O(\sqrt{k}) \). Or, using the known constant factor approximation algorithm for the treewidth, we find a tree decomposition of width \( O(\sqrt{k}) \) for \( G \) and then solve the problem by dynamic programming over the obtained tree decomposition. This approach combined with Catalan structure based dynamic programming over graphs of bounded treewidth has led to \( 2^{O(\sqrt{k})}n^{O(1)} \)-time algorithms for \((k)\)-Feedback Vertex Set, \((k)\)-Edge Dominating Set, \((k)\)-Max Leaf Spanning Tree, \((k)\)-Path, \((k)\)-r-Dominating Set, \((k)\)-Vertex Cover and many others on planar graphs [69, 70, 80, 114] and in some cases like \((k)\)-Dominating Set and \((k)\)-Path on \( H \)-minor free graphs [69, 79]. We refer to surveys by Demaine and Hajiaghayi [73] and Dorn et al. [79] for further details on bidimensionality and subexponential parameterized algorithms.

The use of dynamic programming algorithms over graphs of bounded treewidth
makes all these subexponential algorithms take not only time but also space non-polynomial in the parameter. Whereas worst-case running times of many subexponential-time algorithms have steadily been decreased, space requirements have mostly been neglected. The question of whether we can reduce the space complexity of these algorithms is not only interesting from the theoretical perspective but also has severe practical relevance. In this regard a quote from Woeginger [249] is very relevant: “algorithms with exponential space complexities are absolutely useless for real life applications”. All this leads to a natural question whether we can obtain subexponential time algorithms for \textit{NP}-hard bidimensional problems on sparse graphs which take polynomial space, that is, can we make the space requirements of algorithms coming from the meta-algorithmic theory of bidimensionality of Demaine et al. [69] polynomial?

In this section we show that this is indeed true and obtain polynomial space subexponential time algorithms for almost all bidimensional problems for which subexponential time algorithms were known. The most notable of our results is the polynomial space subexponential time algorithm for $(k)$-\textsc{Path} on graphs excluding a fixed graph $H$ as a minor. Our result unifies all known results on polynomial space subexponential time algorithms for bidimensional problems and also provides several new results including one for $(k)$-\textsc{Path} and $(k)$-\textsc{Partial Vertex Cover}—even for this problem we do not know whether it admits a polynomial kernel. All our results are based on a non-trivial use of small separators provided by the fact that the non-trivial instances of bidimensional problems have $O(\sqrt{k})$-sized balanced separators if the parameter value is $k$.

Until now only problems for which subexponential parameterized algorithms were known were problems admitting linear kernels. Alber, Fernau and Niedermeier [10] used planar graph separators developed by Lipton and Tarjan [183] and obtained polynomial space subexponential FPT-algorithms for a specific kind of problems which apart from satisfying certain “glueable conditions” admits linear kernels. It has been shown by Bodlaender et al. [34] that the $(k)$-\textsc{Path} problem does not admit a polynomial kernel on planar graphs unless the polynomial hierarchy collapses to its third level, a collapse deemed unlikely to happen.

### 6.2.1 Bidimensionality and Separation Property

Let $H$ be a fixed graph. All instances of parameterized problems $\Pi$ considered in this section are pairs $(G, k)$, where $G$ is a graph excluding $H$ as a minor and $k$ is a positive integer.

**Definition 6.1** ([69, 111]). A parameterized problem $\Pi$ is minor-bidimensional if

1. for any pair of graphs $H \preceq_m G$ and integer $k > 0$, $(G, k) \in \Pi$ implies that $(H, k) \in \Pi$, and
2. there exists a $\delta > 0$ such that for every $(r \times r)$-grid $R$, it holds $(R, k) \notin \Pi$ for every $k \leq \delta r^2$.

In other words, contracting or deleting an edge in a graph $G$ cannot increase the parameter, and the value of the solution on $R$ is at least $\delta r^2$.

A parameterized problem $\Pi$ is contraction-bidimensional if
6.2. BIDIMENSIONALITY IN POLYNOMIAL SPACE

(1) for any pair of graphs \( H \leq_c G \) and integer \( k > 0 \), \( (G,k) \in \Pi \) implies that \( (H,k) \in \Pi \); and

(2) there exists a \( \delta > 0 \) such that \( (\Gamma_r,k) \notin \Pi \) for every \( k \leq \delta^2 \).

In either case, \( \Pi \) is called bidimensional.

**Definition 6.2.** Let \( \Pi \) be a minor-bidimensional (contraction-bidimensional) problem. Call \( \Pi \) a minimization problem if for all \( (G,k) \in \Pi \), \( tw(G) \leq O(\sqrt{k}) \) whenever \( G \) excludes a fixed graph (apex graph) \( H \) as a minor. Call \( \Pi \) a maximization problem if for all \( (G,k) \in \Pi \), \( tw(G) \geq O(\sqrt{k}) \) whenever \( G \) excludes a fixed graph (apex graph) \( H \) as a minor.

Demaine and Hajiaghayi [71] define the *separation* property for problems, and show how separability together with bidimensionality is useful to obtain PTASes on \( H \)-minor-free graphs. In our setting a slightly weaker notion of separability is sufficient. In particular, the following definition is just Requirement 3 of the definition of separability by Demaine and Hajiaghayi [71].

**Definition 6.3.** A minor-bidimensional problem has the separation property if given a graph \( G \), a vertex cut \( S \) of \( G \), and an optimal solution \( OPT \) to \( G \), for any union \( G' \subseteq \{ \emptyset \} \), the value \( |OPT \cap G'| \) is between \( |OPT(G')| - O(|S|) \) and \( |OPT(G')| + O(|S|) \).

For contraction-bidimensional parameters, we have a slightly different definition of the separation property than the one introduced by Fomin et al. [112].

**Definition 6.4.** A contraction-bidimensional problem has the separation property if the following condition is satisfied. Given a graph \( G \), a vertex cut \( S \) whose removal disconnects \( G \) into connected components \( C_1, \ldots, C_k \), and a subset \( I \subseteq \{1,2,\ldots,k\} \), we define \( G_I \) to be the graph obtained from \( G \) by contracting for every \( j \notin I \) the component \( C_j \) into the vertex in \( N(C_j) \) with the lowest index. Let \( OPT \) be an optimal solution to \( G \). Then for any subset \( I \subseteq \{1,2,\ldots,k\} \), the value \( |OPT \cap G_I| \) is between \( |OPT(G_I)| - O(|S|) \) and \( |OPT(G_I)| + O(|S|) \).

A separation of a graph \( G \) is a pair \( (A,B) \) of subsets of \( V(G) \) with \( A \cup B = V(G) \), such that no edge of \( G \) joins a vertex in \( A \setminus B \) to a vertex in \( B \setminus A \). Its order is \( |A \cap B| \). For a function \( w : V(G) \rightarrow \mathbb{R}_+ \) is a function and \( X \subseteq V(G) \), we denote \( \sum_{v \in X} w(v) \) by \( w(X) \). A separation \( (A,B) \) of a graph \( G \) is called balanced if \( w(A \setminus B), w(B \setminus A) \leq \frac{\sqrt{2}}{2} w(V(G)) \) and in this case \( A \cap B \) is called a balanced separator.

**Proposition 6.2 ([19]).** Let \( h \geq 1 \) be an integer, let \( H \) be a simple graph with \( h \) vertices, and let \( G \) be an \( H \)-minor free graph with \( n \) vertices, \( m \) edges, and a weight function \( w : V(G) \rightarrow \mathbb{R}_+ \). Then there is a separation \( (A,B) \) of \( G \) of order \( h^{3/2}n^{1/2} \) such that \( w(A \setminus B), w(B \setminus A) \leq \frac{\sqrt{2}}{2} w(V(G)) \). Besides, such a separation can be found in time \( O(h^{3/2}n^{1/2}(n + m)) \).

**Proposition 6.3 ([218]).** Let a graph \( G \) together with a tree-decomposition of width \( t \). Then for any subset \( W \subseteq V(G) \), in time \( O(|V(G)|) \) one can find a separation \( (A,B) \) of \( G \) of order \( t + 1 \) such that \( |(A \setminus B) \cap W|, |(B \setminus A) \cap W| \leq \frac{2}{3} |W| \).
6.2.2 Bidimensional Problems with Polynomial Kernels

In this subsection we give polynomial space subexponential algorithms for bidimensional problems that do have polynomial kernels, though may be not linear kernels. We classify these problems into two classes based on whether they are separable or not. We start with polynomial space subexponential kernels for separable bidimensional problems and then extend it to other problems. Towards this end, given a problem \( \Pi \) we define a notion of constrained problem, \( C(\Pi) \). An input to \( C(\Pi) \) will consist of a graph \( G \), a positive integer \( k \), and finitely many functions from the vertex/edge set to some finite domain \( \{0, 1, \ldots, d^*\} \). These functions will be used to impose some sort of constraints on the vertices or edges. For an example, suppose \( \Pi \) is the \((k)\)-DOMINATING SET problem. Then given a graph \( G \), by a function \( f : V(G) \to \{0, 1, 2, 3\} \) on the vertex set of \( G \) we could impose the following constraints: a vertex \( v \) such that \( f(v) = 0 \) means it is not yet dominated; a vertex \( v \) such that \( f(v) = 1 \) means it is dominated but could be part of a dominating set; a vertex \( v \) such that \( f(v) = 2 \) means the vertex is forced to be part of the dominating set we are trying to construct; and a vertex \( v \) such that \( f(v) = 3 \) means the vertex is not allowed to be selected in the dominating set we are trying to find. Now we are ready to state the main theorem of the section.

**Theorem 6.5.** Let \( \Pi \) be a minor-bidimensional (contraction-bidimensional) parameterized problem that has a polynomial kernel. Furthermore, assume that there is an algorithm that, given a graph \( G \) together with a tree decomposition \((T, B)\) of \( G \) of width \( w \), solves the problem \( \Pi \) in time \( 2^{O(w \log w)} \cdot n^{O(1)} \) via dynamic programming over \((T, B)\) such that the tables stored at any node have size at most \( 2^{O(w \log w)} \). If \( \Pi \) has the separation property then we can solve \( \Pi \) in time \( 2^{O(\sqrt{d} \log k)} n^{O(1)} \) and polynomial space; otherwise, we can solve \( \Pi \) in time \( 2^{O(\sqrt{d} \log^2 k)} n^{O(1)} \) and polynomial space on graphs excluding a fixed graph (apex graph) \( H \) as a minor.

**Proof.** We first show the result for problems \( \Pi \) that have the separation property. Let \((G, k)\) be an input to problem \( \Pi \) where \( G \) excludes a fixed graph \( H \) as a minor. We first apply a kernelization algorithm by Fomin et al. [112, Theorem 1.1] which in polynomial time either decides that \((G, k)\) is a “yes”- or “no”-instance or obtains an equivalent instance \((G', k')\) such that (a) \((G, k)\) is a “yes”-instance to \( \Pi \) if and only if \((G', k')\) is a “yes”-instance to \( \Pi \); (b) \( G' \) excludes \( H \) as a minor; and (c) \(|V(G')| = O(k)\) (essentially linear kernel). If we get a “yes”/“no”-answer by the procedure then we return the same. Henceforth we assume to have an equivalent instance \((G', k')\) for \( \Pi \) with \( O(k) \) vertices. Since \( G' \) excludes \( H \) as a minor, by Proposition 6.2 we have that \( G' \) has a separation \((A, B)\) of order \( O(\sqrt{d} \log k) \leq d \sqrt{k} \), where \( d \) is a fixed constant, such that \(|A \setminus B|, |B \setminus A| \leq \frac{1}{2} |V(G')| \). Our algorithm essentially finds a balanced separator for the reduced graph \( G' \) and recursively solves a constrained version of the problem on two smaller graphs after “guessing the behavior of an optimum solution on the separator.” To make the presentation clear we prove the result for \((k)\)-DOMINATING SET and \((k)\)-FEEDBACK VERTEX SET. These two problems possess all the difficulties one can encounter with other problems. Hence the ideas used in these problems could be used for any other problem.

For the \((k)\)-DOMINATING SET problem, its constrained version takes \( f : V(G) \to \{0, 1, 2, 3\} \), with the interpretation
6.2. BIDIMENSIONALITY IN POLYNOMIAL SPACE

- \( f(v) = 0 \): vertex \( v \) is not yet dominated,
- \( f(v) = 1 \): vertex \( v \) is dominated but could be in the dominating set,
- \( f(v) = 2 \): vertex \( v \) is part of the dominating set, and
- \( f(v) = 3 \): vertex \( v \) is not part of the dominating set.

Initially, the function \( f \) assigns 0 to every vertex, that is, \( f(v) = 0 \) for every \( v \in V(G) \).

Given a separation \((A, B)\) of \( G' \), let \( C = A \cap B \) and let \( C' = \{ f(v) = 2 \mid v \in C \} \).

Now we guess which vertices among \( \{ v \in C \mid f(v) \in \{0,1\} \} \), say \( C'_{\text{v}} \), are in the dominating set \( D \) that we are looking for. Then for those vertices which are not in the dominating set and are also not dominated by a vertex in \( C_1 = C'_1 \cup C' \), we guess whether they are dominated by a vertex from \( A \setminus B \) or they are dominated by a vertex from \( B \setminus A \). This leads to \( 3^{|C|} \) branchings in this stage. Let \((C_1, C_2, C_3, C_4)\) be a fixed partition of \( C \) such that all vertices of \( C_1 \) are to be the dominating set, all vertices of \( C_2 \) are to be dominated by vertices from \( C_1 \), all vertices of \( C_3 \) are to be dominated by vertices from \( A \setminus B \) and all vertices from \( C_4 \) are to be dominated by vertices from \( B \setminus A \). We define graphs \( G_1 = G((A \setminus B) \cup C_3) \) and \( G_2 = G((B \setminus A) \cup C_4) \) and functions \( f_1 \) and \( f_2 \) as follows. Let \( f_1(v) = f(v) \) for \( v \in (A \setminus B) \setminus N(C) \), \( f_1(v) = 1 \) for \( v \in N(C) \cap (A \setminus B) \) and \( f_1(v) = 3 \) for \( v \in C_3 \).

Similarly, let \( f_2(v) = f(v) \) for \( v \in (B \setminus A) \setminus N(C) \), \( f_2(v) = 1 \) for \( v \in N(C) \cap (B \setminus A) \) and \( f_2(v) = 3 \) for \( v \in C_4 \). Now we find minimum dominating sets for \( G_1 \) and \( G_2 \) recursively satisfying the constraints imposed by the functions \( f_1 \) and \( f_2 \), respectively. Essentially, if \( \text{MDS}(G', f) \) denotes the size of a minimum dominating set of \( G' \) with respect to constraints imposed by the function \( f \) then

\[
\text{MDS}(G', f) = \min_{(C_1, C_2, C_3, C_4)} \left\{ \text{MDS}(G_1, f_1) + \text{MDS}(G_2, f_2) + |C_1| \right\}.
\]

We find the minimum dominating set for \( G' \) using recurrence (6.1), and if the size of the minimum dominating set is at most \( k \) then we return “yes” else we return “no”. This leads to following recurrence for the running time of the algorithm:

\[
T(n') \leq \max_{\frac{1}{2} \leq \beta \leq \frac{3}{2}} \left\{ 3^{n'/\sqrt{\beta}} \left( T(\beta n') + T((1-\beta)n') \right) \right\},
\]

where \( |H| = h \) and \( |V(G')| = n' \). In the base case of the recurrence, when the number of vertices in the graph is a constant, then we solve the constrained problem in constant time. By applying the generalization of the Master Theorem due to Akra and Bazzi [6] to the recurrence above we get that this recurrence solves to \( c^{\sqrt{n}} \), leading to an algorithm with running time \( c^{O(\sqrt{n})} + O(n^2) \) and space polynomial in \( n \).

Next, we show how to solve the \((k)\)-Feedback Vertex Set problem. For the \((k)\)-Feedback Vertex Set problem, let \( f : V(G) \to \{0,1,2,\ldots,d \sqrt{k}\} \) with the interpretation

- \( f(v) = 0 \): vertex \( v \) could be part of the feedback vertex set,
• \( f(v) = 1 \): vertex \( v \) is part of the feedback vertex set,

• \( f(v) = i \) for \( 2 \leq i \leq d\sqrt{k} \): vertex \( v \) is not part of the feedback vertex set, and furthermore all vertices \( v' \) with \( f(v') = i \) should be part of the same tree as \( v \) of the forest obtained from deleting the vertices of feedback vertex set.

Initially, the function \( f \) assigns 0 to every vertex, that is, \( f(v) = 0 \) for every \( v \in V(G) \).

Given a separation \((A, B)\) of \( G' \), let \( C = A \cap B \). Now we guess which vertices among \( \{ v \in C \mid f(v) = 0 \} \), say \( C_1 \), are in the feedback vertex set \( F \) that we are looking for. Next, we guess two subsets \( X_1 \subseteq C \setminus C_1 \) and \( X_2 \subseteq C \setminus C_1 \) such that \( X_1 \cup X_2 = C \setminus C_1 \). Now let \( L_1 = \{ v \in A \setminus B \mid f(v) \geq 2 \} \) and \( L_2 = \{ v \in B \setminus A \mid f(v) \geq 2 \} \). Furthermore, let \( P_1 \) and \( P_2 \) be the set of partitions of \( X_1 \cup L_1 \) and \( X_2 \cup L_2 \), respectively. We will use the partition to enforce which vertices will be in the same component of the forest after we have deleted the vertices from the feedback vertex set. A pair of partitions \( P_1 = (Y_1, Y_2, \ldots, Y_t) \) and \( P_2 = (Y'_1, Y'_2, \ldots, Y'_t) \) are called compatible if for any \( 1 \leq i \leq \ell \) and \( 1 \leq j \leq \ell' \) we have that \( |Y_i \cap Y'_j| \leq 1 \). The compatibility condition ensures that the union of forests from two parts of the graph remains a forest. Given a pair of compatible partitions \( P_1 = (Y_1, Y_2, \ldots, Y_t) \in P_1 \) and \( P_2 = (Y'_1, Y'_2, \ldots, Y'_t) \in P_2 \), we assign new functions \( f_1 \) and \( f_2 \) as follows. Let \( f_1(v) = f(v) \) for all \( v \in (A \setminus B) \setminus L_1 \), and \( f_1(v) = i \) for \( v \in Y_i \), \( 1 \leq i \leq \ell \). Similarly, let \( f_2(v) = f(v) \) for all \( v \in (B \setminus A) \setminus L_2 \), and \( f_2(v) = i \) for \( v \in Y'_i \), \( 1 \leq i \leq \ell' \).

We define graphs \( G_1 = G[(A \setminus B) \cup X_1] \) and \( G_2 = G[(B \setminus A) \cup X_2] \). Now we find minimum feedback vertex sets for \( G_1 \) and \( G_2 \), recursively satisfying the constraints imposed by functions \( f_1 \) and \( f_2 \), respectively. Essentially, if \( MFVS(G', f) \) denotes the size of a minimum feedback vertex set of \( G' \) with respect to constraints imposed by the function \( f \) then

\[
MFVS(G', f) = \min_{(C_1, X_1, X_2) \in P_1 \times P_2} \left\{ MFVS(G_1, f_1) + MFVS(G_2, f_2) + |C_1| \right\}.
\]

We find the minimum feedback vertex set for \( G' \) using recurrence (6.2), and if the size of the minimum feedback vertex set is at most \( k \) then we return "yes" else we return "no". To establish the running time of the recursive procedure we first bound the size of \( |L_j \cup X_j|, j \in \{1, 2\} \) possible at any stage. A vertex gets assigned to a non-zero value by the function \( f \) if and only if it is part of some separator while branching. Observe that the upper bound on the size of \( |L_j \cup X_j| \) is governed by the following recurrence: \( s(k^*) \leq s(k^*/c) + O(\sqrt{k^*}) \), where \( k^* \) is the size of the current graph and \( c \) is a constant. Hence, on any one branch of recursive calls it holds that \( |L_j \cup X_j| \leq t \sqrt{k} \). This leads to the following recurrence for the running time of the algorithm:

\[
T(n') \leq \max_{\frac{1}{2} \leq \beta \leq \frac{3}{2}} \left\{ 3h^{3/2} \sqrt{\pi} (\sqrt{\pi})^{(\ell' \sqrt{k})} \left( T(\beta n') + T((1 - \beta)n') \right) \right\},
\]

where \( |H| = h \) and \( |V(G')| = n' \). In the base case, when the number of vertices in the graph is constant, then we solve the constrained problem in constant time. Now
by applying the generalization of the Master Theorem due to Akra and Bazzi [6] to the recurrence above we get that it solves to $c^{O(\sqrt{\log k})}$, leading to an algorithm with running time $c^{O(\sqrt{\log k})} + O(n^2)$ and space polynomial in $n$.

Essentially, our recursive algorithm finds a balanced separator and then guesses the "necessary information" required to decompose the problem into two parts and then solve the two independent problems recursively. The guessing part is best viewed as branching on rows of a table stored in a dynamic programming algorithm for the problem $\Pi$ on a bag of a given tree-decomposition. Here, the bag consists of all the vertices ever being part of some separator on this path of recursion tree. The fact that the size of a separator is at most $O(\sqrt{k})$, where $O(k')$ is the size of the current graph, together with the observation that in every recursive step the size of graph drops by a fraction, we have that the bag size is always bounded by $O(\sqrt{k})$. All this together implies the statement of the theorem for bidimensional problems that have separation property.

Next, we assume that $\Pi$ is a problem which does not satisfy the separation property but has a polynomial kernel. We show the case when $\Pi$ is a minimization problem. The proof for the case when $\Pi$ is a maximization problem is analogous. Let $(G,k)$ be an input to $\Pi$, where $G$ is a graph excluding a fixed graph $H$ as a minor. We first apply the kernelization algorithm on $(G,k)$ to obtain an equivalent instance $(G',k')$ such that (a) $(G,k)$ is a "yes"-instance to $\Pi$ if and only if $(G',k')$ is a "yes"-instance to $\Pi$; (b) $G'$ excludes $H$ as a minor; and (c) $|V(G')| = k'$, where $c$ is a fixed constant. We start by computing a tree-decomposition of $G'$ of width $t$ using the constant-factor approximation algorithm of Demaine et al. [70] for computing the treewidth of a $H$-minor free graph. Let $a$ be the approximation factor. Since $\Pi$ is a bidimensional minimization problem, we know that if $G$ is a "yes"-instance then the treewidth of $G'$ is upper bounded by some $\delta \sqrt{k}$. So if $t > a \delta \sqrt{k}$ then $G'$ is a "no" instance for $\Pi$. Hence, we assume that $t \leq a \delta \sqrt{k}$. Since $G'$ has treewidth $t$, using Proposition 6.3 we can find a balanced separation $(A,B)$ of $G'$ such that $|C = A \cap B| \leq t + 1$. So for our algorithm we recurse as we did for separable parameterized bidimensional problems but on a separator of size at most $t + 1$. Once we reach a stage where $G''$, the graph in the current recursive step, has at most $O(k)$ vertices then we start finding separation using Proposition 6.2 on the whole graph $G''$ and then recurse on this separator. From here onwards, we do as we did for the case when the problem had a linear sized kernel. The only thing we need to note is that the set of vertices which appears in some separator on this path of the recursion tree could have size $O(\sqrt{k})$, as the size of the separator may not decrease until we reach the stage when the graph in the recursive step has $O(k)$ vertices. This implies that if $\Pi$ has a dynamic programming algorithm over graphs of bounded treewidth such that each table stored at any node has size at most $2^{O(|w| \log w)}$, then we can solve $\Pi$ in time $2^{O(\sqrt{\log^2 k})} n^{O(1)}$ and polynomial space on graphs excluding a fixed graph $H$ as a minor. This completes the proof of the theorem.

We get the following corollary to Theorem 6.5.

**Corollary 6.2.** Let $H$ be a graph. Then

-
• (k)-Edge Dominating Set, (k)-Vertex Cover, (k)-Minimum Maximal Matching and (k)-Independent Set have algorithms with running time \(2^{O(\sqrt{kn})}n^{O(1)}\) and polynomial space on graphs excluding \(H\) as a minor,

• (k)-Connected Vertex Cover, (k)-Cycle Packing and (k)-Feedback Vertex Set have algorithms with running time \(2^{O(\sqrt{kn}\log n)}n^{O(1)}\) and polynomial space on graphs excluding \(H\) as a minor, and

• (k)-Independent Dominating Set, (k)-Minimum Leaf Out-Branching and (k)-Connected Feedback Vertex Set have algorithms with running time \(2^{O(\sqrt{kn}\log n)}n^{O(1)}\) and polynomial space on graphs of bounded Euler genus.

Provided that \(H\) is an apex graph,

• (k)-Dominating Set, (k)-r-Dominating Set and (k)-q-Threshold Dominating Set have algorithms with running time \(2^{O(\sqrt{n})}n^{O(1)}\) and polynomial space on graphs excluding \(H\) as a minor, and

• (k)-Connected Dominating Set and (k)-Full-Degree Spanning Tree have algorithms with running time \(2^{O(\sqrt{kn}\log n)}n^{O(1)}\) and polynomial space on graphs excluding \(H\) as a minor.

The desired kernel bounds for respective problems follow from results by Bodlaender et al. [36] and Fomin et al. [112].

For the (k)-Connected Dominating Set and (k)-r-Dominating Set problems, our subexponential parameterized algorithms extend to the non-minor closed class of map graphs. Again, the idea is to go from the map graph \(G_M\) to some witness \(H\) of \(G_M\). The translation for (k)-Connected Dominating Set is the same as in Lemma 3.13. For (k)-r-Dominating Set, it is not difficult to observe [69] that \(G_M\) has a distance-\(r\) dominating set of size at most \(k\) if and only if \(H\) has a distance-2r dominating set of size \(k\).

**Theorem 6.6.** (k)-Connected Dominating Set and (k)-r-Dominating Set have polynomial-space algorithms with running times \(2^{O(\sqrt{kn}\log n)}n^{O(1)}\) and \(2^{O(\sqrt{n}\log n)}n^{O(1)}\), respectively, on map graphs.

Our result for (k)-r-Dominating Set on map graphs improves Demaine et al.’s [69] algorithm with running time \(2^{O(\sqrt{n}(r+\log r))n^{O(1)}}\) and superpolynomial space requirements.

### 6.2.3 Bidimensional Problems Unlikely to Have Polynomial Kernels

In the last subsection we gave an algorithm for problems that are known to have polynomial kernels. Essentially, the idea was to branch on small sized separators and to obtain an instance of a constrained version of the problem where the size of the current input is fraction smaller than the original input. This kind of approach can not be used to obtain parameterized algorithms for problems for which no polynomial kernels are known.
In this section we exhibit our approach on the \((k)\)-Path problem for which no polynomial kernels exists even on planar graphs unless the polynomial hierarchy collapses to the third level. It is known \([128],[104],\) Lemma 11.16 that in a graph \(G\) of treewidth at most \(t\), every subset \(W \subseteq V(G)\) has a balanced \(t + 1\)-separator. Our result is based on an algorithmic “generalization” of this result. We essentially prove that given a graph \(G\) on \(n\) vertices of treewidth at most \(t\), we can find a family \(\mathcal{F}\), \(|\mathcal{F}| \leq O(n)\), of subsets of \(V(G)\) of size at most \(t + 1\), such that for any \(W \subseteq V(G)\) there exists a balanced \(t + 1\)-separator in \(\mathcal{F}\). Towards this end we show the following lemma.

**Lemma 6.11.** Let \(G\) be a graph, let \((T, \mathcal{B})\) be a nice tree-decomposition of \(G\) of width \(t\) and let \(S \subseteq V\) be a vertex set of size \(k\). Then there exists a bag \(B \in \mathcal{B}\) and disjoint vertex sets \(X, Y \subseteq V(G)\), possibly depending on \(B\), such that \(V(G) = X \cup B \cup Y\) and there is no edge from any vertex in \(X\) to any vertex in \(Y\). Moreover, \(\frac{k'}{2} \leq |X \cap S| \leq |Y \cap S| \leq \frac{3k'}{2}\) for \(k' = k - |B \cap S|\).

**Proof.** Let the tree \(T\) be rooted at an arbitrary vertex \(r\). Let \(\text{Cur}\) denote the current vertex of \(T\) we are working on. Initially we set \(\text{Cur} := r\). Let \(X_r\) be the bag corresponding to \(r\). Let \(T_1, T_2\) and \(T_3\) be the subtrees obtained after deleting \(r\) from \(T\). It is possible that we get only two subtrees \(T_1\) and \(T_2\) if the degree of \(r\) is 2 in \(T\). We show how to proceed when we have three subtrees; the proof for two subtrees is similar. Let \(A_i = (\bigcup_{i \in V(T)} \bigcup_{X_i} X_{r_i}, i \in \{1, 2, 3\}\). Suppose first that \(|A_i \cap S| < k'/2\) for all \(i \in \{1, 2, 3\}\). Observe that in this case, it must hold \(|A_i \cap S| > k'/3\) for some \(i \in \{1, 2, 3\}\). Assume, without loss of generality, that \(|A_3 \cap S| > k'/3\). It follows that \(k'/2 \leq |A_1| + |A_2| \leq k - |A_3| \leq 2k'/3\). Thus, the sets \(X = A_1 \cup A_2\) and \(Y = A_3\) yield the desired partition of \(V(G)\) into \(\{X, B, Y\}\).

Suppose now that \(|A_i \cap S| > k'/2\) for some \(i \in \{1, 2, 3\}\). Let \(z\) be the neighbor of \(r\) in \(T_r\), then we assign \(\text{Cur} := z\) and proceed as in the previous paragraph. After at most \(|V(T)|\) steps, we will end up with a bag \(B\) for which \(|A_i \cap S| < k'/2\) for all \(i \in \{1, 2, 3\}\) and on which first part of the proof will apply. 

Observe that by Lemma 6.11, given a graph \(G\) together with a nice tree-decomposition \((T, \mathcal{B})\) of \(G\) of width at most \(t\), we can choose the family \(\mathcal{F}\) to be the bags of \((T, \mathcal{B})\) such that for any \(W \subseteq V(G)\) there exists a balanced \(t + 1\)-separator in \(\mathcal{F}\). Clearly, \(|\mathcal{F}| \leq O(n)\) and any set in the family \(\mathcal{F}\) has size at most \(t + 1\).

Suppose that \(G\) is a graph containing a path \(P\) of length \(k\) and let \(S\) be the set of vertices on \(P\). Then the balanced separator with respect to set \(S\), provided by Lemma 6.11, allows to search for a \(k\)-path in \(G\) by a divide-and-conquer approach. The difficulty, however, is that we do not know \(S\) a priori. To overcome this difficulty, we loop over every separator in the family \(\mathcal{F}\). We are ready to prove the main theorem of this section.

**Theorem 6.7.** Let \(G\) be a graph on \(n\) vertices excluding a fixed graph \(H\) as a minor and let \(k \geq 0\) be an integer. Then in time \(2^{O(\sqrt{k \log^2 k})} \cdot n^{O(1)}\) and polynomial space we can decide whether \(G\) contains a path of length \(k\).

**Proof.** We give an algorithm that solves the problem in a recursive manner. The algorithm starts by computing a tree-decomposition of \(G\) of width \(t\), using the constant-factor approximation algorithm of Demaine et al. \([70]\) for computing the
treewidth of a $H$-minor free graph. Let $a$ be the approximation factor. We show that if $t > d\sqrt{k}$ for some constant $d$ then $G$ has a path of length $k$. Towards this end, we utilize the result of Demaine and Hajiaghayi [73] that for any $H$-minor-free graph $G$ with treewidth more than $r$ there is a constant $\delta > 1$ only dependent on $H$ such that $G$ has a $r/\delta \times r/\delta$-grid minor. Hence, since $(k)$-Path is a minor-closed parameter, if $t > a\delta\sqrt{k}$ then $tw(G) > \delta\sqrt{k}$. In this case $G$ contains a $\sqrt{k} \times \sqrt{k}$-grid minor which accommodates a path of length $k$ we are looking for.

For the rest of the proof we will assume that $t \leq a\delta\sqrt{k}$. Given a tree-decomposition of $G$ of width $t$ we can obtain a nice tree-decomposition of same width in linear time. This allows us to assume that we are given a nice tree-decomposition $(T,B)$ of $G$ of width $t$. Our goal is to partition $G$ into graphs $G_1, G_2$ and partition $k$ into “balanced” integers $k_1, k_2 \geq 0$ and then to solve the problem independently for $(G_1, k_1)$ and $(G_2, k_2)$.

In recursive steps our algorithm has as input a graph $G$, a positive integer $k$ and a set $P = \{(x,y) \mid x, y \in V(G)\}$ of vertex pairs on $V(G)$. Let $V(P)$ be the set of vertices appearing in some pair of $P$. In recursive steps the algorithm either correctly determines whether $G$ has a path of length $k$, or it branches into recursive steps, or it outputs a set of internally vertex-disjoint paths between the pairs in $P$, such that lengths of these paths sum up to at least $k$ and the internal path vertices do not belong to $V(P)$. We call such set of paths a path-system on $P$, and its length is the sum of the lengths of the paths contained in it. Initially, $P = \{\{x,y\}\}$, where $x$ and $y$ are the guessed end vertices of the path of length $k$ in $G$ that we are looking for.

Suppose that graph $G$ has a path $P$ of length $k$ and let $S$ be the set of vertices on $P$. By applying Lemma 6.11 to the set $S$, we know that there exists a bag $B \in B$ such that $V(G) = X \cup B \cup Y$, there is no edge from any vertex in $X$ to any vertex in $Y$ and $\frac{k}{2} \leq |X \cap S| \leq |Y \cap S| \leq \frac{2k'}{3}$, where $k' = k - |B \cap S|$. First, we guess a bag $B$ satisfying the desired properties. Then we guess the following things in the order of their appearance:

1. Guess which vertices from $C \subseteq (B \setminus V(P))$ are on a path system of $P$.
2. Guess two subsets $A_1 \subseteq C \cup (V(P) \cap B)$ and $A_2 \subseteq C \cup V(P)$ such that $A_1 \cup A_2 = C \cup V(P)$.
3. Guess a partition of $k' = k - |C \cup (V(P) \cap B)|$ into $k'_1, k'_2$ such that $\frac{k'}{2} \leq k'_1, k'_2 \leq \frac{2k'}{3}$.
4. Let $L_1 = V(P) \cap X$ and $L_2 = V(P) \cap Y$. Furthermore, for $i = 1, 2$ let $I_i$ be the set of all partitions of $A_i \cup L_i$ such that each part of a partition is a pair of $A_i \cup L_i$. Every partition in $I_i$ is called a set of pairings. We guess a pair of partitions $P_i, P'_i$ such that they are compatible. Recall that a pair of partitions $P_i = (Y_1, Y_2, \ldots, Y_{\ell})$ and $P'_i = (Y'_1, Y'_2, \ldots, Y'_{\ell'})$ are compatible if for any $1 \leq i \leq \ell$ and $1 \leq j \leq \ell'$ we have that $|Y_i \cap Y'_j| \leq 1$.

Let $G_1 = G[X \cup A_1], G_2 = G[Y \cup A_2]$ and $k_i = k'_i + |A_i|$ for $i = 1, 2$. The recursive step of the algorithm is now as follows. Given tuples $(G_1, P_1, k_1)$ and $(G_2, P_2, k_2)$,
we try to find path-systems on \( P_1 \) in \( G_1 \) of length \( k_1 \) and path-systems on \( P_2 \) in \( G_2 \) of length \( k_2 \). The recursion stops when \( k_1, k_2 = \Theta(\sqrt{\epsilon}) \).

Let us analyze the running time of this algorithm. We first bound the size of \( |L_j \cup A_j| \) for \( j \in \{1, 2\} \). A vertex belongs to \( L_j \) if and only if it is part of some balanced separator while branching. Observe that, since the tree decomposition \((T,B)\) has width \( t \leq a\sqrt{\epsilon} \), the size of a balanced separator is at most \( a\delta \sqrt{\epsilon} + 1 \) throughout the recursion. Our recursive procedure stops when \( k_1, k_2 = \Theta(\sqrt{\epsilon}) \). This implies that the length of any path in the recursion tree is \( O(\log k) \) and hence, on any one branch of recursive calls it holds that \( |L_j \cup A_j| \leq O(\sqrt{\epsilon} \log k) \). In summary, we obtain the following recurrence for the running time of the algorithm:

\[
T(t, k) \leq n \cdot \max_{\frac{1}{2} \leq \beta \leq \frac{3}{2}} \left\{ c \sqrt{\epsilon} \log t \left( T(t, \beta k) + T(t, (1 - \beta)k) \right) \right\}, \tag{6.3}
\]

where \( t = a\delta \sqrt{\epsilon} \). Now by applying the generalization of the Master Theorem due to Akra and Bazzi [6] to recurrence (6.3) we get that it solves to \( n^{O(\log k)} e^{O(\sqrt{\epsilon} \log^2 k)} \), leading to an algorithm with running time \( n^{O(\log k)} e^{O(\sqrt{\epsilon} \log^2 k)} \) and space polynomial in \( n \).

What remains to be discussed is the base case of the recursion. Unlike in Theorem 6.5, we cannot use any brute force algorithms in the base case, because we cannot assume that \( k_1, k_2 = O(1) \). The reason for this anomaly is that we can not guarantee that the treewidth of the graph decreases in recursive steps, even though the parameter \( k \) decreases. So when \( k_1, k_2 = \Theta(\sqrt{\epsilon}) \) we can not guarantee that there exists a balanced separator that decreases \( k \) to a fraction \( c_i k_i \) for some \( c_i \in (0, 1) \) and \( i = 1, 2 \). Therefore, we use an entirely different algorithm when \( k_1, k_2 = \Theta(\sqrt{\epsilon}) \).

That algorithm runs in time \( e^{\sqrt{\epsilon} \log k} n^{O(1)} \) and space polynomial in \( n \), and works as follows.

Given a partition \( P \) and a positive integer \( k_1 \), for every pair \( p \in P \) we guess a number \( k_p \), denoting the size of path \( p \) in the path system on \( P \) we are looking for, such that \( \sum_{p \in P} (k_p - 1) = k_1 \). Let \( P_p \) denote a path of length \( k_p \) where the endpoints of the path are assigned vertices from the pair \( p \in P \). Now we form the graph \( F \) that is the disjoint union of \( P_p \), that is, \( F = \cup_{p \in P} P_p \). Then we use the algorithm of Amini et al. [22, Theorem 12] that combines inclusion-exclusion with color-coding to solve instances of the subgraph isomorphism problem. More precisely, given graphs \( F \) of treewidth \( t \) and a graph \( G \) on \( n \) vertices, the algorithm by Amini et al. [22] in time \( (2e)^{|F|+o(|F|)} n^{O(1)} \) and space polynomial in \( n \) determines whether in \( G \) there is a subgraph isomorphic to \( F \). Observe that \( F = \cup_{p \in P} P_p \) is a graph of treewidth 1 and has size at most \( O(\sqrt{\epsilon}) \). The only problem is that some of the vertices in \( F \) are fixed. To overcome this difficulty, we apply color coding in the following way: every vertex in \( V(P) \) is made a color class in itself, which contains only that particular vertex. This completes the description of our algorithm for the \((k)\)-\textsc{Path}, which runs in time \( n^{O(\log k)} e^{O(\sqrt{\epsilon} \log^2 k)} \) and space polynomial in \( n \).

An algorithm with running time \( n^{O(\log k)} e^{O(\sqrt{\epsilon} \log^2 k)} \) does not even imply that the \((k)\)-\textsc{Path} problem is fixed parameter tractable. So to complete the proof of the theorem we do as follows. Dorn et al. [79] gave an algorithm that for a graph \( G \)
excluding a fixed graph \( H \) as a minor and integer \( k \) decides in time and space \( 2^{O(\sqrt{k})}n^{O(1)} \) whether \( G \) has a path of length \( k \). Thus, if \( n > 2^{O(\sqrt{k})} \) then this algorithm runs in time and space \( n^{O(1)} \), which gives the desired running time. Whereas if \( n \leq 2^{O(\sqrt{k})} \) then our polynomial-space algorithm presented in the last paragraph with running time \( n^{O(\log k)}\sqrt{\sqrt{k}} \) actually runs in time \( 2^{O(\sqrt{k} \log^{2} k)} \).

We remark that the algorithm presented in Theorem 6.7 for (k)-Path can be made to run in time \( 2^{O(\sqrt{k} \log k)}n^{O(1)} \) and space polynomial in \( n \) on planar graphs using sphere cut decompositions introduced by Dom et al. [80].

Another problem, though not bidimensional, which can be handled in the same way as (k)-Path is (k)-Partial Vertex Cover. The problem is not known to admit a polynomial kernel, even on planar graphs. Here we are given a graph \( G \), positive integers \( k \) and \( t \) and we look for a subset \( S \subseteq V(G) \) such that \( |S| \leq k \) and the number of edges incident to \( S \) is at least \( t \). This problem was studied by Amini et al. [21] and Fomin et al. [111] by taking \( k \) as a parameter, and an algorithm with time and space \( 2^{O(\sqrt{k})}n^{O(1)} \) was obtained. Let \( G \) be a graph that excludes a fixed apex graph \( H \) as a minor. Then the algorithm by Fomin et al. [111] in polynomial time either finds a solution for an instance \((G, k, t)\) or obtains an equivalent instance \((G', k, t)\) such that \( tw(G') \leq O(\sqrt{k}) \). We obtain this equivalent instance and then we follow the same steps on \((G', k, t)\) as we did for the algorithm for (k)-Path. The only thing we need to do is to find an alternate algorithm that runs in polynomial space so that we can use it when the size of parameter is roughly \( O(\sqrt{k}) \). At this stage we can use the branching algorithm described by Amini et al. [21, Theorem 8] which will run in time \( 2^{O(\sqrt{k} \log k)}n^{O(1)} \) if the parameter is \( O(\sqrt{k}) \). We summarize the analysis for (k)-Partial Vertex Cover in the following theorem.

**Theorem 6.8.** (k)-Partial Vertex Cover can be solved in time \( 2^{O(\sqrt{k} \log k)}n^{O(1)} \) and polynomial space on graphs excluding an apex graph as a minor.

### 6.3 Concluding Remarks

In this chapter we gave subexponential FPT-algorithms for two kinds of instances, dense instance and sparse instances.

For dense instances, the details of chromatic coding are still highly problem-tailored. A first step to make the technique a fully versatile one would be a proof of the hypergraph coloring conjecture, that is, Conjecture 6.1. Another intriguing field to investigate for problems with dense input are the connections between subexponential FPT-algorithms, linear kernels and approximations. Such close connections have recently been established for sparse graphs, through the theory of Bidimensionality [72, 112]. For the (k)-Feedback Arc Set problem in tournaments, and the Betweenness problem on dense instances parameterized by the number of constraints to be removed, all three components—subexponential FPT-algorithms, linear kernels [30, 252] and PTASs [5, 169, 170]—are known. But for (k)-Minimum Triplet Inconsistency, in this chapter we only obtained a quadratic kernel and the currently best approximation factor is \( n \) [47]. One way to obtain a linear kernel
seems to be through a constant-factor approximation, and we conjecture both these components to exist.

Further, we conjecture that the running time of our algorithm can be improved to $2^{O(k^{1/3})}$, by analogy to a recent $2^{O(k^{1/2})}$-time algorithm [93] for the \((k)\)-feedback Arc Set problem in tournaments.

For sparse instances, we obtained fairly general polynomial-space subexponential FPT-algorithms for a large class of bidimensional problems on graphs excluding a fixed graph or apex graph $H$ as a minor. The most notable among our results is the polynomial-space subexponential FPT-algorithm for \((k)\)-Path on graphs excluding $H$ as a minor. Here, an interesting problem which remains open is whether \((k)\)-Path can be solved in time $2^{O(\sqrt{k})^{O(1)}}$ and space polynomial in $n$, even on planar graphs. Such a running time would be optimal under the Exponential Time Hypothesis [66].
We give an outlook on future research related to the results presented in this thesis. The results in this thesis raise many open questions that are worth to be answered. Apart from the concrete questions presented at the end of each chapter, let us raise a few more programmatic directions.

Connections to other Algorithmic Paradigms

Several parameterized algorithms and moderately exponential-time algorithms are based on deep results from other algorithmic approaches. However, the connections often seem problem-specific. Deeper links to approximation algorithms, enumeration and other algorithmic paradigms potentially lead to many new fast algorithms, and probably even reveal hidden relations of intractable problems with respect to their time and space complexities. For example, studying the parameterized complexity of enumeration problems and counting problems might lead to such results, or further investigations of the “hybrid” algorithms by Williams [246] which either return a good approximate solution or an optimum solution quickly. Besides, extending the many available polynomial time approximation schemes (PTASes) to efficient PTASes—fixed-parameter algorithms with parameter the reciprocal of the approximation factor—could be a possible route for feasible algorithms for intractable problems. A step further is to investigate which efficient PTASes admit kernels of polynomial size. For that, the “structure approximation” by Hamilton et al. [149] could be potentially useful.

Connections between Parameterized Complexity and Classical Complexity

Parameterized Complexity brings its own set of complexity classes, hierarchies and hardness concepts. Until now, only few ties to classical complexity and inapproximability have been established. This applies for instance to linking the Exponential-Time Hypothesis to the equivalence of FPT and M[1], or to lower bounds on running times of fixed-parameter algorithms. Recently, lower bounds on kernel sizes were linked to the conjecture that coNP ≠ NP/poly. Many more connections we expect to surface. For instance, restricted definitions of kernels might allow for stronger kernel lower bounds based on assumptions about the distinctness of classical complexity classes. Or we want to link the power of randomized fixed-parameter algorithms to conjectures about randomized complexity classes.
Algorithm Engineering

A major motivation for investigating intractable problems in the realm of parameterized complexity is to better understand when large instances of those problems can be solved fast in practice. It is by now the case that fixed-parameter tractability and kernels have been established for many parameterized problems, yet those algorithms are still awaiting their application to real-life problems. With the prominence of Parameterized Complexity growing beyond theoretical computer science communities, more attention on turning theoretically interesting into practically feasible results will receive more devotion. Algorithm engineering, and concerns about space demands, will play a central role in that respect.

Multivariate Complexity Analysis and Algorithmics

The computational complexity analysis within the classical $P$ vs. $NP$-dichotomy is now often refined by identifying structural parameters and analyzing them by means of the bivariate parameterized complexity theory. Also, by measuring the running time of algorithms in more than one parameter, practitioners may be offered a larger choice of different algorithms better suiting their needs. Yet, it may very well turn out that two dimensions are not the end of the line, but that a whole multi-dimensional framework of problem analysis unfolds. One step in this direction is to understand what the “good” class of functions, and the “bad” class of functions, in such a framework would be.

New Fields of Application

Parameterized Complexity has successfully been applied to a great variety of problems from social choice theory, computational biology, constraint satisfaction, VLSI design, and many other areas of application, by either providing fast algorithms or ruling out the existence of such by appropriate hardness proofs. Very likely, more fields of applications will employ the toolbox of fixed-parameter algorithms to solve important problems optimally. In particular, theoretical physics and economics seem to offer exciting opportunities to apply existing techniques, but also to discover new tricks and theories.

We close with two invitations to concrete challenges for the reader:

1. Find a “good” analogue of treewidth in directed graphs. In undirected graphs, treewidth is a central notion for establishing fixed-parameter tractability of problems, and tree decompositions allow to design general and simple algorithms. Despite many efforts, such a notion is still lacking for directed graphs.

2. Develop a theory of fixed-parameter approximation and parameterized inapproximability. Recently, the first results on fixed-parameter approximation algorithms have appeared, as well as certain inapproximability results. Yet at this moment, we lack an understanding which properties of parameterized problems are essential for those algorithms, and which properties prohibit not only fixed-parameter tractability but even fixed-parameter approximation.
Program for Second Moment Evaluation

We give the Python program by which we compute the values $E[X_lX_{l'}]$ in Table 5.2.

```python
def between():
    # six variables 0, 1, 2, 3, 4, 5
    # two clauses (0,{1,2}) and (3,{4,5})
    # charvec indicates which of these variables
    # are same/distinct
    charvec = []
    charvec.append([0, 1, 2, 0, 3, 4])
    charvec.append([0, 1, 2, 3, 1, 4])
    charvec.append([0, 1, 2, 3, 0, 4])
    charvec.append([0, 1, 2, 0, 1, 3])
    charvec.append([0, 1, 2, 3, 1, 2])
    charvec.append([0, 1, 2, 3, 0, 1])
    charvec.append([0, 1, 2, 1, 0, 3])
    charvec.append([0, 1, 2, 1, 0, 2])
    for i in range(0, 8):
        print charvec[i]
        computeExp(charvec[i])

def computeExp(charvec):
    from fractions import Fraction
    X_p = [Fraction(0), Fraction(-1, 3), Fraction(1, 6),
           Fraction(2, 3), Fraction(-1, 3)]

    # phi takes one of these values
    num0123 = range(0, 4)

    # len(set(charvec)) = number of distinct variables
    length = len(set(charvec))
    howmanyphi = 4**length

    counter = 0
```

sum = 0
#assignment of 0 1 2 3 to distinct variables
#encoded from counter: quadratic representation
# of counter

while (counter < howmanyphi):
    phi = []
    assign = []
    temp = counter
    for i in range(length - 1, -1, -1):
        phi.append(temp / (4**i))
        temp = counter % (4**i)

    j = 0
    for i in range(0,6):
        if (charvec[i] in charvec[0:i]):
            # first position of the variable i
            same = charvec.index(charvec[i])
            assign.append(assign[same])
        else:
            assign.append(phi[j])
            j = j+1
    if len(assign) != 6:
        print 'problem'
        #print assign
        #Actually compute X_lX_l'
        X_1 = X_p[caseFind(assign[0:3])]
        #print X_1
        X_2 = X_p[caseFind(assign[3:6])]
        #print X_2
        sum = sum + X_1 * X_2
    counter = counter + 1

print sum / howmanyphi

def caseFind(assign):
    if assign[0] == assign[1] == assign[2]:
        return 0
    elif (assign[1] == assign[2]):
        return 1
    elif (assign[0] in assign[1:3]):
        return 2
    elif (assign[1] < assign[0] < assign[2])
        or (assign[2] < assign[0] < assign[1]):
        return 3
    else:
        return 4
Curriculum Vitæ

Matthias Mnich was born in Parchim, Germany, on February 24, 1983. He studied mathematics at the universities of Halle-Wittenberg, Kaiserslautern and Jena in Germany, and obtained his Master of Science in Applicable Mathematics from the London School of Economics and Political Science, United Kingdom.

From October 2006, Matthias was a PhD student in the Combinatorial Optimization group at Eindhoven University of Technology, in the Netherlands, under the supervision of Gerhard Woeginger. He will defend his thesis on September 22, 2010.

In 2008, Matthias was involved in the production of a mathematical theater play, that was awarded a prize within the “Year of Mathematics” and had its premiere at the Höhenrausch Festival 2008 in Rostock. In 2010, he was awarded the Philips Prize of the Royal Mathematical Society in the Netherlands.

Matthias is keen on traveling and hiking, and takes an interest in sports and cultural activities.
Bibliography


Computer Science. An EATCS Series. Springer-Verlag. (pp. 19, 20, 30, 31, 185)

compression and exact algorithms. Theoretical Computer Science, 411(7-9):1045 
– 1053, 2010. (p. 90)

feedback vertex set problem: exact and enumeration algorithms. Algorithmica, 
52(2):293–307, 2008. (pp. 17, 89)

[107] F. V. Fomin, F. Grandoni, and D. Kratsch. Some new techniques in design and 
analysis of exact (exponential) algorithms. Bulletin of the EATCS, 87:47–77, 
2005. (p. 14)

[108] F. V. Fomin, F. Grandoni, and D. Kratsch. Solving connected dominating set 

[109] F. V. Fomin, F. Grandoni, and D. Kratsch. A measure & conquer approach for 

[110] F. V. Fomin, F. Grandoni, A. V. Pyatkin, and A. A. Stepanov. Combinatorial 
bounds via measure and conquer: Bounding minimal dominating sets and 

algorithms for partial cover problems. In IARCS Annual Conference on Founda-
tions of Software Technology and Theoretical Computer Science, volume 4 of Leib-
niz International Proc. Informatics, pages 193–201. Schloss Dagstuhl–Leibniz-
Zentrum für Informatik, 2009. (pp. 178, 188)

and kernels. In Proc. 21st Annual ACM-SIAM Symposium on Discrete Algo-
(pp. 83, 179, 180, 184, 188)

[113] F. V. Fomin and A. V. Pyatkin. Finding minimum feedback vertex set in bipar-
tite graph. Reports in Informatics 291, University of Bergen, Norway, Febru-
ary 2005. (p. 105)

[114] F. V. Fomin and D. M. Thilikos. Fast parameterized algorithms for graphs 
on surfaces: linear kernel and exponential speed-up. In Automata, Languages 
Springer, 2004. (p. 177)

[115] F. V. Fomin and D. M. Thilikos. Dominating sets in planar graphs: branch-
2006. (p. 177)


[209] A. Perlis. Personal communication to Richard Lipton. Date unknown. (p. 13)


