The simple, little and slow things count: On parameterized counting complexity

Radu Curticapean

Dissertation for Obtaining the Title of

Doctor rerum naturalium (Dr. rer. nat)

of the Faculties of Natural Sciences and Technology
of Saarland University

Saarbrücken, Germany
2015

Colloquium Information

Date:	22.6.2015 14:30 CEST Saarbrücken, Germany
Dean:	Prof. Dr. Markus Bläser *Saarland University*
Chairman:	Prof. Dr. Gert Smolka *Saarland University*
Reviewers:	Prof. Dr. Markus Bläser *Saarland University*
	Prof. Dr. Martin Grohe *Rheinisch-Westfälische Technische Hochschule Aachen*
	Dr. Dániel Marx *Institute for Computer Science and Control,* *Hungarian Academy of Sciences (MTA SZTAKI)*
Scientific Assistant:	Dr. Tobias Mömke *Saarland University*

Acknowledgements

I wish to express my gratitude to my advisor Markus Bläser, who introduced me to the area of complexity theory in 2008 and supported me constantly since then, especially in the last year prior to completing my PhD. I am very glad that he always supplied me with the space I needed for my research while pointing out interesting problems to study and being reliably available for oracle calls whenever required. Without this guidance, the present thesis would not exist.

Thanks also to my collaborators Mingji Xia in Beijing and Dániel Marx in Budapest, whose work and ideas directly contributed to this thesis. I consider myself very lucky to be able to collaborate with such experts in counting and parameterized complexity. I also thank Holger Dell for interesting discussions and for proofreading parts of this thesis.

I am grateful to my external reviewers Martin Grohe and Dániel Marx for taking the time to read and assess the work presented in the remaining 230+ pages, and for traveling to my PhD defense. Thanks also to Gert Smolka, my very first lecturer, for being the chairman of my PhD committee and thus "closing the circle" of my education at Saarland University.

During the last years, the institutionalized coffee breaks with Marvin, Max, Thomas, and Karl were an important source for caffeine and gloriously absurd conversations. We should definitively meet again sometime in this constellation. Thanks a lot to Bernd, Bojana, Britta and Verena for helping me with accommodation during the last winter. Thanks also to all involved in making the awesome hat I received after my defense.

I wish to express my deep gratitude to my parents Dan and Dana Curticapean for their unwavering support, for the developments we underwent together, and for providing a safe haven, including non-pizza food, close (but not too close) to Saarbrücken.

Thanks also to all people who indirectly supported me during the past years. This includes the creators of Adventure Time and all musicians whose work helped me focusing during mine, including Tycho and Boards of Canada.

Finally, thank you for reading this thesis.

Abstract

In this thesis, we study the parameterized complexity of counting problems, as introduced by Flum and Grohe [FG04]. This area mainly involves questions of the following kind: On inputs x with a parameter k, can we solve a given counting problem in time $f(k)|x|^{\mathcal{O}(1)}$ for a function f that depends only on k? In the positive case, we call the problem *fixed-parameter tractable* (fpt). Otherwise, we try to prove its #W[1]-hardness, which is the parameterized analogue of #P-hardness.

We introduce a general technique that bridges parameterized counting complexity and the so-called *Holant framework*, which was already used successfully in the study of classical counting problems. We then apply this technique to the problem of counting perfect matchings (or equivalently, the permanent) subject to structural parameters of the input graph G: On the algorithmic side, we introduce a new tractable structural parameter, namely, the minimal size of an excluded single-crossing minor of G. We complement this by showing that counting perfect matchings is #W[1]-hard when parameterized by the size of an *arbitrary* excluded minor.

Then we turn our attention to counting general subgraphs H other than perfect matchings in a host graph G. Instead of imposing structural parameters on G, we parameterize by the size of H, giving rise to the problems $\#\mathsf{Sub}(\mathcal{H})$ for fixed graph classes $\mathcal{H}$: For inputs H and G with $H \in \mathcal{H}$, we wish to count H-copies in G. Here, $\mathcal{H}$ could be the class of matchings, cycles, paths, or any other recursively enumerable class. We give a full dichotomy for these problems: Either $\#\mathsf{Sub}(\mathcal{H})$ has a polynomial-time algorithm or it is #W[1]-complete. Assuming that $\mathsf{FPT} \neq \#\mathsf{W}[1]$, we can thus precisely identify the graph classes $\mathcal{H}$ for which the subgraph counting problem $\#\mathsf{Sub}(\mathcal{H})$ admits polynomial-time algorithms.

Furthermore, we obtain an unexpected application of our extensions to the Holant framework: We show that, given two unweighted graphs, it is $\mathsf{C}_=\mathsf{P}$-complete to decide whether they have the same number of perfect matchings.

Finally, we prove conditional lower bounds for counting problems under the counting exponential-time hypothesis #ETH. This hypothesis, introduced by Dell et al. [DHM$^+$14], asserts that the satisfying assignments to n-variable formulas in 3-CNF cannot be counted in time $2^{o(n)}$. Building upon this, we introduce a general technique that allows to derive tight lower bounds for other counting problems, such as counting perfect matchings, the Tutte polynomial, and the matching polynomial.

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der parametrisierten Komplexität von Zählproblemen, einem von Flum und Grohe [FG04] gegründeten Gebiet, in welchem Fragen der folgenden Art betrachtet werden: Können gegebene Probleme auf Eingaben x mit Parameter k in Zeit $f(k)|x|^{\mathcal{O}(1)}$ gelöst werden, wobei f eine Funktion ist, die nur von k abhängt? Im positiven Falle bezeichnen wir das Problem als parametrisierbar (FPT). Andernfalls versuchen wir typischerweise, dessen #W[1]-Härte zu beweisen − diese lässt sich vereinfachend als ein parametrisiertes Äquivalent der #P-Härte auffassen.

Wir führen zunächst eine allgemeine Technik ein, welche die parametrisierte Zählkomplexität mit dem sogenannten Holant-Rahmenwerk verbindet. Anschließend setzen wir diese zum Zählen perfekter Paarungen (oder äquivalent, zur Auswertung der Permanente) unter strukturellen Parametern des Eingabegraphens G ein: Wir zeigen, dass das Zählen perfekter Paarungen parametrisierbar ist durch die minimale Größe eines ausgeschlossenen Minors von G, der höchstens eine Kreuzung besitzt. Dieses algorithmische Resultat komplementieren wir durch die #W[1]-Härte des Zählens perfekter Paarungen, wenn die minimale Größe eines *beliebigen* ausgeschlossenen Minors als Parameter betrachtet wird.

Anschließend widmen wir uns dem Zählen beliebiger Subgraphen H in Graphen G. Anstelle von strukturellen Parametern betrachten wir die Größe von H als Parameter und erhalten hierdurch die Probleme #Sub($\mathcal{H}$) für feste Graphklassen $\mathcal{H}$: Auf Eingaben H und G mit $H \in \mathcal{H}$ gilt es, die H-Kopien in G zu zählen. Hierbei kann $\mathcal{H}$ die Klasse der Paarungen, Zyklen, Pfade, oder eine beliebige andere Klasse von Graphen darstellen. Wir zeigen eine vollständige Dichotomie für diese Probleme: Das Problem #Sub($\mathcal{H}$) ist entweder in P oder #W[1]-hart. Unter der gängigen Annahme FPT $\neq$ #W[1] erhalten wir somit eine vollständige Klassifikation der Polynomialzeit-lösbaren Probleme #Sub($\mathcal{H}$).

Weiterhin erhalten wir eine unerwartete Anwendung unserer Erweiterungen des Holant-Rahmenwerks: Wir zeigen die $\mathsf{C}_{=}$P-Vollständigkeit der Frage, ob die Anzahlen perfekter Paarungen in zwei gegebenen ungewichteten Graphen übereinstimmen.

Schlussendlich zeigen wir bedingte untere Schranken für Zählprobleme unter der Zählversion der Exponentialzeithypothese #ETH, eingeführt durch Dell et al. [DHM$^+$14]. Diese postuliert, dass die erfüllenden Belegungen in 3-KNF-Formeln mit n Variablen nicht in Zeit $2^{o(n)}$ gezählt werden können. Darauf aufbauend führen wir eine allgemeine Technik ein, die es ermöglicht, scharfe untere Schranken für andere Zählprobleme zu erhalten: Dies umfasst das Zählen perfekter Paarungen, das Tutte-Polynom und das Paarungs-Polynom.

Introduction to this Thesis

The study of counting problems has become a classical subfield of computational complexity theory since Valiant's seminal paper [Val79b] that introduced the complexity class #P and proved that counting perfect matchings is complete for this class. For instance, it is now known by Toda's celebrated result [Tod91] that the problems in the polynomial-time hierarchy PH can be solved in polynomial time with an oracle to #P, and this establishes counting solutions as something much harder than deciding their existence. The fact that counting problems are typically hard has also allowed to map the complexity of counting solutions to constraint satisfaction problems exhaustively [Bul13, CC12], a situation yet to be brought about for their decision versions.

From day zero, the development of counting complexity theory was steadily accompanied by the concrete problem of *counting perfect matchings*, which is also a focus of the present thesis. In algebra, the quantity of perfect matchings (in a bipartite graph G) is known as the *permanent* (of the biadjacency matrix of G), and it is central to the field of *algebraic complexity theory* [Agr06]. Furthermore, a celebrated randomized *approximation scheme* for the number of perfect matchings in bipartite graphs has been obtained [JSV04], and the new field of *holographic algorithms* [Val08] continues to push the limits of polynomial-time solvability, ultimately by reductions to counting perfect matchings.

In fact, counting problems have already been studied in the area of statistical physics, long before the advent of computational complexity theory, under the notion of so-called *partition functions* [TF61, Kas61, Kas67]. In this area, the problem of counting perfect matchings is known as the partition function of the *dimer model*, and the first algorithms for counting perfect matchings stem from this area. This includes, most notably, a polynomial-time algorithm for planar graphs [Kas67]. As noted in [DHM$^+$14], physicists at that time were puzzled about their inability to "exactly solve" the dimer model and other partition functions (on not necessarily planar graphs), and the reasons for this became clear only with Valiant's result on the #P-completeness of counting perfect matchings.

Being active research targets, counting problems have been refined along various dimensions in the past, including approximate counting [DL92, JS93, AR02, JSV04, GJ08, GJ14b, GJ14a, CDG$^+$15], counting modulo a fixed number [Her90, Val06, GHLX11, GLV13, GGR14, GGR15], counting on restricted graph classes [DL92, Vad01, XZZ07], and combinations thereof.

In this thesis, we consider the *parameterized complexity* of counting problems, following in the wake of Flum and Grohe's seminal paper [FG04], which introduced one of the most recent refinements to counting complexity. Parameterized complexity theory is a relatively young field within complexity theory [DF95, DF99], and it is concerned with questions of the following kind: On inputs x with a parameter $k \in \mathbb{N}$, can we solve a given problem in

time $f(k)n^{\mathcal{O}(1)}$ for a computable function f that depends only on k, but not on $n = |x|$? Problems that can be solved in such running times are called *fixed-parameter tractable*.

The field of parameterized complexity theory was initiated with a focus on *decision* problems, and it was later extended to counting problems in [FG04, McC06]. For instance, we can count k-vertex covers in time $2^k n^{\mathcal{O}(1)}$ [FG06], and we can compute the number of perfect matchings in graphs of genus g in time $4^g n^{\mathcal{O}(1)}$ [GL98]. This makes these problems fixed-parameter tractable with respect to their parameters, which are the solution size k, and the genus g, respectively. But what about, say, counting matchings with k edges in a general graph? Or what about counting matchings with precisely k *unmatched* vertices in a planar graph? We will answer these and various other questions in this thesis and try to explain why they are particularly interesting.

As mentioned before, a clear focus of this thesis lies on variations on the theme of counting perfect matchings. We will also often consider the following *weighted* number of perfect matchings, for edge-weighted graphs G that are given together with a weight function $w : E(G) \to \mathbb{Q}$:

$$\text{PerfMatch}(G) = \sum_{\substack{M \text{ is a perfect} \\ \text{matching of } G}} \prod_{e \in M} w(e).$$

While we have already remarked above that the computation of PerfMatch is an important and well-studied problem on its own, there are further reasons for focusing on this problem: As we will observe in multiple occasions in this thesis, counting (perfect) matchings has the qualities of a "bottleneck problem": Once hardness for this problem can be established in a given model of interest, other results follow by comparatively easy reductions.

Guided tour of this thesis

In **Chapter 1**, we provide the necessary preliminaries for later chapters by introducing the notational conventions and concepts from complexity theory, graph theory, combinatorics and algebra that are required later on.

The author has invested quite some work in an attempt to embed the ideas presented in this thesis into a general framework. To this end, we use the *Holant framework*, a model for counting problems that has recently been introduced [Val08, CL07], and which has proven immensely useful to the study of classical counting problems. In **Chapter 2**, we present our interpretation of this framework and introduce extensions towards parameterized complexity, some of which were obtained in joint work with Mingji Xia. Additionally, we present a direct and uniform reduction from a large class of counting problems to PerfMatch, including the problem #SAT in particular. This gives a flexible #P-completeness proof for PerfMatch, which is useful for other applications as well.

The main body of this thesis is then divided into Parts I-III, which contain more actual technical results. Each such part begins with an extended introduction into the material it covers, including notes and references that state possible involved collaborators.

Part I - Counting perfect matchings in H-minor-free graphs

In the first part, we study the problem of counting perfect matchings in graphs excluding a fixed minor. After observing that essentially every known polynomial-time solvable graph class for counting perfect matchings excludes a fixed minor, our initial goal was to obtain a polynomial-time algorithm for *every* class excluding a fixed minor H.

To cut short, we could not solve this general problem. However, in **Chapter 3**, we present a polynomial-time algorithm for computing PerfMatch on every graph class that excludes a fixed *single-crossing minor*, a minor that can be drawn in the plane with at most one crossing. Graphs excluding such minors can be obtained by gluing planar graphs and bounded-treewidth graphs along triangles, and their structural complexity pales in comparison to that of general H-minor-free graphs. But even this simpler structure was not fully exploited in the literature yet, and our algorithm generalizes almost all known algorithms [Kas67, Lit74, Vaz89, STW14], with one exception [GL98]. The ideas used in our algorithm also embed nicely into the Holant framework.

On the complexity side, we prove in joint work with Mingji Xia that one of the building blocks of general H-minor free graphs, namely *apices*, make counting perfect matchings intractable. More precisely, we show in **Chapter 4** that it is #W[1]-hard to count the perfect matchings in a graph that can be made planar after removal of k apex vertices. This is the first major application of our parameterized Holant framework. We obtain as a simple consequence that counting perfect matchings in a graph G is #W[1]-hard when parameterized by the *Hadwiger number* of G, the size of the largest clique minor in G. From this, we can conclude that algorithms for PerfMatch on H-minor-free graphs cannot be fixed-parameter tractable in the size of H.

In the same chapter, we also extend the hardness proof for k-apex graphs by showing that it is already #W[1]-hard to count matchings with exactly k unmatched vertices in planar graphs. This solves a conjecture by the author from 2010. Finally, we also complement the result by an fpt-algorithm for graphs whose apices can see only a bounded number of faces, and we will explain why this is interesting in the context of H-minor-free graphs.

Part II - Counting small subgraphs

In the second part of the thesis, which is joint work with Dániel Marx, we consider the following problem #Sub($\mathcal{H}$) for arbitrary fixed graph classes $\mathcal{H}$: Given a host graph G and a pattern graph $H \in \mathcal{H}$ as input, we want to count subgraphs of G that are isomorphic to H. This problem is considered to be parameterized by $|V(H)|$, so we are intuitively interested in counting occurrences of a small pattern in a large host graph.

The family of problems #Sub($\mathcal{H}$) includes counting paths, cycles and matchings of size k, and literally more problems than can be described with words.[1] The problems of counting paths and cycles were proven #W[1]-complete in the seminal paper on parameterized counting complexity [FG04], which also posed the complexity of counting k-matchings as an open problem. This was later settled by the author, showing that this problem is

[1] We therefore restrict ourselves to recursively enumerable classes $\mathcal{H}$.

indeed #W[1]-complete [Cur13], and following up on joint work with Markus Bläser for a weighted version of the problem [BC12]. This original proof appears in the appendix of this thesis. In **Chapter 5**, we instead present a novel #W[1]-completeness proof for counting k-matchings that is significantly simpler and embeds nicely into the parameterized Holant framework. Apart from this, we also obtain an almost-tight running time lower bound of $n^{\Omega(k/\log k)}$ for counting k-matchings under the exponential-time hypothesis #ETH. By simple reductions, we then obtain #W[1]-hardness proofs and the same lower bounds for counting paths and cycles as well.

Building upon this, we then give a full dichotomy for the problem #Sub($\mathcal{H}$) in **Chapter 6**: If the maximum matching number of $\mathcal{H}$ is bounded, then it is known that #Sub($\mathcal{H}$) even admits a polynomial-time algorithm. On the other hand, if this number is unbounded, then we show that the problem is #W[1]-complete. Hence, assuming FPT $\neq$ #W[1], we can precisely say when #Sub($\mathcal{H}$) has a polynomial-time algorithm. The main part of the hardness proof is a reduction from counting k-matchings to #Sub($\mathcal{H}$) that is enabled by a machinery of gadgets whose existence was proven by Dániel Marx in an involved graph-theoretical analysis. The author of this thesis explicitly denies any contribution to this latter part, which appears in the appendix.

Part III - Quantitative lower bounds

In the final part of this thesis, we study conditional lower bounds for counting problems under the exponential time hypothesis #ETH introduced in [DHM$^+$14]. This is the counting analogue of its decision version ETH, introduced in [IPZ01, IP01]. It turns out, in this setting as well, that lower bounds for counting perfect matchings give a formidable base camp for further expeditions.

The #P-hardness proof for PerfMatch from Chapter 2 will also provide us with a tight lower bound under #ETH, however only on graphs with edge-weights -1 and 1, and we will observe that the classical reductions to unweighted graphs are incompatible with tight lower bounds. Still, we would like to remove the weight -1, e.g., because it is not clear how to simulate this weight in reductions from PerfMatch to other problems. We solve this with two orthogonal approaches, each of which has its own specific benefits.

In **Chapter 7**, we show, given an edge-weighted graph with weights -1 and 1, how to construct two unweighted graphs G_1 and G_2 such that PerfMatch(G) is the mere difference of PerfMatch(G_1) and PerfMatch(G_2). The size of these graphs depends linearly on the size of G. And while our technique was conceived for quantitative lower bounds, it can be used as well to investigate the *structural* complexity of the problem PerfMatch: Using it, we show that deciding whether PerfMatch agrees on two unweighted graphs is complete for the complexity class C$_=$P. To the best of our knowledge, this is the first C$_=$P-completeness result for a counting problem that is not #P-complete under parsimonious reductions.

Our second approach to remove the weight -1 is less specific to the actual problem PerfMatch we started with: In **Chapter 8**, we present a collection of ideas that we dub the *block interpolation framework*. In short, this framework applies to #P-hardness proofs that are based around univariate interpolation, a technique we can observe to be largely

incompatible with tight lower bounds. We identify a certain pattern in such proofs and show that their centerpiece, the interpolation step, can be uniformly replaced by a "block interpolation" step, a certain variation of multivariate interpolation we introduce in this thesis. Using this framework, we can exemplarily prove tight lower bounds (that were not known before) for PerfMatch on *unweighted* graphs, as well as for the evaluation of the matching polynomial, the independent set polynomial, and the Tutte polynomial.

Appendices

In **Appendix A**, we include the first #W[1]-hardness proof for counting k-matchings, with minor editorial changes from the version that appeared in [BC12, Cur13]. At least from the present point of view, this proof has been entirely superseded by the material we present in Chapter 5. However, the techniques used in these two proofs differ substantially, and the old proof might therefore not be entirely obsolete.

The remaining appendices contain deferred proofs from the main part. In particular, **Appendix B** proves the existence of certain gadgets that we require in order to prove the #W[1]-hardness result for uncolored subgraph counting in Chapter 6. The content of Appendix B was authored by Dániel Marx, and it appears isolated from the main text to help clarifying that it was not written by the author of the present thesis.

Finally, in some instances, we resort to computer-aided verifications of certain claims. To make these accessible to the reader, we wrote a simple verification script in MATLAB, listed in **Appendix C**, whose output proves such claims when given the relevant data as input.

Contents

1. Preliminaries

Throughout this thesis, we will use some notational conventions that we make explicit in Section 1.1. We then proceed, in Section 1.2, to give brief introductions into the subfields of computational complexity theory that appear in this thesis. Furthermore, we introduce graph polynomials in Section 1.3 and certain useful algebraic techniques in Section 1.4, before concluding with a trimmed introduction into graph minor theory in Section 1.5.

1.1. Notational conventions

1.1.1. Sets, numbers, and polynomials

For $n \in \mathbb{N}$, we write $[n] = \{1, \dots, n\}$. Given a set A, we write $\#A = |A|$ for the cardinality of A and call A a *k-set* if $k = |A|$. Given sets A and B, we define the following notions.

- We write both B^A and $A \to B$ for the set of all functions from A to B. In the case $B = \{0, 1\}$, we call an element $a \in \{0, 1\}^A$ a *binary assignment* to A.
  - For $f : A \to B$, we write $\operatorname{supp}(f) = f^{-1}(B \setminus \{0\})$ for the support of f.
  - When clear from the context, we will sometimes identify $\{0, 1\}^A$ with the power set $2^A = \{S \mid S \subseteq A\}$. In particular, for $a \in \{0, 1\}^A$ and $x \in A$, we may write $x \in a$ if $a(x) = 1$.
  - For $d \in \mathbb{N}$, we sometimes identify the set of assignments $\{0, 1\}^{[d]}$ with the set of d-bitstrings $\{0, 1\}^d$ in the canonical way.
  - For strings $x \in \{0, 1\}^*$, we write $\operatorname{hw}(x)$ for the number of 1-entries in x, and we extend this notion canonically to binary assignments $x \in \{0, 1\}^A$.
- We define certain subsets of $A \times B$. This notation is used only when A and B are clear from the context.
  - For $b \in B$, we write $(\star, b) = \{(a, b) \mid a \in A\}$ for the column at b.
  - For $a \in A$, we write $(a, \star) = \{(a, b) \mid b \in B\}$ for the row at a.
- We write $A \simeq B$ if there exists a bijection between A and B.

For $k \in \mathbb{N}$, let us say that $(i, j) \in [k]^2$ and $(i', j') \in [k]^2$ are *vertically adjacent* if $|i - i'| = 1$ and $j = j'$. Likewise, we call these pairs *horizontally adjacent* if $|j - j'| = 1$ and $i = i'$.

We will also frequently use the Iverson bracket notation: Given a statement P, let

$$[P] = \begin{cases} 1 & \text{if } P \text{ holds,} \\ 0 & \text{otherwise.} \end{cases}$$

Given an indeterminate x and $k \in \mathbb{N}$, we write $(x)_k$ for the falling factorial

$$(x)_k = x \cdot (x-1) \cdot \ldots \cdot (x-k+1).$$

Note that $(x)_k$ is a polynomial of degree k. Generally, we denote the *degree* of a polynomial $p \in \mathbb{Q}[x]$ by $\deg(p)$.

If $\mathbf{x} = (x_1, \ldots, x_t)$ is a list of indeterminates, for $t \in \mathbb{N}$, then we write $\mathbb{N}^{\mathbf{x}}$ for the set of all monomials over $\mathbf{x}$. Note that each monomial is a *finite* product of indeterminates. A *multivariate polynomial* $p \in \mathbb{Q}[\mathbf{x}]$ is a polynomial

$$p = \sum_{\theta \in \mathbb{N}^{\mathbf{x}}} a(\theta) \cdot \theta$$

with $a(\theta) \in \mathbb{Q}$ for all $\theta \in \mathbb{N}^{\mathbf{x}}$, where a has finite support. The polynomial p *contains* a given monomial $\theta \in \mathbb{N}^{\mathbf{x}}$ if $a(\theta) \neq 0$ holds.

If x is an indeterminate from $\mathbf{x}$, then we write $\deg_x(p)$ for the *degree of* x in p. This is the maximum number $k \in \mathbb{N}$ such that p contains a monomial θ with factor x^k. Generally, for a subset $\mathbf{y}$ of $\mathbf{x}$, we denote the *total degree of* $\mathbf{y}$ in p as the maximum degree of any monomial $\mathbb{N}^{\mathbf{y}}$ that is contained as a factor of a monomial in p.

Furthermore, if $p \in \mathbb{Q}[x, y]$ is a bivariate polynomial and $\xi \in \mathbb{Q}$ is some arbitrary fixed value, we write $p(\cdot, \xi)$ for the result of the substitution $y \leftarrow \xi$ in p, and we observe that $p(\cdot, \xi) \in \mathbb{Q}[x]$. Likewise, we write $p(\xi, \cdot)$ for the result of substituting $x \leftarrow \xi$.

1.1.2. Graphs

Every chapter will rely on graphs in some way. Unless otherwise stated, graphs are undirected and feature no self-loops. We denote the vertices of a graph G by $V(G)$ and its edges by $E(G)$, and we abbreviate undirected edges $\{u, v\}$ by uv.

A graph G is *simple* if G features at most one edge $uv \in E(G)$ between vertices $u, v \in V(G)$. Otherwise, G is a multigraph. While multigraphs will be used in several occasions in this thesis, all hardness results we prove will hold even when restricted to simple graphs. This is not always trivial to ensure, but it will be prove to be relevant.

When a graph named G is present in the context of an argument, its *order* $|V(G)|$ will often be denoted by n. Likewise, whenever a graph named H is present in the context, we often write $k = |V(H)|$.

Given a graph $G = (V, E)$ and a vertex $v \in V$, let $I(v)$ denote the set of edges incident with v, and note that $|I(v)| = \deg(v)$, where $\deg(v)$ denotes the *degree* of v. We call $v \in V$ *isolated* if $\deg(v) = 0$, and we write $\Delta(G)$ for the *maximum degree* of G.

Given a vertex set $A \subseteq V$, we write $G - A$ for the graph obtained from G by deleting all vertices in A and their incident edges, and we write $G[A] = G - (V \setminus A)$ for the *subgraph induced by* A. Given an edge set $B \subseteq E$, we write $G[B]$ for the *subgraph induced by* B, which is obtained from G by deleting all edges in $E \setminus B$ and all isolated vertices in the resulting graph, unless we explicitly state that isolated vertices are kept.

We consider the following substructures of graphs $G = (V, E)$ on n vertices.

- A *k-clique* in G is a k-set $K \subseteq V$ with $uv \in E$ for all $u, v \in K$ with $u \neq v$. An *independent k-set* in G is a k-set $I \subseteq V$ with $uv \notin E$ for all $u, v \in I$.
- A *k-vertex cover* is a k-set $C \subseteq V$ such that $C \cap e \neq \emptyset$ for all $e \in E$.
- A *k-matching* is a k-set $M \subseteq E$ of vertex-disjoint edges. We write $\mathrm{usat}(G, M)$ for the set of vertices $v \in V$ that are not contained in any edge of M, and for $\ell \in \mathbb{N}$, we say that M is an *ℓ-defect matching* if $|\mathrm{usat}(M)| = \ell$ holds. We sometimes merely write $\mathrm{usat}(M)$ when G is clear from the context. It is trivially observed that $\ell + 2k = n$ holds for any ℓ-defect k-matching of G.
- A *subgraph of G* is a graph that can be obtained from G by deletion of edges and/or vertices. We write $H \subseteq G$ if H is a subgraph of G. An *induced subgraph of G* is a graph that can be obtained from G by deletion of vertices together with all of their incident edges.
- For a graph H, any subgraph F of G that is isomorphic to H is called a *H-copy in G*. Likewise, we call any *induced* subgraph F of G that is isomorphic to H an *induced H-copy in G*.

A graph G may be *edge-weighted*, then it is given together with a weight function $w : E(G) \to \mathbb{Q}$, or *vertex-weighted*, then the weight function is of type $w : V(G) \to \mathbb{Q}$.

We will also consider *vertex-colored* graphs and *edge-colored* graphs. In the following, let $k \in \mathbb{N}$ and let G be an unweighted graph.

- We say that a pair (G, c) with $c : V(G) \to [k]$ is a *$[k]$-vertex-colored* graph.[1] For $i \in [k]$, we write $V_i(G)$ for the set of i-colored vertices in G, and for $i, j \in [k]$, we write $E_{i,j}(G)$ for the set of edges between vertices of colors i and j. We call a set $S \subseteq V(G)$ (vertex-)colorful if $|S \cap V_i(G)| = 1$ holds for all $i \in [k]$. Note that S then is a k-set.
- The pair (G, c) is *a $[k]$-edge-colored* graph if G is given together with a function $c : E(G) \to [k]$. We call a set $S \subseteq E(G)$ (edge-)colorful if S contains exactly one edge of color i, for all $i \in [k]$.

Finally, a *plane graph* is a pair (G, π), where π is a planar drawing of G. A graph G is planar if G admits a planar drawing. Given a plane graph (G, π) and a simple cycle C in G, we say that C *bounds a face* in (G, π) if one of the two regions bounded by C in π contains no vertices or edges.

1.2. Computational complexity theory

We provide some basic notions from counting complexity theory, as introduced in [Val79a], from parameterized counting complexity [FG04], and from the emerging area of exponential-time complexity [IP01, IPZ01], especially for counting problems [DHM$^+$14]. In particular,

[1] Note that c is not required to be a *proper* k-coloring, that is, we allow $c(u) = c(v)$ for $uv \in E$.

in Section 1.2.1, we define the "classical" counting complexity classes

$$\mathsf{FP}, \quad \#\mathsf{P}, \quad \oplus\mathsf{P}, \quad \mathsf{Mod_t P} \text{ for } t \in \mathbb{N},$$

then proceed in Section 1.2.2 with the parameterized classes

$$\mathsf{FPT}, \quad \mathsf{W}[1], \quad \oplus\mathsf{W}[1], \quad \#\mathsf{W}[1], \quad \#\mathsf{W}[2], \quad \mathsf{XP}, \quad \mathsf{Mod_t W}[1] \text{ for } t \in \mathbb{N},$$

and conclude in Section 1.2.3 with the exponential-time hypotheses

$$\mathsf{ETH} \quad \text{and} \quad \#\mathsf{ETH}.$$

In the course of these sections, we also encounter the following reduction notions:

$$\begin{matrix} \leq_p & \leq_p^{pars} & \leq_p^T & \leq_p^{lin} \\ \leq_{fpt} & \leq_{fpt}^{pars} & \leq_{fpt}^T & \leq_{fpt}^{lin} \end{matrix} \quad \leq_{serf}$$

1.2.1. Classical counting complexity

Computational complexity theory mostly focuses on decision problems, such as the well-known NP-complete problem SAT, which asks to *decide* whether a given CNF-formula φ is satisfiable. In applications, it might however also be required to *find* a solution, to output *all* solutions, or to determine the *number* of solutions. This last task gives rise to the notion of *counting problems*, for which we show three examples:

#SAT: Given as input a CNF-formula φ on variables X, determine the number of assignments $a \in \{0,1\}^X$ that satisfy φ.

#HamCycle: Given a directed graph G as input, count the Hamiltonian cycles in G, i.e., simple cycles in G that visit every vertex exactly once.

perm^{0,1}: Given an undirected, unweighted bipartite graph G as input, count the perfect matchings in G.[2]

In the following definition, we "formalize" the notion of counting problems and define a complexity class #P, which can be considered as the analogue of NP for the field of counting complexity. It can be easily observed that the three exemplary problems are all contained in #P.

Definition 1.1 ([Val79a])**.** A *counting problem* is a function $\mathsf{A} : \{0,1\}^* \to \mathbb{Q}$. For the purposes of this thesis, let the class FP denote all counting problems that can be solved in polynomial time, i.e., in time $|x|^{\mathcal{O}(1)}$ on inputs $x \in \{0,1\}^*$.

The class #P contains all counting problems $\mathsf{A} : \{0,1\}^* \to \mathbb{N}$ that admit a non-deterministic Turing machine M_A such that the following holds: For all $x \in \{0,1\}^*$, the

[2] The problem perm^{0,1} is central to this thesis and will be revisited later in the preliminaries.

length of each computation path of M_{A} is bounded by $|x|^{\mathcal{O}(1)}$, and the number of accepting paths of M_{A} on x is equal to $\mathsf{A}(x)$.

Every language $L \in \mathsf{NP}$ can be trivially decided with an oracle for $\#\mathsf{P}$: If L is contained in NP, then there is a polynomially time-bounded non-deterministic Turing machine M that has an accepting path on input $x \in \{0,1\}^*$ iff $x \in L$. If we can *count* the accepting paths of M on x, we can trivially decide whether one exists. In other words, we have $\mathsf{NP} \subseteq \mathsf{P}^{\#\mathsf{P}}$. Even more so, by Toda's theorem [Tod91], the entire polynomial-time hierarchy PH is contained in $\mathsf{P}^{\#\mathsf{P}}$. Since PH is believed to be infinite and contains NP at its first level, we obtain convincing evidence that the hardest problems in $\#\mathsf{P}$ are much harder than NP-complete problems, such as SAT.

Completeness and hardness for $\#\mathsf{P}$

To identify the hardest problems in $\#\mathsf{P}$, we proceed as usual in complexity and say that a counting problem B is $\#\mathsf{P}$-*hard* if every problem $\mathsf{A} \in \#\mathsf{P}$ admits a reduction to B. However, for reasons that will become clear soon, different types of reductions (and consequently, different types of hardness) are distinguished for counting problems:

Firstly, there are *parsimonious* reductions that reduce instances x of A to instances x' of B such that $\mathsf{A}(x) = \mathsf{B}(x')$ holds. This strict notion of reduction can be relaxed by allowing normalization factors, which gives rise to *weakly parsimonious* reductions. By providing the reduction with full oracle access to B, the requirement on a reduction can be relaxed even further, and we obtain *Turing* reductions.

Definition 1.2. Let A and B be counting problems.

- Let $f : \{0,1\}^* \to \{0,1\}^*$ and $g : \{0,1\}^* \to \mathbb{Q}$. If $\mathsf{A}(x) = g(x) \cdot \mathsf{B}(f(x))$ holds for all $x \in \{0,1\}^*$, then we call (f, g) a *weakly parsimonious reduction* from A to B. If f and g can be computed in polynomial time, then we call (f, g) a *weakly parsimonious polynomial-time reduction* and write $\mathsf{A} \leq_p \mathsf{B}$.
- If (f, g) is a weakly parsimonious reduction, and additionally $g(x) = 1$ holds for all $x \in \{0,1\}^*$, then we call f *parsimonious*. If f can be computed in polynomial time, then we call f a *parsimonious polynomial-time reduction* and write $\mathsf{A} \leq_p^{pars} \mathsf{B}$.
- If $\mathbb{T}$ is a deterministic algorithm that solves A with an oracle for B, then we call $\mathbb{T}$ a *Turing reduction* from A to B. If $\mathbb{T}$ runs in polynomial time, then we call $\mathbb{T}$ a *polynomial-time Turing reduction* and write $\mathsf{A} \leq_p^T \mathsf{B}$.

The problem B is $\#\mathsf{P}$-*hard under parsimonious polynomial-time reductions* if $\mathsf{A} \leq_p^{pars} \mathsf{B}$ holds for all $\mathsf{A} \in \#\mathsf{P}$. Hardness under the notions $\leq_p$ and $\leq_p^T$ is defined likewise.

If A and B are counting problems, then we obviously have

$$\mathsf{A} \leq_p^{pars} \mathsf{B} \;\Rightarrow\; \mathsf{A} \leq_p \mathsf{B} \;\Rightarrow\; \mathsf{A} \leq_p^T \mathsf{B}. \tag{1.1}$$

Furthermore, it is obvious that all three reduction notions are transitive and that FP is closed under all three reductions. We discuss in the following why different reduction types are reasonable in the study of counting problems. To this end, let us introduce the following definitions: By the *decision version* of a problem $\mathsf{A} \in \#\mathsf{P}$ with associated Turing machine M_{A}, we denote the problem L_{A} of deciding, on input $x \in \{0,1\}^*$, whether M_{A} has an accepting path on x. Likewise, A is the *counting version* of L_{A}.

It can be observed that the classical NP-hardness reduction for SAT and the reduction from SAT to $\mathsf{HamCycle}$ are both parsimonious, so their counting versions $\#\mathsf{SAT}$ and $\#\mathsf{HamCycle}$ are $\#\mathsf{P}$-hard under parsimonious reductions. In fact, it is conjectured that *every* NP-complete decision problem has a $\#\mathsf{P}$-complete counting version, and some progress has been made towards a proof of this statement [FHT95, Liv09], but the general conjecture remains unresolved [Wil13].

However, there are counting problems in $\#\mathsf{P}$ that admit easy decision versions, and for such problems, we probably cannot show $\#\mathsf{P}$-hardness under weakly parsimonious reductions. For instance, if $\mathrm{perm}^{0,1}$ were $\#\mathsf{P}$-hard under $\leq_p$, then we could use classical polynomial-time algorithms for finding maximum matchings, such as [Edm87], to solve the decision problem SAT and hence prove $\mathsf{P} = \mathsf{NP}$. However, polynomial-time algorithms for $\mathrm{perm}^{0,1}$ are still very unlikely, since a seminal result of Valiant [Val79a] asserts that $\mathrm{perm}^{0,1}$ is $\#\mathsf{P}$-complete under *Turing* reductions. It was later shown [DL92] that this holds even if we may assume the input graph to have maximum degree 3.

Theorem 1.3 ([Val79a, DL92])**.** *The problem* $\mathrm{perm}^{0,1}$ *is* $\#\mathsf{P}$*-complete under* $\leq_p^T$*, even when restricted to (bipartite) input graphs with maximum degree* 3*.*

Non-parsimonious reductions, such as those used to prove Theorem 1.3, are *the* intellectual justification for studying counting complexity, as hardness results for counting problems would otherwise amount to nothing more than the copious task of checking whether NP-completeness reductions are parsimonious.

It was also shown in [Val79a] that a *weighted* version of $\mathrm{perm}^{0,1}$, namely $\mathrm{perm}^{-1,0,1}$, is in fact $\#\mathsf{P}$-hard under weakly parsimonious reductions. We will present an alternative proof of this fact in Section 2.2.3, and in Chapter 7, we will reduce this weighted version to the difference of two instances to counting perfect matchings (in not necessarily bipartite graphs). This gives us an alternative proof for the $\#\mathsf{P}$-completeness of counting perfect matchings, which additionally allows us to derive further consequences.

We note that counting problems can also be defined over ranges other than $\mathbb{Q}$, such as $\mathbb{R}$ or $\mathbb{C}$, provided that we can efficiently perform arithmetic on the relevant elements from these sets. The range of counting problems in $\#\mathsf{P}$ however *must* be fixed to $\mathbb{N}$ by

definition. Hence, for problems with range $\mathbb{Q}$, we will usually only speak of $\#\mathsf{P}$-*hardness* rather than completeness, since some $\#\mathsf{P}$-hard problems fail to be contained in $\#\mathsf{P}$ for the purely syntactical reason that their outputs might be non-integer.

Counting modulo a fixed number

An interesting family of ranges for counting problems arises from the residual class rings $\mathbb{Z}_t = \mathbb{Z}/t\mathbb{Z}$ for $t \in \mathbb{N}$. For *fixed* $t \in \mathbb{N}$, the problem of testing whether the output of an integer-valued counting problem is divisible by t gives rise to the modular counting class $\mathsf{Mod_t P}$, defined in [CH90, Her90].

Definition 1.4 ([CH90, Her90])**.** For $t \in \mathbb{N}$, the class $\mathsf{Mod_t P}$ contains all languages $L \subseteq \{0,1\}^*$ for which the following holds: There is a counting problem $\mathsf{A} \in \#\mathsf{P}$ such that, for all $x \in \{0,1\}^*$, we have $\mathsf{A}(x) \not\equiv_t 0$ iff $x \in L$. We write $\oplus\mathsf{P}$ for $\mathsf{Mod_2 P}$.

Hardness and completeness for $\mathsf{Mod_t P}$ are defined via the usual polynomial-time reductions for decision problems. If a counting problem $\mathsf{A} : \{0,1\}^* \to \mathbb{N}$ is $\#\mathsf{P}$-complete under parsimonious reductions, then for all $t \in \mathbb{N}$, it is obviously $\mathsf{Mod_t P}$-complete to decide whether $\mathsf{A}(x) \not\equiv_t 0$. If A is not $\#\mathsf{P}$-complete under parsimonious reductions, but possibly under weakly parsimonious reductions, then counting modulo a fixed number *might* admit a polynomial-time algorithm.

For instance, we will observe the following in Section 1.3.2: Given a bipartite graph G, the problem of counting its perfect matchings modulo 2 admits a simple algorithm with running time $\mathcal{O}(n^3)$ by reduction to the determinant, and it was shown in [Val79a] that, for $k \in \mathbb{N}$, perfect matchings can even be counted modulo 2^k in time $\mathcal{O}(n^{4k-3})$. This is however contrasted by the following hardness result for moduli that feature odd divisors, which is also implied by our alternative proof of Theorem 1.3.

Theorem 1.5 ([Val79a])**.** *Let $s, t \in \mathbb{N}$ be fixed such that s is an odd divisor of t. Given a bipartite unweighted graph as input, it is $\mathsf{Mod_s P}$-hard to compute the number of perfect matchings in G modulo t.*

We close this subsection by a remark on the position of the classes $\mathsf{Mod_t P}$ in the general landscape of complexity classes: By the Valiant-Vazirani theorem [VV86], it is known that SAT remains NP-complete (under randomized RP-reductions) even when we may assume that the input formula has at most one satisfying assignment. Here, a randomized reduction from L to L' is a polynomial-time randomized algorithm that, given an instance x for L, outputs an instance x' for L' such that:

- If $x \notin L$, then $x' \notin L'$.
- If $x \in L$, then with probability at least $\frac{1}{2}$, the output x' satisfies $x' \in L'$.

Since 1 is not divisible by $t > 1$, this implies that NP is contained in $\mathsf{Mod_t P}$ under randomized reductions, for all $t > 1$. In fact, Toda's theorem, which was already mentioned before, even proves that the polynomial-time hierarchy PH is contained in $\oplus\mathsf{P}$ under (a different notion of) randomized reductions.

1.2.2. Parameterized counting complexity

Parameterized counting complexity was introduced in [FG04, McC06] to bridge classical counting complexity and the relatively young field of parameterized complexity theory. In this field, the objects in study are *parameterized counting problems*, which are pairs A/π consisting of a counting problem $\mathsf{A} : \{0,1\}^* \to \mathbb{Q}$ and a *parameterization* $\pi : \{0,1\}^* \to \mathbb{N}$. Here, a parameterization is a polynomial-time computable function that assigns a parameter $\pi(x)$ to each instance x. Parameterized *decision* problems can be defined likewise, by replacing $\mathbb{Q}$ with $\{0,1\}$.

The idea underlying parameterized algorithms and complexity is to find sensible parameter functions $\pi(x)$, which may be independent of the input size $|x|$, and which allow for a more fine-grained complexity analysis than could be achieved by considering $|x|$ alone. As first examples, consider the following parameterized counting problems:

#Clique/k: Given as input a pair (G,k) consisting of a graph G and $k \in \mathbb{N}$, count the k-cliques in G. The parameterization π in this problem is defined by $(G,k) \mapsto k$.

#HittingSet/k: Given as input $(n,k,\mathcal{A})$ with $n,k \in \mathbb{N}$ and a set system $\mathcal{A} \subseteq 2^{[n]}$, count the hitting k-sets of $\mathcal{A}$. These are k-subsets $S \subseteq [n]$ that satisfy $S \cap A \neq \emptyset$ for all $A \in \mathcal{A}$. The parameterization is defined by $(n,k,\mathcal{A}) \mapsto k$.

#HittingSet/$k+d$: Count hitting sets as above, but with the parameterization $\pi : (n,k,\mathcal{A}) \mapsto k+d$, where $d = \max_{A \in \mathcal{A}} |A|$.

perm^{0,1}/apex: Given an unweighted bipartite graph G as input, count the perfect matchings of G, parameterized by $\mathrm{apex}(G)$, which is the minimum size of a set $S \subseteq V(G)$ such that $G - S$ is planar. We assume $\mathrm{apex}(G)$ to be given as part of the input; otherwise apex would not be a valid parameterization (unless $\mathsf{P} = \mathsf{NP}$), since it is NP-complete [LY80].

perm$^{0,1}/\Delta$: Count perfect matchings as above, but with a parameterization that maps $G \mapsto \Delta(G)$, where $\Delta(G)$ is the maximum degree of G.

For simplicity, we will always assume that $\pi(x)$ is given together with the input instance x. Furthermore, if the context allows it, we sometimes omit the explicit mention of the parameterization π and write A rather than A/π.

Note that the first three problems can all be solved by brute force over all possible solutions, which requires $n^{\mathcal{O}(k)}$ time when $n = |x|$ and $k = \pi(x)$. The problem perm^{0,1}/apex also admits such an algorithm, as we will see in Chapter 4. In other words, when restricted to instances x with $\pi(x) = \mathcal{O}(1)$, each of these problems admits a polynomial-time algorithm. This does however *not* apply to perm$^{0,1}/\Delta$, since Theorem 1.3 asserts #P-completeness of perm^{0,1} even when restricted to graphs of maximum degree 3. The problems that can be solved in running times similar to $n^{\mathcal{O}(k)}$ give rise to the class XP:[3]

[3] Usually, the class XP and other classes to be defined later are classes of *decision* problems. For the purposes of this thesis, we however define these as classes of counting problems.

Definition 1.6. The class XP contains all parameterized (counting) problems A/π for which the following holds: There exists a computable function $f : \mathbb{N} \to \mathbb{N}$ such that, for all $x \in \{0,1\}^*$, the problem $\mathsf{A}(x)$ can be solved in deterministic time $|x|^{f(\pi(x))}$.

The central question in parameterized complexity is whether running times of the XP-type can be lowered to $f(\pi(x))n^{\mathcal{O}(1)}$ for computable functions f, in which case one speaks of *fixed-parameter tractability*. In other words, can we allow some (possibly extremely slow) growth of $\pi(x)$ and still obtain a polynomial-time algorithm? This is obviously not true for problems like $\mathrm{perm}^{0,1}/\Delta$, since they are not even contained in XP unless FP = #P.

Definition 1.7. A parameterized problem A/π is fixed-parameter tractable (fpt) if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ such that, for all $x \in \{0,1\}^*$, the problem $\mathsf{A}(x)$ can be solved in deterministic time $f(\pi(x))|x|^{\mathcal{O}(1)}$. The class FPT contains all fixed-parameter tractable (counting) problems.

Many classical problems become tractable under suitable parameterizations; this is prominently witnessed by the problem VertexCover of deciding whether a graph G admits a k-vertex-cover: This problem is NP-complete in the unparameterized setting[4], but it was shown in [DF95] that the parameterization by the solution size k renders the problem feasible, that is, VertexCover/k is fixed-parameter tractable. Note that this problem is subsumed by HittingSet/$k + d$ on instances with $d = 2$, and it was actually even shown in [FG04] that the counting version #HittingSet/$k + d$ is fixed-parameter tractable, where d may be larger than 2.

Hardness of parameterized counting problems

However, for other problems, such as #Clique or #HittingSet/k, it seems that no fpt-algorithm can be found, and this actually even applies to the decision versions Clique and HittingSet/k. Note that the parameterization of a problem indeed matters for its complexity: The problem HittingSet/k seems hard, but an algorithm is known for HittingSet/$k + d$.

While parsimonious reductions would allow us to obtain hardness results for counting problems with hard decision versions, we will require more permissive notions, as in Section 1.2.1, to prove hardness for problems with easy decision versions, such as $\mathrm{perm}^{0,1}$/apex. To this end, we use the following definitions, adapted from Definition 1.2 and [FG04].

Definition 1.8. Let A/π and B/π' be parameterized counting problems.

- Let (f, g) be a weakly parsimonious reduction from A to B. We say that (f, g) is a *weakly parsimonious fpt-reduction*, and we write $\mathsf{A}/\pi \leq_{fpt} \mathsf{B}/\pi'$, if the following holds for all $x \in \{0,1\}^*$:

[4] In fact, it is one of the 21 original problems to be shown NP-complete [Kar72].

1. The running times of f and g on x are each bounded by $r(\pi(x))|x|^{\mathcal{O}(1)}$ for some computable function r.
2. We have $\pi'(f(x)) \leq s(\pi(x))$ for some computable function s.

- If (f, g) is a weakly parsimonious fpt-reduction and $g(x) = 1$ holds for all $x \in \{0,1\}^*$, then we call f a *parsimonious fpt-reduction* and write $\mathsf{A}/\pi \leq_{fpt}^{pars} \mathsf{B}/\pi'$.
- Let $\mathbb{T}$ be a Turing reduction from A to B. We call $\mathbb{T}$ a *Turing fpt-reduction*, and write $\mathsf{A}/\pi \leq_{fpt}^{T} \mathsf{B}/\pi'$, if the following holds for all $x \in \{0,1\}^*$:
  1. The running time of $\mathbb{T}$ on x is bounded by $r(\pi(x))|x|^{\mathcal{O}(1)}$ for some computable function r.
  2. Every oracle query y issued by $\mathbb{T}$ on x satisfies $\pi'(y) \leq s(\pi(x))$ for some computable function s.

It is easily verified that FPT is closed under any of the three reduction notions from Definition 1.8, see [FG04]. Furthermore, it is clear that a similar inclusion as in (1.1) holds, namely

$$\mathsf{A}/\pi \leq_{fpt}^{pars} \mathsf{B}/\pi' \Rightarrow \mathsf{A}/\pi \leq_{fpt} \mathsf{B}/\pi' \Rightarrow \mathsf{A}/\pi \leq_{fpt}^{T} \mathsf{B}/\pi'.$$

Using parameterized reductions, we can define complexity classes #W[1] and #W[2] that will capture all of the parameterized counting problems arising in this thesis. For the following definition, we chose to characterize these classes as the closures of the problems #Clique and #HittingSet under parsimonious fpt-reductions. This might be somewhat unsatisfying to a complexity theorist, but it is entirely sufficient for all results shown later. It should be noted that machine-based characterizations of these classes around the notion of W[1]-*programs* are also known [FG06], but these are not relevant for our purposes.

Definition 1.9. We define the following complexity classes:

- Let #W[1] be the set of all problems A/π with $\mathsf{A}/\pi \leq_{fpt}^{pars} \#\mathsf{Clique}/k$.
- Let #W[2] be the set of all problems A/π with $\mathsf{A}/\pi \leq_{fpt}^{pars} \#\mathsf{HittingSet}/k$.

Hardness and completeness for #W[1] and #W[2] are defined as in Section 1.2.1, using the notions of fpt-reductions introduced in Definition 1.8.

Note that counting problems in FPT with integer-valued output are trivially contained in #W[1] and #W[2]. On the other hand, it is a standard assumption of parameterized complexity theory that FPT $\neq$ #W[1] holds. Since the problem $\#\mathsf{Clique}/k$ is #W[1]-complete by definition, this boils down to assuming that $\#\mathsf{Clique}/k$ is not fixed-parameter tractable.

It is also known that #Clique/k is contained in #W[2], which implies #W[1] $\subseteq$ #W[2]. However, #HittingSet/k is assumed not to be contained in #W[1], so this inclusion is believed to be strict.

Examples for parameterized reductions

To obtain further #W[1]-hardness results, the problem #Clique/k can be reduced to other parameterized counting problems, such as its colorful variant #ColClique/k, which we define in the following. Please recall the notion of vertex-colored graphs from Section 1.1.2.

Problem 1.10 (#ColClique/k)**.** Given as input a $[k]$-vertex-colored graph G, for $k \in \mathbb{N}$, count the colorful subsets $K \subseteq V(G)$ that are cliques in G. Here, the parameter is k.

Lemma 1.11. *We have* #Clique/$k \leq_{fpt}^{pars}$ #ColClique/k.

Proof. Let G be an uncolored graph, with $V(G) = [n]$, whose k-cliques we wish to count, for a given input $k \in \mathbb{N}$. To this end, we create a $[k]$-vertex-colored graph G' on vertices $V(G') = [n] \times [k]$, where vertices in $(\star, i)$ for $i \in [k]$ are assigned the color i. For all $u, v \in [n]$ and $i, j \in [k]$ with $u < v$ and $i < j$ and $uv \in E(G)$, add an edge between the vertices (u, i) and (v, j) in G'. We claim that the set $\mathcal{K}$ of colorful k-cliques in G' stands in bijection with the set of k-tuples

$$\mathcal{S} = \{(u_1, \ldots, u_k) \in [n]^k \mid \forall 1 \leq i < j \leq k : \; u_i < u_j \wedge u_i u_j \in E(G)\}.$$

These in turn clearly correspond to the k-cliques of the uncolored graph G: Every uncolored k-clique of G appears as a sorted tuple in $\mathcal{S}$. This implies that the mapping $G \mapsto G'$ is a parsimonious fpt-reduction from #Clique/k to #ColClique/k.

To see that $\mathcal{S} \simeq \mathcal{K}$ holds, we define a bijection $C : \mathcal{S} \to \mathcal{K}$. For $S \in \mathcal{S}$ with $S = (u_1, \ldots, u_k)$, let $C(S) := \{(u_i, i) \mid i \in [k]\}$, and observe that $C(S) \in \mathcal{K}$: It is clear that $C(S)$ is colorful, and for all $1 \leq i < j \leq k$, the edge between (u_i, i) and (u_j, j) is present in G' since $u_i u_j \in E(G)$ and $i < j$. Since $C(S) \neq C(S')$ holds for ordered k-tuples $S \neq S'$, it follows that C is injective.

To see surjectivity of C, let $K \in \mathcal{K}$. Then $K = \{(u_i, i) \mid i \in [k]\}$ with $u_1, \ldots, u_k \in [n]$, and the construction of G' implies that $u_i < u_j$ and $u_i u_j \in E(G)$ hold for $i < j$. Hence, we can write $K = C(S)$ for $S = (u_1, \ldots, u_k)$ with $S \in \mathcal{S}$. □

It is easily verified that #ColClique is contained in #W[1], hence it follows that the problem is #W[1]-complete. Note also that the reduction from Lemma 1.11 does not exploit the full power of parsimonious fpt-reductions: Firstly, the parameter was not increased at all, and secondly, the reduction can be computed in polynomial time.

We will use #ColClique as a reduction source for other parameterized counting problems, such as counting grid tilings, whose decision version was introduced in [Mar12]. This problem is particularly useful for proving #W[1]-hardness of problems on planar structures, as seen in [MP14, Mar12]. Please recall the notions of horizontal/vertical adjacency from Section 1.1.

Problem 1.12 (#GridTiling)**.** Given $n, k \in \mathbb{N}$ and a function $\mathcal{T} : [k]^2 \to 2^{[n]^2}$, we wish to count the *consistent grid tilings* of $\mathcal{T}$, that is, assignments $a : [k]^2 \to [n]^2$ which satisfy the following properties:

(H) For horizontally adjacent indices $\kappa, \kappa' \in [k]^2$, the first components of $a(\kappa)$ and $a(\kappa')$ agree.

(V) For vertically adjacent indices $\kappa, \kappa' \in [k]^2$, the second components of $a(\kappa)$ and $a(\kappa')$ agree.

(C) For all $\kappa \in [k]^2$, we have $a(\kappa) \in \mathcal{T}(\kappa)$.

In the following, we show that #GridTiling is #W[1]-hard under parsimonious fpt-reductions. We also show that, for each instance $\mathcal{T}$, we may assume $\mathcal{T}$ to be balanced in a certain way, and this will prove useful later.

Lemma 1.13. *We have* $\#\mathsf{ColClique}/k \leq_{fpt}^{pars} \#\mathsf{GridTiling}/k$*, even when each instance* $(n, k, \mathcal{T})$ *to the second problem is given with some* $T \in \mathbb{N}$ *such that the following balance property holds: For all* $\kappa \in [k]^2$ *and* $v \in [n]$*, we have* $|\mathcal{T}(\kappa) \cap (\star, v)| = T$*.*

Proof. We assume $k > 1$ without limitation of generality. Let G be a $[k]$-vertex-colored graph on vertices $[n]$. We replace each edge $uv \in E(G)$ by the directed edges uv and vu, then we add all self-loops to G to obtain a digraph G'. The colorful k-cliques in G stand in bijection with the colorful K-copies in G', where K is the complete digraph on k vertices, including self-loops.

For $i, j \in [k]$, write $E_{i,j} = E_{i,j}(G')$ for the set of directed edges in G' from i-colored to j-colored vertices. Note that $E_{i,j} \subseteq [n]^2$; we use this to define an instance $\mathcal{T}$ to #GridTiling by declaring $\mathcal{T}(i, j) = E_{i,j}$ for all $i, j \in [k]$.

We claim that the consistent grid tilings of $\mathcal{T}$ bijectively correspond to the colorful K-copies in G'. To see this, note that due to property (C), every consistent assignment $a : [k]^2 \to [n]^2$ encodes an edge-subset $S_a \subseteq E(G')$ that picks, for each $i, j \in [k]$, exactly one element from $E_{i,j}$. This implies $|S_a| = k^2$. Hence, if the edges in S_a are incident with exactly k distinct vertices, then S_a corresponds to a colorful K-copy in G'. By properties (H) and (V) of a consistent grid tiling, the edge set S_a contains exactly k distinct endpoints and k distinct starting points. But since $E_{i,i}$ for $i \in [k]$ contains only self-loops, the sets of endpoints and starting points of edges in S_a are identical, which implies that S_a is indeed a K-copy in G'.

In the remainder of the proof, we ensure the balance property. To this end, define

$$\begin{aligned} T_{\kappa,v} &:= |\mathcal{T}(\kappa) \cap (\star, v)| \quad \text{for } \kappa \in [k]^2 \text{ and } v \in [n], \\ T &:= \max_{\kappa \in [k]^2,\, v \in [n]} T_{\kappa,v}, \end{aligned}$$

and let $n' := n + k^2T$. We modify $\mathcal{T}$ to an instance $\mathcal{T}' : [k]^2 \to 2^{[n']^2}$ that fulfills the statement of the lemma. To this end, consider $[n']$ to be partitioned into $[n]$ and k^2 consecutive "dummy" blocks B_κ for $\kappa \in [k]^2$, where $|B_\kappa| = T$. For $\kappa \in [k]^2$ and $v \in [n]$, add $T - T_{\kappa,v}$ arbitrary distinct "dummy" elements from the set $\{(f, v) \mid f \in B_\kappa\}$ to $\mathcal{T}(\kappa)$ in order to obtain $\mathcal{T}'(\kappa)$.

This ensures the balance property, and we observe that $\mathcal{T}'$ has the same consistent grid tilings as $\mathcal{T}$: Every consistent grid tiling of $\mathcal{T}$ is also a consistent grid tiling of $\mathcal{T}'$. Furthermore, dummy elements cannot be part of any consistent grid tiling of $\mathcal{T}'$: This is because for all κ and κ', the dummy elements in $\mathcal{T}'(\kappa)$ and $\mathcal{T}'(\kappa')$ have disjoint first coordinates, which are also distinct from $[n]$. Thus, in particular, any assignment using dummy elements cannot satisfy (H). □

Parameterized modular counting

Finally, we define modular parameterized counting classes $\mathsf{Mod_tW[1]}$ for $t \in \mathbb{N}$, which are analogous to the modular counting classes $\mathsf{Mod_tP}$ from Section 1.2.1. Such classes have also been studied in [BDH15].

Definition 1.14. For fixed $t \in \mathbb{N}$, let $\mathsf{Mod_tClique}/k$ denote the problem of deciding on input (G, k) whether the number of k-cliques in G is not divisible by t.

Let $\mathsf{Mod_tW[1]}$ denote the set of problems that can be reduced to $\mathsf{Mod_tClique}/k$ under parsimonious fpt-reductions.[5] We write $\oplus\mathsf{W[1]}$ for $\mathsf{Mod_2W[1]}$.

Analogously to the inclusion of NP in $\mathsf{Mod_tP}$ via the Valiant-Vazirani theorem mentioned in Section 1.2.1, it was shown in [BDH15] that the parameterized decision problem Clique/k admits a randomized fpt-reduction to $\oplus\mathsf{Clique}/k$. Such reductions are defined like the randomized reductions from the previous section, with the exception that they may run in time $f(k)n^{\mathcal{O}(1)}$, but must map instances with parameter k to instances with parameter $g(k)$, for computable functions f and g. We can hence conclude that $\mathsf{W[1]}$, the closure of Clique/k under fpt-reductions, is contained in $\oplus\mathsf{W[1]}$ under randomized fpt-reductions.

1.2.3. Exponential-time complexity

A modern branch of complexity theory, launched in [IPZ01, IP01], focuses on conditional quantitative lower bounds for computational problems under assumptions that are stronger than $\mathsf{P} \neq \mathsf{NP}$, $\mathsf{FP} \neq \#\mathsf{P}$ or $\mathsf{FPT} \neq \#\mathsf{W[1]}$.[6] The most popular and widely-believed assumption in exponential-time complexity is the *exponential-time hypothesis* ETH, which postulates the following:

ETH: The decision problem 3-SAT on formulas with n variables cannot be solved in time $2^{o(n)}$.

[5] Note that these reductions also yield a reduction notion for decision problems.
[6] For a more extensive introduction into this field, please consider [LMS11].

This hypothesis also supports a counting version [DHM+14], namely #ETH, which asserts the same statement for 3-#SAT and is a priori weaker than ETH.

#ETH: The counting problem 3-#SAT on formulas with n variables cannot be solved in time $2^{o(n)}$.

If we assume ETH, then its lower bound for 3-SAT can be transferred to other problems, such as the problem HamCycle of deciding the existence of a Hamiltonian cycle in a graph G: The classical reductions from 3-SAT to HamCycle, as in [GJ79, Pap94], map 3-CNF formulas φ on n variables and m clauses to graphs $G_{ham}(\varphi)$ on $\mathcal{O}(n+m)$ vertices and edges, where each variable and each clause is represented by some gadget of size $\mathcal{O}(1)$. Since we may always assume $m = \mathcal{O}(n^3)$, because duplicated clauses in φ are irrelevant, the number of vertices in $G_{ham}(\varphi)$ is also bounded by $\mathcal{O}(n^3)$. Hence, an algorithm with running time $2^{o(n^{1/3})}$ for HamCycle on n-vertex graphs would imply a $2^{o(n)}$ time algorithm for 3-SAT and hence refute ETH.

However, this lower bound seems far from tight, since even an algorithm for HamCycle with running time $2^{o(n)}$ on n-vertex graphs is not known, and would in fact be considered a striking and unlikely result. To improve the lower bound in the setting described above, we would need to map formulas φ with n vertices and n^3 clauses to graphs $G_{ham}(\varphi)$ on $\mathcal{O}(n)$ vertices, and it is unclear how to achieve this. If we may however assume that φ has only $\mathcal{O}(n)$ clauses along with its n variables, then the reduction sketched above would yield an asymptotically tight lower bound.

Subexponential reductions

To enable this assumption of sparsity on φ, we use the relaxed reduction notion of *subexponential Turing reduction families* [IPZ01]. To define these, it is useful to consider the involved problems as parameterized problems, such as SAT/n and HamCycle/n, where n refers to the number of variables (or vertices) of the instances.

Definition 1.15 ([IPZ01])**.** A *subexponential Turing reduction family* between parameterized problems A/π and B/π' is an algorithm $\mathbb{T}$ with oracle access to B whose inputs are pairs (x, ϵ), where x is an instance for A and ϵ with $0 < \epsilon \leq 1$ is a running time parameter. Furthermore, there are computable functions $f, g : \mathbb{Q} \to \mathbb{N}$ such that the following holds on input (x, ϵ):

1. $\mathbb{T}$ computes $\mathsf{A}(x)$ in time $f(\epsilon) \cdot 2^{\epsilon \cdot \pi(x)} |x|^{\mathcal{O}(1)}$, and
2. whenever $\mathbb{T}$ invokes the oracle for B on a query y, then $\pi'(y) \leq g(\epsilon) \cdot \pi(x)$.

We write $\mathsf{A}/\pi \leq_{serf} \mathsf{B}/\pi'$ if such a reduction exists, where "serf" abbreviates **s**ub**e**xponential **r**eduction **f**amily.

In a subexponential Turing reduction family, the exponential part of the running time can be chosen as $2^{\epsilon \cdot \pi(x)}$ for arbitrarily small ϵ, and intuitively, also for arbitrarily slowly decreasing $\epsilon = o(1)$. This way, we can ensure a running time of $2^{o(\pi(x))}$, which however comes at the cost of incurring a multiplicative factor of $g(\epsilon)$ in the parameter. Such reductions can be observed to preserve lower bounds as expected:

Lemma 1.16 ([IPZ01])**.** *If* $\mathsf{A}/\pi \leq_{serf} \mathsf{B}/\pi'$ *and* B *can be solved in time* $2^{o(\pi'(x))}|x|^{\mathcal{O}(1)}$*, then* A *can be solved in time* $2^{o(\pi(x))}|x|^{\mathcal{O}(1)}$*.*

Under this notion of reduction, it can be shown that instances to satisfiability problems may be assumed to have only $\mathcal{O}(n)$ clauses, provided that we consider the problems d-SAT and d-#SAT for *fixed* clause-width $d \in \mathbb{N}$, rather than their versions SAT and #SAT on unbounded clause-width. For the decision version, this was shown in [IPZ01], and for the counting version in [DHM$^+$14].

Theorem 1.17. *For every fixed* $d \geq 3$*, the problem* d-SAT/n *admits a subexponential Turing reduction family to* d-SAT/m*. The same applies from* d-#SAT/n *to* d-#SAT/m*. Here,* n *and* m *denote the numbers of variables and clauses of a formula, respectively.*

Corollary 1.18. *If* 3-SAT/m *could be solved in time* $2^{o(m)}n^{\mathcal{O}(1)}$*, then Lemma 1.16 and Theorem 1.17 imply that* 3-SAT/n *could be solved in time* $2^{o(n)}$*, which would refute* ETH*. Likewise, an algorithm for the counting problem* 3-#SAT/m *with running time* $2^{o(m)}n^{\mathcal{O}(1)}$ *would refute* #ETH*.*

By reductions, this implies tight lower bounds under ETH for many problems, including the problem $\mathsf{HamCycle}/n$ that we already considered at the beginning of this subsection: Given a 3-CNF formula φ on t variables and m clauses, note first that we can always assume $t = \mathcal{O}(m)$. Since the graph $G_{ham}(\varphi)$ described in the beginning of this subsection has $\mathcal{O}(t + m) = \mathcal{O}(m)$ vertices, an algorithm for $\mathsf{HamCycle}$ with running time $2^{o(n)}$ on n-vertex graphs would imply a $2^{o(m)}$ time algorithm for 3-SAT/m and hence refute ETH.

Note that the classical reduction from 3-SAT/m to $\mathsf{HamCycle}/n$ via the graph $G_{ham}(\varphi)$ is in fact a subexponential Turing reduction family as well: Since this reduction runs in polynomial time, it trivially satisfies property (1) of Definition 1.15 on all inputs ϵ. Furthermore, since its images have cm vertices for some fixed $c \in \mathbb{N}$ that is independent of the input, it also satisfies (2) with $g(\epsilon) = c$. Such constant bounds on the parameter blowup allow for transfers of lower bounds, and they can be used to define a "light" version of subexponential Turing reduction families. This simplified and more restrictive notion will suffice for almost all applications in this thesis, with the exception of Chapter 8.

Definition 1.19. For $c \in \mathbb{Q}$, a reduction $\mathbb{T}$ from A/π to B/π' *incurs blowup* c if the following holds: On every instance x to A, all reduction images y invoked by $\mathbb{T}$ satisfy $\pi'(y) \leq cy$.

We say that a reduction *incurs linear blowup* if it incurs blowup $c = \mathcal{O}(1)$, and we write $\leq_p^{lin}$ for polynomial-time reductions that incur linear blowup, and $\leq_{fpt}^{lin}$ for fpt-reductions that incur linear blowup.

Lower bounds for #W[1]-hard problems

The assumptions ETH and #ETH in exponential-time complexity are also known to imply statements in "classical" parameterized complexity. For instance, the following theorem shows that ETH implies FPT $\neq$ W[1], and even more so, it establishes an asymptotically tight lower bound on the time needed to solve Clique/k. Recall that Clique/k can be solved in time $n^{k+\mathcal{O}(1)}$ by brute-force.

Theorem 1.20 ([CCF+05, CHKX06])**.** *Assuming* ETH*, the problem* Clique/k *admits no* $f(k)n^{o(k)}$ *time algorithm for any computable function* f*. The same holds for* #Clique/k *under* #ETH*.*

Using $\leq_{fpt}^{lin}$, the parameterized version of linear-blowup reductions, we can transfer such lower bounds to other problems.

Example 1.21. The reduction from #Clique to #ColClique in Lemma 1.11 actually did not increase the parameter at all, so we obtain that #ColClique admits no $f(k)n^{o(k)}$ algorithm under #ETH.

1.3. Graph polynomials

In this section, we consider *graph polynomials*, which are functions p that map graphs G to some element $p(G) \in \mathfrak{O}$ from a polynomial ring $\mathfrak{O}$.[7] They are similar to the *generating functions* used in enumerative combinatorics, and as such, they provide an elegant bridge between combinatorics, graph theory and algebra. Graph polynomials are usually defined such that isomorphic graphs G and G' are required to satisfy $p(G) = p(G')$, but we will ignore this restriction for more flexibility. Furthermore, as a notational convention, we write $p(G;\xi) = (p(G))(\xi)$ for the evaluation of $p(G)$ at ξ, and we write $\mathcal{G}$ for the class of all graphs.

A prominent graph polynomial is the *chromatic polynomial* $\chi : \mathcal{G} \to \mathbb{Z}[x]$, which counts, for all graphs G and integers $k \in \mathbb{N}$, the *proper* vertex-colorings of G with k colors at its evaluation point $\chi(G;k)$. Here, a coloring $c : V(G) \to [k]$ is called *proper* iff $c(u) \neq c(v)$ holds for all $uv \in E(G)$. It can be verified that $\chi(G)$ is indeed a polynomial for all graphs G, and it should be mentioned that its existence in the mathematical literature is due to an attempt by Birkhoff [Bir12] to prove the Four Color Theorem: Using algebraic methods, he tried to show that $\chi(G;4) \neq 0$ holds for all planar graphs G.

The present thesis applies graph polynomials in a more profane way: Firstly, we use them as notational items in arguments that would otherwise require opaque coefficient-wise formulations. Secondly, we require them in hardness proofs to enable the method of *polynomial interpolation*, which will be introduced in Section 1.4. More general and more respectful introductions to graph polynomials can be found in [Big94, Mak06].

For any graph polynomial p, we define a coefficient problem Coeff(p), an evaluation problem Eval(p), and an evaluation problem Eval$_S$(p) at points with entries from a set S.

[7] In particular, graph polynomials are not actually polynomials, but rather *functions* from graphs *to* polynomials.

Definition 1.22. For any graph polynomial p, define the following problems:

$\mathsf{Coeff}(p)$: We wish to compute all coefficients of $p(G)$ when given as input a graph G.

$\mathsf{Eval}(p)$: We obtain as input a graph G and $\xi \in \mathbb{Q}$, and we wish to evaluate $p(G;\xi)$.

$\mathsf{Eval}_S(p)$: This is in fact a family of problems, each defined for a *fixed* choice of $S \subseteq \mathbb{Q}$. On input G and a tuple ξ whose entries are all from S, evaluate $p(G;\xi)$. If p is univariate and $S = \{a\}$, this simply asks to compute $p(G;a)$ for fixed $a \in \mathbb{Q}$, and we write $\mathsf{Eval}_a(p)$ in this case.

In the following, we introduce the matching polynomials μ and M, the independent set polynomial I, the perfect matching polynomial PerfMatch, the permanent perm, and the Tutte polynomial T together with its random-cluster formulation Z.

1.3.1. Matching polynomials and MatchSum

Let $\mathcal{M}[G]$ denote the set of matchings in G, and recall from Section 1.1 that the unmatched vertices of $M \in \mathcal{M}[G]$ are denoted by $\mathrm{usat}(G, M)$, or simply $\mathrm{usat}(M)$. With these definitions, we formulate the *edge-generating matching polynomial* M and the *defect-generating matching polynomial* μ, using slightly modified definitions from [Far79].

Definition 1.23. Let $\mathcal{G}$ denote the set of undirected finite simple graphs, let x be an indeterminate, and let the *matching polynomials* M and μ of type $\mathcal{G} \to \mathbb{Z}[x]$ be defined as

$$M(G) := \sum_{M \in \mathcal{M}[G]} x^{|M|}, \tag{1.2}$$

$$\mu(G) := \sum_{M \in \mathcal{M}[G]} x^{|\mathrm{usat}(M)|}. \tag{1.3}$$

The sums used to define $M(G)$ and $\mu(G)$ both range over the matchings of G, but they assign weights to the matchings that can be considered dual. We observe that

$$\mu(G;x) = \sum_{M \in \mathcal{M}[G]} x^{n-2|M|} = x^n \cdot M(G;x^{-2}), \tag{1.4}$$

and depending on the application, we will choose the polynomial that suits us better.

The complexity of the polynomials μ and M

The complexity of evaluating μ and M has already been investigated in the literature. For instance, it was shown as part of [CLX10, Theorem 6.1] that the problem $\mathsf{Eval}_\xi(\mu)$

of evaluating $\mu(G;\xi)$ is #P-complete for all fixed $\xi \in \mathbb{C} \setminus \{0\}$, even on planar graphs of maximum degree 3. Note that only G is the input in this problem, whereas ξ is fixed.

Theorem 1.24 ([CLX10])**.** *For all fixed $\xi \in \mathbb{Q} \setminus \{0\}$: The problem $\mathsf{Eval}_\xi(\mu)$ of evaluating $\mu(G;\xi)$ is #P-complete, even on planar bipartite graphs G of maximum degree 3.*

Note that the evaluation $\mu(G;0)$ counts the perfect matchings of G. We will later see that this admits a polynomial-time algorithm on planar graphs, whereas it is #P-hard on general graphs. In Section 4.2, we additionally show #W[1]-hardness of computing the *coefficient* of x^k in $\mu(G)$ for planar graphs when parameterized by k. In other words, we show that $\mathsf{Coeff}(\mu)$ is #W[1]-hard on planar graphs. Note that the coefficient of x^k in $\mu(G)$ counts the k-defect matchings in G. This is complemented in Section 5.2.2 by a #W[1]-completeness proof of computing the coefficient of x^k in the dual polynomial $M(G)$ on *general* graphs G, which amounts to counting k-matchings in general graphs.

A multivariate generalization of μ

The polynomials M and μ can also be generalized to multivariate versions, and this will be useful in several occasions throughout the thesis. For μ, this generalization yields a multivariate graph polynomial that is called MatchSum in [Val08].

Definition 1.25. Let $\mathcal{V}$ denote the set of all vertices of graphs, and let $\mathcal{X}_\mathcal{V} = \{x_v\}_{v\in\mathcal{V}}$ be a family of indeterminates. (We may assume for simplicity that $\mathcal{V} = \mathbb{N}$.) Then $\mathrm{MatchSum} : \mathcal{G} \to \mathbb{Z}[\mathcal{X}_\mathcal{V}]$ is defined by

$$\mathrm{MatchSum}(G) = \sum_{M\in\mathcal{M}[G]} \prod_{v\in\mathrm{usat}(G,M)} x_v.$$

We only consider finite graphs G, so $\mathrm{MatchSum}(G)$ is indeed a polynomial.

For every graph G, the multivariate polynomial $\mathrm{MatchSum}(G)$ specializes to its univariate version $\mu(G)$ when the substitution $x_v \leftarrow x$ is carried out for all $v \in V(G)$. In most applications, we will use MatchSum on vertex-weighted graphs G with weight functions $w : V(G) \to \mathbb{Q}$. Then the evaluation of MatchSum on G is the result of substituting $x_v \leftarrow w(v)$ for all $v \in V(G)$, and hence, some rational number itself. In other words, if G is a vertex-weighted graph, then we simply write $\mathrm{MatchSum}(G)$, which is simpler than writing $\mathrm{MatchSum}(G';\xi)$ for an unweighted version G' of G and a tuple ξ that encodes the vertex-weights of G. We will revisit such multivariate generalizations in Chapter 8.

The independent set polynomial

The *edge-generating* matching polynomial stands in direct correspondence with the so-called *independent set polynomial* $I : \mathcal{G} \to \mathbb{Z}[x]$, also known as the hard-core gas model[8]

[8] More specifically, as the *"partition function of the lattice gas with hard-core self-repulsion and hard-core pair interaction"* [SS05].

in statistical physics [SS05]: If $\mathcal{I}[G]$ denotes the independent sets of G, then this graph polynomial is defined by

$$I(G) := \sum_{S \in \mathcal{I}[G]} x^{|S|}. \tag{1.5}$$

Let $L(G)$ denote the line graph of G, that is, the graph $L(G) = (E(G), F)$ where $ef \in F$ for $e, f \in E(G)$ holds iff $e \cap f \neq \emptyset$. Then the k-matchings of G stand in bijection with the independent k-sets in $L(G)$, which implies by comparing (1.2) and (1.5) that

$$M(G) = I(L(G)). \tag{1.6}$$

1.3.2. PerfMatch and the permanent

Let $\mathcal{PM}[G] \subseteq \mathcal{M}[G]$ denote the set of perfect matchings in G, that is, the 0-defect matchings of G. We consider two graph polynomials that are related to counting perfect matchings, and which play a central role in this thesis.

Definition 1.26. Let $\mathcal{E}$ denote the set of all edges of all graphs and let $\mathcal{X}_{\mathcal{E}} = \{x_e\}_{e \in \mathcal{E}}$ be a family of indeterminates. (We may assume for simplicity that $\mathcal{E} = \mathbb{N}^2$.) Then we define $\mathrm{PerfMatch} : \mathcal{G} \to \mathbb{Z}[\mathcal{X}_{\mathcal{E}}]$ as

$$\mathrm{PerfMatch}(G) = \sum_{M \in \mathcal{PM}[G]} \prod_{e \in M} x_e. \tag{1.7}$$

If G is a bipartite graph, we may also speak of the permanent $\mathrm{perm}(G)$, defined by

$$\mathrm{perm}(G) = \mathrm{PerfMatch}(G).$$

The function perm is usually defined on the bi-adjacency matrix A of G: This matrix is defined as follows: If G has a (fixed) bipartition into $\{\ell_1, \ldots, \ell_n\}$ and $\{r_1, \ldots, r_n\}$, then

$$A_{i,j} = \begin{cases} x_e & \text{if } \ell_i r_j \in E(G), \\ 0 & \text{otherwise.} \end{cases}$$

We also write $\mathrm{perm}(A)$ if A is the bi-adjacency matrix of G.

As for MatchSum, we consider $\mathrm{PerfMatch}(G)$ on edge-weighted graphs G with weight functions $w : E(G) \to \mathbb{Q}$ by substituting $x_e \leftarrow w(e)$ for all $e \in E(G)$. For unweighted graphs G, where every edge has weight 1, the quantity $\mathrm{PerfMatch}(G)$ plainly counts the perfect matchings of G. Note that algorithms for evaluating PerfMatch are stronger than those for perm, and hardness results for perm are stronger than those for PerfMatch.

The polynomial PerfMatch will be of central importance for the concept of matchgates, which we introduce in Chapter 2. Furthermore, the material in Parts I and III is largely concerned with the computational problem of evaluating PerfMatch.

Remark. We could define PerfMatch and perm such that these polynomials receive only *complete* graphs as arguments, and we could then reduce the case of *general* graphs G to this by assigning weight 0 to the non-edges of G. This way however, we lose the capacity to speak naturally of properties of the argument graph G such as its planarity.

Permanent versus determinant

If G is an edge-weighted bipartite graph with bi-adjacency matrix $A \in \mathbb{Q}^{n\times n}$, then it can be checked easily that

$$\operatorname{perm}(A) = \sum_{\sigma\in S_n} \prod_{i\in[n]} A_{i,\sigma(i)}, \tag{1.8}$$

where S_n denotes the set of permutations on $[n]$. Note that, apart from the missing factor $\operatorname{sgn}(\sigma)$, this parallels the Leibniz expansion of the determinant

$$\det(A) = \sum_{\sigma\in S_n} \operatorname{sgn}(\sigma) \prod_{i\in[n]} A_{i,\sigma(i)}. \tag{1.9}$$

In fact, a central task of *algebraic complexity theory* is to find "useful" differences between perm and det, which could ultimately be used to prove $\mathsf{VP} \neq \mathsf{VNP}$, an algebraic variant of $\mathsf{P} \neq \mathsf{NP}$, see [Agr06].

By Gaussian elimination, $\det(A)$ can be computed in time $\mathcal{O}(n^3)$, and this can be improved to $\mathcal{O}(n^\omega)$, where $\omega \leq 2.38$ denotes the exponent of matrix multiplication, as noted in [BH74]. However, a similar result is not expected for $\operatorname{perm}(A)$, as we recall from Theorem 1.3 that the evaluation of perm is #P-complete, even on *unweighted* graphs.

Algorithmic results for PerfMatch

Despite the #P-hardness result for the permanent, there are nontrivial algorithms for evaluating PerfMatch that improve upon the naive running time of $\mathcal{O}(n! \cdot n)$ suggested by (1.8) or the running time $\mathcal{O}(2^{|E(G)|} \cdot n)$ suggested by (1.7). For instance, Ryser's formula [Rys63] establishes that

$$\operatorname{perm}(A) = (-1)^n \sum_{S\subseteq[n]} (-1)^{|S|} \prod_{i=1}^{n} \sum_{j\in S} A_{i,j},$$

so we can evaluate $\operatorname{perm}(A)$ in time $\mathcal{O}(2^n \cdot n^2)$ on an $n \times n$ matrix A. Note that this implies an $\mathcal{O}(2^{n/2} \cdot n^2)$ time algorithm for computing $\operatorname{PerfMatch}(G)$ when G is a bipartite graph on n vertices. This was generalized in [Bjö12] to an algorithm with the same running time that does not require G to be bipartite.

Furthermore, if we are willing to admit some relaxations of the problem, we can even obtain polynomial-time algorithms. For instance, the celebrated approximation algorithm from [JSV04] relaxes the evaluation of permanents to *randomized approximate* evaluation: On matrices A with non-negative entries, an approximation with multiplicative factor $(1+\epsilon)$ to $\operatorname{perm}(A)$ can be found with high probability, and in time $\operatorname{poly}(n, 1/\epsilon)$. Unless

$\mathsf{RP} = \mathsf{NP}$, this cannot be extended to matrices with *negative* edge-weights, but it is open whether an approximation scheme is possible for *general* graphs, and thus for PerfMatch, on input graphs that feature only non-negative edge weights.

Even concerning exact evaluation, there are surprisingly large classes on which PerfMatch admits polynomial-time algorithms. This includes the planar graphs, as was shown by researchers in the area of statistical physics during the 1960s with an algorithm that predates Valiant's #P-hardness result for the permanent.

Theorem 1.27 ([Kas61, TF61, Kas67])**.** *For planar edge-weighted graphs G, the value* $\mathrm{PerfMatch}(G)$ *can be computed in time* $\mathcal{O}(n^{1.5})$*.*

Proof sketch. Given the graph G as input, it is possible to compute a *Pfaffian orientation* of G in linear time. This is an edge subset $S \subseteq E(G)$ such that the following holds: After flipping the sign of $w(e)$ for each edge $e \in S$, the resulting graph G', with adjacency matrix A', satisfies $(\mathrm{PerfMatch}(G))^2 = \det(A')$. Since A' is the adjacency matrix of a planar graph, an algorithm in [LRT79], noted also in [Val08], allows to compute $\det(A')$ in time $\mathcal{O}(n^{1.5})$ rather than $\mathcal{O}(n^{\omega})$. □

It was shown in [GL98, Tes00] that Theorem 1.27 can be extended to an fpt-algorithm for PerfMatch when parameterized by the genus of the input graph. Here, the genus $\gamma(G)$ of a graph G is the minimum genus of a surface that G can be drawn on, see [Die12]. For instance, planar graphs have genus 0, and graphs that can be embedded on a torus have genus 1. This extended algorithm also relies on Pfaffian orientations, and it proceeds by expressing $\mathrm{PerfMatch}(G)$ as a linear combination of $4^{\gamma(G)}$ determinants.

Theorem 1.28 ([GL98, Tes00])**.** *On graphs G with n vertices and genus γ, the value* $\mathrm{PerfMatch}(G)$ *can be computed in time* $\mathcal{O}(4^{\gamma} n^{\omega})$*.*

In Part I, we will present some additional algorithmic results for PerfMatch, some of which extend Theorem 1.28.

Simulating weights

By a simple argument, we can observe that the evaluation of PerfMatch on graphs with positive integer edge-weights can be reduced to the unweighted case. As a notational convention, given a set X, let us abbreviate

$$\mathrm{PerfMatch}^X := \mathsf{Eval}_X(\mathrm{PerfMatch}),$$

and recall that this denotes the problem of evaluating $\mathrm{PerfMatch}(G)$ on edge-weighted graphs G with weight functions $w : E(G) \to X$.

Given an edge-weighted graph G containing an edge $e \in E(G)$ of weight $t \in \mathbb{N}$, we write $G_{t \cdot e}$ for the graph obtained from G by replacing e by t parallel copies of e, each of weight

1. Then it can be verified easily that

$$\text{PerfMatch}(G) = \text{PerfMatch}(G_{t\cdot e}). \tag{1.10}$$

For $t \geq 2$, the graph $G_{t\cdot e}$ is however not simple, which may interfere with our goal to prove hardness results for simple graphs. To circumvent this, let us observe that any edge uv of weight 1 in G may be subdivided twice, as shown below.

If G' denotes the graph obtained from G by this operation, then we observe that G' is bipartite if G is. More importantly, we also have

$$\text{PerfMatch}(G) = \text{PerfMatch}(G'). \tag{1.11}$$

To see this, note that every perfect matching $M \in \mathcal{PM}[G']$ corresponds bijectively to a perfect matching $N \in \mathcal{PM}[G]$, because either

$e^* \in M$, and then M corresponds to some $N \in \mathcal{PM}[G]$ with $uv \notin N$, or

$e^* \notin M$, which implies that both black edges intersecting e^* are present in M, so M corresponds to some $N \in \mathcal{PM}[G]$ with $uv \in N$.

In fact, this simple gadget was a first example of a *matchgate*, a concept we will introduce formally in Section 2.2, and which will be of *major* importance to this thesis. By combining (1.10) and (1.11), we conclude the following lemma:

Lemma 1.29 (folklore)**.** *Let G be an edge-weighted graph on n vertices and m edges, with weight function $w : E(G) \to \mathbb{N}$, and let $T = \max_{e\in E(G)} w(e)$. Then we can compute an unweighted simple graph G' on $\mathcal{O}(n + Tm)$ vertices and edges such that*

$$\text{PerfMatch}(G) = \text{PerfMatch}(G').$$

Furthermore, if G is bipartite, then so is G'.

We will observe later that negative edge-weights can be simulated as well, and this will be revisited in full depth in Chapter 7. Fractional edge-weights can be simulated as well, as we show in the following remark.

Remark 1.30. Consider a graph G with edge-weights $w : E(G) \to \mathbb{Q}$ and let q denote the lowest common denominator of the weights appearing in G. Then we can obtain a graph G' with weights $w' : E(G) \to \mathbb{Z}$ by defining, for each $e \in E(G)$,

$$w'(e) = q \cdot w(e).$$

It is then clear that

$$\begin{aligned}\mathrm{PerfMatch}(G) &= \sum_{M\in\mathcal{PM}[G]}\prod_{e\in M} w(e)\\ &= \sum_{M\in\mathcal{PM}[G']}\prod_{e\in M}\frac{w'(e)}{q}\\ &= q^{-|V(G)|/2}\cdot\mathrm{PerfMatch}(G').\end{aligned}$$

Finally, we can also ensure that the only negative edge-weight appearing in G is -1: If $e = uv$ is an edge with negative weight $w(e) \in \mathbb{Z}$, we can subdivide e twice to obtain subdivision vertices s_1 and s_2 as on the facing page. Then assign weight $|w(e)|$ to the edge us_1, assign weight -1 to the edge s_2v, and weight 1 to the edge s_1s_2.

Evaluation modulo powers of two

It should also be noted that for all integer-valued matrices A, the quantities $\mathrm{perm}(A)$ and $\det(A)$ are equivalent modulo 2, as can be seen trivially from (1.8) and (1.9). This yields a polynomial-time algorithm for computing the parity of $\mathrm{perm}(A)$, and in fact, we can also compute the parity of $\mathrm{PerfMatch}(G)$ for *general* graphs G in polynomial time by a simple argument. For bipartite graphs, the following generalization of the above-mentioned observation was shown [Val79b]:

Theorem 1.31 ([Val79b])**.** *Given an integer-valued matrix $A \in \mathbb{Z}^{n\times n}$ and $k \in \mathbb{N}$, the quantity* $\mathrm{perm}(A) \bmod 2^k$ *can be evaluated in time $\mathcal{O}(n^{4k-3})$.*

1.3.3. The Tutte polynomial

As a last example of a graph polynomial, we consider the *Tutte polynomial*. Since this polynomial will only be used in Chapter 8, we limit its treatment to a bare minimum. Along with the Tutte polynomial, we define the *random-cluster model*, which can be seen to be equivalent to the Tutte polynomial after a substitution of variables, similar to the equivalence of the matching polynomials M and μ we have already observed in (1.4).

Definition 1.32. Let $G = (V, E)$ be an undirected graph and let x, y be indeterminates. For $A \subseteq E$, let $k(A)$ denote the number of connected components of the graph (V, A), including isolated vertices. Then the *Tutte polynomial* $T(G; x, y)$ is defined as

$$T(G;x,y) = \sum_{A\subseteq E}(x-1)^{k(A)-k(E)}(y-1)^{k(A)+|A|-|V|}. \tag{1.12}$$

For indeterminates q and w, the *random-cluster model* is defined as

$$Z(G;q,w) = \sum_{A\subseteq E} q^{k(A)}w^{|A|}. \tag{1.13}$$

It follows elementarily that

$$T(G;x,y) = (x-1)^{-k(E)}(y-1)^{-|V|} \cdot Z(G;(x-1)(y-1),y-1). \tag{1.14}$$

The Tutte polynomial encodes a variety of interesting graph parameters, such as the number of spanning trees at $T(G;1,1)$ if G is connected, the number of acyclic orientations at $T(G;2,0)$, or the entire chromatic polynomial at $y=0$. For our purposes, it will be easier to work with Z, and in fact with its following variant, used also in [DHM+14]:

$$Z'(G;q,w) = \sum_{A \subseteq E} q^{k(A)-k(E)} w^{|A|}, \tag{1.15}$$

which, unlike Z, is not zero at $Z'(G;0,\cdot)$. We have

$$Z(G;q,w) = q^{k(E)} \cdot Z'(G;q,w). \tag{1.16}$$

1.4. Algebraic techniques

We apply several algebraic techniques in our proofs, including, most notably, the inclusion-exclusion principle and the method of polynomial interpolation.

1.4.1. Inclusion-exclusion principle

The inclusion-exclusion principle is an elementary tool from enumerative combinatorics with surprisingly powerful algorithmic applications, surveyed by [Hus11]. Given a universe Ω and several "bad" subsets of Ω, it allows us to count those elements of Ω that avoid all bad subsets, provided that we know the sizes of intersections of bad subsets.

Lemma 1.33 (Inclusion-Exclusion Principle)**.** *Let Ω be a set and let $A_1,\dots,A_t \subseteq \Omega$. For $\emptyset \subset S \subseteq [t]$, let $A_S := \bigcap_{i\in S} A_i$ and define $A_\emptyset := \Omega$. Then we have*

$$\left|\Omega \setminus \bigcup_{i\in[t]} A_i\right| = \sum_{S\subseteq[t]} (-1)^{|S|} \left|A_S\right|. \tag{1.17}$$

In applications of Lemma 1.33, the left-hand side of (1.17) will correspond to a quantity we wish to determine, while the numbers $|A_S|$ for $S \subseteq [t]$ will be computed by algorithms or oracle calls. As an example, consider the following lemma that allows to reduce counting of vertex-colorful or edge-colorful H-copies to counting of uncolored H-copies.

Lemma 1.34. *Let (G,c) be a $[k]$-vertex-colored graph, for $k \in \mathbb{N}$, and let H be an uncolored k-vertex graph. Then we can count the colorful H-copies in (G,c) by invoking 2^k oracle calls to counting uncolored H-copies in uncolored graphs $G' \subseteq G$. The same holds when considering a $[k]$-edge-colored graph G instead.*

Proof. Let Ω denote the set of all (not necessarily colorful) H-copies in G. For $i \in [k]$, let A_i denote the set of H-copies in G that have no vertices of color i when colored by c. For the edge-colorful variant, consider edges instead of vertices here. Then $\Omega \setminus \bigcup_{i\in[t]} A_i$ is precisely the set of colorful H-copies in (G, c).

For $S \subseteq [k]$, let $A_S = \bigcap_{i\in S} A_i$ and observe that $|A_S|$ is equal to the number of H-copies in the graph G_S obtained from G by deleting all vertices with colors in S. For the edge-colorful variant, consider edges instead of vertices here. We can hence determine $|A_S|$ by an oracle call, and using calls for all $S \subseteq [k]$, we can invoke Lemma 1.33 to determine $|\Omega \setminus \bigcup_{i\in[t]} A_i|$, the number of colorful H-copies in (G, c). □

1.4.2. Polynomial interpolation

It is an elementary fact that any univariate polynomial p is uniquely determined by its evaluations at sufficiently many distinct points. More precisely, if p has degree n and we can evaluate $p(\xi)$ at $n + 1$ distinct values ξ, then we can recover the coefficients of p.

Lemma 1.35. *For $n \in \mathbb{N}$ and an indeterminate x, let $p = \sum_{i=0}^{n} a_i x^i$ and let*

$$\Xi = \{\xi_0, \ldots, \xi_n\} \subseteq \mathbb{Q}$$

be a set of size $n + 1$. Then we can compute the coefficients $a_0, \ldots, a_n$ of p with $\mathcal{O}(n^3)$ arithmetic operations when given as input the set

$$\{(\xi, p(\xi)) \mid \xi \in \Xi\}.$$

Proof. The $n + 1$ known pairs induce the linear system of equations

$$\begin{pmatrix} \xi_0^0 & \ldots & \xi_0^n \\ \vdots & \ddots & \vdots \\ \xi_n^0 & \ldots & \xi_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} p(\xi_0) \\ \vdots \\ p(\xi_n) \end{pmatrix}. \tag{1.18}$$

This system features a Vandermonde matrix on $n + 1$ pairwise distinct values, and by elementary linear algebra, such matrices can be verified to have full rank. Hence, we can solve (1.18) with $\mathcal{O}(n^3)$ arithmetic operations for the coefficients of p. □

Let us consider as a first example of Lemma 1.35 how $\text{PerfMatch}^{-1,0,1}$, the problem of evaluating PerfMatch on graphs with edge-weights -1 and 1, can be reduced to the problem PerfMatch on graphs with positive integer weights by means of univariate interpolation. This is a well-known idea in counting complexity.

Lemma 1.36. *For any n-vertex graph G with edge-weights -1 and 1, we can compute* $\text{PerfMatch}(G)$ *with $\frac{n}{2} + 1$ oracle calls to* $\text{PerfMatch}(G')$ *on graphs G' derived from G by replacing all occurrences of the edge-weight -1 in G by an edge-weight from $\{0, \ldots \frac{n}{2}\}$.*

Proof. Replace the edge-weight -1 in G by an indeterminate x to obtain a graph G_x, and observe that

$$p := \text{PerfMatch}(G_x) \in \mathbb{Z}[x]$$

is a polynomial of maximum degree $\frac{n}{2}$. We can evaluate $p(\xi)$ at $x \in \{0, \dots \frac{n}{2}\}$ by means of oracle calls to PerfMatch(G') for graphs G' with edge-weights from $\{0, \dots \frac{n}{2}\}$. Hence, we can interpolate p via Lemma 1.35 and recover $p(-1) = \text{PerfMatch}(G)$. □

Using edge-weight simulations as described in Lemma 1.29, we can furthermore reduce this to the unweighted case.

Lemma 1.37. *For any graph G on n vertices and m edges, with edge-weights -1 and 1, we can compute* PerfMatch(G) *with $\frac{n}{2} + 1$ oracle calls to* PerfMatch(G') *on unweighted graphs G' with $\mathcal{O}(nm)$ vertices and edges.*

Polynomial interpolation can be generalized to multivariate polynomials, but then some care has to be taken when choosing the evaluation points. In this thesis, we will always assume that the evaluation points constitute a grid. That is, if p is defined on indeterminates $x_1, \dots, x_n$ and has maximum degree d_i in x_i, for $i \in [n]$, then we assume that we are given sets $\Xi_1, \dots, \Xi_n$ with $|\Xi_i| = d_i + 1$ for $i \in [n]$, along with evaluations of p on all grid points in $\Xi_1 \times \ldots \times \Xi_n$. More intricate evaluation sets could also be chosen, but this will not be relevant for our purposes.

Theorem 1.38 (Grid Interpolation)**.** *Let $p \in \mathbb{Z}[x_1, \dots, x_n]$ be a multivariate polynomial, and for $i \in [n]$, let the degree of x_i in p be bounded by $d_i \in \mathbb{N}$. Let*

$$\Xi = \Xi_1 \times \ldots \times \Xi_n \subseteq \mathbb{Q}^n$$

with $|\Xi_i| = d_i + 1$ for all $i \in [n]$. Then we can compute the coefficients of p with $\mathcal{O}(|\Xi|^3)$ arithmetic operations when given as input the set

$$\{(\xi, p(\xi)) \mid \xi \in \Xi\}.$$

Proof. For $s, t, s', t' \in \mathbb{N}$ and matrices $A \in \mathbb{Q}^{s \times t}$ and $B \in \mathbb{Q}^{s' \times t'}$, we write $A \otimes B$ for the Kronecker product of A and B, which is the matrix $A \otimes B \in \mathbb{Q}^{s \cdot s' \times t \cdot t'}$ whose rows are indexed by $[s] \times [s']$, whose columns are indexed by $[t] \times [t']$, and which satisfies

$$(A \otimes B)_{(i,i'),(j,j')} = A_{i,j} \cdot B_{i',j'} \quad \text{for } (i, i') \in [s] \times [s'] \text{ and } (j, j') \in [t] \times [t'].$$

For $\ell \in [n]$, let $\Xi_\ell = \{a_{\ell,1}, \dots, a_{\ell,d_\ell+1}\}$ and let $A^{(\ell)}$ denote the $(d_\ell + 1) \times (d_\ell + 1)$ Vandermonde matrix with $A^{(\ell)}_{i,j} = (a_{\ell,i})^j$ for all $i, j \in [d_\ell + 1]$. As noted before, such a matrix has full rank. Let

$$A := A^{(1)} \otimes \ldots \otimes A^{(n)}.$$

Since each $A^{(\ell)}$ for $\ell \in [n]$ has full rank, so does A, by an elementary property of the Kronecker product [Lau04, Corollary 13.11].

Let $\mathbf{c}$ denote the vector that lists the coefficients of p in lexicographic order,[9] and let $\mathbf{v}$ denote the vector that lists the evaluations $p(\xi)$ for $\xi \in \Xi$ in lexicographic order. Then it can be verified that

$$A\mathbf{c} = \mathbf{v}.$$

Since A has full rank, this system of linear equations can be solved with $\mathcal{O}(|\Xi|^3)$ arithmetic operations for $\mathbf{c}$, and we obtain the coefficients of p. □

1.5. Structural graph theory

In the final section of the preliminaries, we briefly survey some concepts and results from structural graph theory that will be required in Parts I and II.

1.5.1. Minors

A graph H is a *minor* of $G = (V, E)$, written as $H \preceq G$, if H can be obtained from G by repeated edge-contractions as well as deletions of edges or vertices. Here, the *contraction* of an edge $uv \in E$ is the operation that identifies its endpoints u and v to a new vertex w and replaces every edge $uz \in E$ or $vz \in E$ for $z \in V$ by a new edge wz. Contractions may yield non-simple graphs (with parallel edges), and in Chapter 2, we will in fact explicitly consider the resulting graphs to be non-simple.

An equivalent definition of a minor is obtained by requiring H to have a *minor model* in G. This is an assignment of pairwise disjoint *branch sets* $B_v \subseteq V(G)$ to the vertices $v \in V(H)$ such that the induced graphs $G[B_v]$ are connected, and for every edge $uv \in E(H)$, the graph G contains an edge between B_u and B_v.

It was shown by Robertson and Seymour [RS95] that, on input H and G, it can be tested in time $f(H)n^3$ whether $H \preceq G$ holds, where f is a computable function. The running time was later improved to $f(H)n^2$ with a simplified proof [KKR12]. We obtain:

Theorem 1.39 ([RS95, KKR12])**.** *Given graphs H and G as input, deciding whether $H \preceq G$ holds is fixed-parameter tractable in the parameter $|H|$.*

For a set of graphs A, denote the class of graphs that exclude every minor in A by

$$\mathrm{Excl}[A] = \{G \mid \forall H \in A : H \not\preceq G\}.$$

In this notation, Kuratowski's classical theorem [Kur30] states that $\mathrm{Excl}[K_{3,3}, K_5]$ coincides with the class of planar graphs.

A graph class $\mathcal{H}$ is minor-closed if, whenever $G \in \mathcal{H}$ and $H \preceq G$ hold, then $H \in \mathcal{H}$ holds as well. The planar graphs are closed under minors, as it can be seen that all three operations allowed for obtaining a minor in fact preserve planarity. Other minor-closed graph classes can also be succinctly expressed by forbidden minors, and in fact, the Graph

[9] This vector includes the coefficients of all monomials with degree at most d_i in x_i, even if some of these coefficients may be zero.

Minor Theorem [RS04], a deep result in structural graph theory, asserts that *every* minor-closed graph class can be characterized by a *finite* set of forbidden minors. Together with Theorem 1.39, this yields a strong corollary for the recognition of graphs from minor-closed graph classes.

Theorem 1.40 (Graph Minor Theorem [RS04])**.** *Let $\mathcal{H}$ be a minor-closed graph class. Then there exists a finite set A such that $\mathcal{H} = \mathrm{Excl}[A]$.*

Corollary 1.41. *Let $\mathcal{H}$ be a minor-closed graph class. Given a graph G, we can decide in time $\mathcal{O}(n^2)$ whether $G \in \mathcal{H}$ holds, where the factor in the $\mathcal{O}$-notation hides a constant depending only on $\mathcal{H}$.*

Theorem 1.40 relies upon the *Graph Structure Theorem* [RS03], whose statement we sketch in Section 1.5.3 for its algorithmic implications. Roughly speaking, this structure theorem allows to constructively describe the structure of graphs in $\mathrm{Excl}[H]$ for any single fixed graph H. In order to state this result, and for various other purposes throughout this thesis, we need to define the concept of *tree decompsition*s, which is the task of the following subsection. For a more extensive introduction into the field of graph minor theory, please consider the textbook [Die12].

1.5.2. Tree decompositions

Tree decompositions of graphs and the related notion of *treewidth* are powerful tools for graph algorithms, as well as in structural graph theory, and they will be used often throughout this thesis. The notion of treewidth was first described in [Hal76], then reinvented in [RS84]. Roughly speaking, a tree decomposition of a graph G shows how to decompose G hierarchically along separators.

Definition 1.42. A *tree decomposition* of a graph G is a pair $\mathcal{T} = (T, \mathcal{B})$, where T is a rooted tree and $\mathcal{B} = \{B_t\}_{t \in V(T)}$ is a family of subsets from $V(G)$, so-called *bags*, such that the following holds:

1. Every vertex of G is contained in some bag, that is, $\bigcup_{t \in V(T)} B_i = V(G)$.
2. Every edge $uv \in E(G)$ is contained in a bag, that is, $u, v \in B_t$ for some $t \in V(T)$.
3. For every vertex $v \in V(G)$, the set $\{t \in V(T) \mid v \in B_t\}$ is connected in T.

To avoid confusion, we call the vertices of T *nodes*. The *width* of $\mathcal{T}$ is defined as $\max_{t \in V(T)} |B_t| - 1$. The *treewidth* of G, denoted by $\mathrm{tw}(G)$, is the minimum width over all tree decompositions of G.

While it is NP-complete to compute the treewidth of a graph [ACP87], there are fixed-parameter tractable algorithms for finding a tree decomposition of width k when given as input a graph G of treewidth at most k, see [Bod96].

Graphs of small treewidth may be considered "very tractable", in the sense that exceedingly many NP-hard and #P-hard problems become fixed-parameter tractable when parameterized by $\mathrm{tw}(G)$. This includes computing the treewidth itself, as seen before, but also counting independent sets, counting matchings of arbitrary sizes, evaluating the Tutte polynomial [Nob98], and various other problems. All these algorithmic results can be shown by a general dynamic programming approach on the tree decomposition of G.

Monadic second-order logic on bounded-treewidth graphs

In fact, the dynamic programming approach mentioned in the last paragraph is *so* general that it can be phrased as an algorithmic meta-theorem for counting models of monadic second-order logic (MSOL) formulas. This logic extends first-order logic by allowing variables X that range over *sets* of vertices or edges rather than merely individual vertices or edges. The semantics of this logic is defined as expected, and we illustrate it by the following example.

Example 1.43. We consider MSOL formulas over the vocabulary of graphs, which consists of a universe symbol Ω, unary relation symbols V, E and the binary relation symbol I.[10] In structures over this vocabulary, Ω is interpreted as the set of vertices and edges of a graph, while V and E express whether $x \in \Omega$ is a vertex or an edge, and $I(x, y)$ is true if (the vertex) x is incident with (the edge) y. We define

$$\begin{aligned} \varphi_{PerfMatch}(X) &:= (\forall x \in X : x \in E) \wedge \phi_{vtxdis}(X) \wedge \phi_{cover}(X), \\ \varphi_{Match}(X) &:= (\forall x \in X : x \in E) \wedge \phi_{vtxdis}(X), \end{aligned}$$

using auxiliary formulas

$$\begin{aligned} \phi_{vtxdis}(X) &:= \forall x, y \in X : x \neq y \rightarrow \neg \exists z \in V : I(z, x) \wedge I(z, y), \\ \phi_{cover}(X) &:= \forall x \in V : \exists y \in X : I(x, y). \end{aligned}$$

Given a simple graph G, encoded as a structure $A = (\Omega, V, E, I)$ over the vocabulary of graphs, and a subset $X \subseteq \Omega$, we have $A \models \varphi_{PerfMatch}(X)$ iff X is a perfect matching of G, and $A \models \varphi_{Match}(X)$ iff X is a matching of G. Here, we write $A \models \varphi$ if A is a model of φ.

It is by now a classical result that the models to MSOL formulas φ over the vocabulary of graphs can be found [ALS91, CM93] or counted [CMR01] in time $f(|\varphi|, k) \cdot n$ on graphs with treewidth k, for a computable function f. This generalizes also to a weighted counting version [Mak04] that we state in the following. To simplify notation, let us tacitly identify G with its encoding structure (Ω, V, E, I).

Theorem 1.44 ([CMR01, Mak04])**.** *Let $\varphi(X)$ be a formula in MSOL over the vocabulary of graphs, with a free set variable X. Given as input φ and a graph G of treewidth k, we*

[10] There are several vocabularies of graphs. A more restrictive version allows the universe only to contain vertices, and the edges are given by a binary relation E. This more restrictive vocabulary will not be considered in this thesis.

can determine in time $f(|\varphi|,k)\cdot n$ *the number of sets* $X \subseteq V(G) \cup E(G)$ *such that* G *is a model of* $\varphi(X)$*. Here,* f *is a computable function.*

In fact, we may assume that G *is given with a weight function* $w : V(G) \cup E(G) \to \mathbb{Q}$*, and then we can evaluate, in the same time, the quantity*

$$p_\varphi(G) = \sum_{\substack{X \subseteq V(G) \cup E(G) \\ G \models \varphi(X)}} \prod_{x \in X} w(x).$$

Example 1.43 and Theorem 1.44 together imply the following corollary for the tractability of PerfMatch on graphs of bounded treewidth:

Corollary 1.45. *The problem* PerfMatch/tw *of evaluating* PerfMatch(G) *on parameter* tw(G) *admits an algorithm with running time* $f(\mathrm{tw}(G))n$*, where* f *is a computable function.*

Structural implications of tree decompositions

Given a tree decomposition $\mathcal{T} = (T, \mathcal{B})$ and a node $t \in V(T)$, let us we write $T_{\downarrow t}$ for the subtree rooted at t. For sets $X \subseteq V(T)$, we write $B(X) = \bigcup_{t \in X} B_t$. We also slightly abuse this notation: If S is a subtree of T, we abbreviate $B(S) = B(V(S))$. As we observe in the following, taken from [FG06, Lemma 11.3], the intersections of bags in a tree decomposition allow to separate G.

Lemma 1.46. *Let* G *be a graph with tree decomposition* $\mathcal{T} = (T, \mathcal{B})$*, where* $\mathcal{B} = \{B_t\}_{t \in V(T)}$*. Let* $s, t \in V(T)$ *be such that* s *is the parent of* t*, and write* $X = B_s \cap B_t$*. Then every path from a vertex in* $B(T_{\downarrow t})$ *to a vertex in* $B(T \setminus T_{\downarrow t})$ *contains a vertex from* X*.*

This lemma illustrates a key property of tree decompositions: Let $t \in V(T)$ be fixed and consider the subtree $T_{\downarrow t}$ rooted at t. In the same way any *node* in $T_{\downarrow t}$ can only "communicate" with nodes outside of $T_{\downarrow t}$ by passing through t, so can the *vertices* of G contained in $B(T_{\downarrow t})$ only communicate with other vertices by passing through some vertices in B_t. Using this lemma, it is easily shown that, if G contains a clique K, then no tree decomposition can split this clique, see [FG06, Exercise 11.4]

Lemma 1.47. *Let* G *be a graph with tree decomposition* $\mathcal{T} = (T, \mathcal{B})$*, where* $\mathcal{B} = \{B_t\}_{t \in V(T)}$*. If* K *is a clique in* G*, then there exists some node* $t \in V(T)$ *such that* $K \subseteq V(B_t)$*.*

We also observe that the treewidth is monotone under the minor relation:

Lemma 1.48. *If* $G \preceq G'$*, then* $\mathrm{tw}(G) \leq \mathrm{tw}(G')$*.*

The treewidth of a graph can also be related to its excluded minors: By the Excluded Grid Theorem [RS86], every graph with sufficiently large treewidth contains the $k \times k$ square grid as a minor. Here, the $k \times k$ square grid is the graph on vertices $[k]^2$ where two vertices are connected by an edge iff they are horizontally or vertically adjacent, as specified in Section 1.1. Since every planar graph H is the minor of some square grid, as can be seen from a rectilinear drawing of H where vertices of H are represented by sufficiently large squares, this theorem implies that the graphs in Excl[H] for fixed planar graphs H have constant treewidth $c = c(H)$.

Theorem 1.49 (Excluded Grid Theorem [RS86])**.** *For every $k \geq 1$, there is a computable integer $b(k)$ such that every graph of treewidth at least $b(k)$ contains the $k \times k$ square grid as a minor.*

In the original proof [RS86], the function b is exponential, but a recent proof shows that $b(k) \leq k^{\mathcal{O}(1)}$ can also be achieved [CC14]. For our applications, we only require the growth rate of b to be bounded by some computable function of k.

1.5.3. Structure of H-minor free graphs

In this section, we state the Graph Structure Theorem [RS03], a result in graph minor theory that will be required in Part I and was already mentioned in the beginning of this section. Let us define the *Hadwiger number* $\mathrm{hadw}(G)$ of a graph G as

$$\mathrm{hadw}(G) = \max\{k \in \mathbb{N} \mid K_k \preceq G\}.$$

The Graph Structure Theorem shows how to decompose graphs in $\mathrm{Excl}[H]$, for arbitrary fixed H, into simpler constituents that *almost* admit drawings on surfaces of genus $c = c(H)$, apart from certain defects, namely so-called *vortices* and *apices*. In other words, it describes the structure of graphs that satisfy $\mathrm{hadw}(G) \leq k$ for some fixed $k \in \mathbb{N}$. The constituents guaranteed by this decomposition are described in the following definition.

Definition 1.50. Let $k, p, w, \gamma \in \mathbb{N}$ and let G be a graph. We say that G *has genus γ apart from p vortices of width w* if G contains p subgraphs $G_1, \ldots, G_p$ (so-called vortices) such that the following holds: The graph G_0 obtained from G after deleting $E(G_i)$ for all $i \in [p]$ admits a drawing π on a surface of genus γ such that:

1. There are faces $F_1, \ldots F_p$ of π such that G_i for $i \in [p]$ can be drawn within F_i without crossing F_i. The drawing of G_i itself may however feature crossings.

2. For each $i \in [p]$, write $F = F_i$ and let $f_1, \ldots, f_\ell$ with $\ell = |F|$ denote the vertices of F in clockwise order around F. Then the graph G_i has a tree decomposition $\mathcal{T} = (T, B)$ of width w such that $V(T) = [\ell]$ and T is a path. Furthermore, for all $s \in [\ell]$, we have $f_s \in B_s$.

For $k \in \mathbb{N}$, we say that G is *k-almost embeddable* if there is a k-set $A \subseteq V(G)$ of *apices* such that $G - A$ has genus k apart from k vortices of width k.

By the Graph Structure Theorem, any graph $G \in \mathrm{Excl}[H]$ admits a tree decomposition into parts that are k-almost embeddable, for $k = f(H)$. To state this precisely, we have to add the notions of adhesion and torsos to tree decompositions. The adhesion is simply the maximum intersection of adjacent bags B_s and B_t. The torso G_s is obtained from the subgraph $G[B_s]$ induced by a bag B_s by completing the intersections with adjacent bags to cliques.

Definition 1.51. Let G be a graph and let $\mathcal{T} = (T, \mathcal{B})$ with $\mathcal{B} = \{B_t\}_{t \in V(T)}$ be a tree decomposition of G. Then the *adhesion* of $\mathcal{T}$ is defined as $\max_{st \in E(T)} |B_s \cap B_t|$.

For $s \in V(T)$, the *torso* of s is the graph G_s obtained from the induced subgraph $G[B_s]$ by adding all edges between vertices in $B_s \cap B_t$, for all neighbors t of s.

With Definitions 1.50 and 1.51, we can finally state the Graph Structure Theorem [RS03]. Its algorithmic variant was shown in [DHT05] and later simplified in [KW11].

Theorem 1.52 (Graph Structure Theorem [RS03])**.** *For every graph H, there is a computable constant $k = k(H)$ such that the following holds: Every graph $G \in \mathrm{Excl}[H]$ admits a tree decomposition $\mathcal{T} = (T, \mathcal{B})$ in which every torso G_s for $s \in V(T)$ is k-almost embeddable. Furthermore, such a decomposition can be found in time $f(H)n^{\mathcal{O}(1)}$ for a computable function f.*

Note that the *width* of the tree decomposition $\mathcal{T}$ guaranteed by Theorem 1.52 may be unbounded. It can however be verified from the requirements on torsos that the *adhesion* of $\mathcal{T}$ is bounded by a function of k. Together with Lemma 1.46, this implies that G has small separators which decompose G into k-almost embeddable graphs, and this is useful for various algorithmic purposes.

Remark 1.53. In view of the constituents guaranteed by Theorem 1.52, it does not come as a surprise that $\mathrm{hadw}(G)$ is a lower bound for several graph parameters we have already encountered, such as the apex number $\mathrm{apex}(G)$, the genus $\gamma(G)$, and the treewidth $\mathrm{tw}(G)$. That is, there exists a (polynomial) function f such that

$$\mathrm{hadw}(G) \leq f(\min\{\mathrm{apex}(G), \mathrm{tw}(G), \gamma(G)\}).$$

Recall that $\mathrm{apex}(G)$, already encountered in Section 1.2.2, is the minimum size of a set $S \subseteq V(G)$ such that $G - S$ is planar.

2. The Holant framework

In this chapter, we present the concept of *Holant problems*, which were defined implicitly by Valiant in his seminal paper on holographic algorithms [Val08], and were made explicit by Cai et al. [CL07] in their extensive study of Holant problems and the associated notions of holographic algorithms and matchgates.[1] See also [CLX09b, KC10, CHL10, CLX11]. For the scope of the present thesis, we limit our treatment of these concepts to a bare minimum, just enough to obtain a language that allows to express the technical content of subsequent parts cleanly.

Holant problems provide a powerful framework for counting problems, and they subsume the problems of counting graph homomorphisms [DGP07] and counting constraint satisfaction problems [Bul13]. The instances to Holant problems are *signature graphs* Ω, which are multigraphs with a function f_v at each vertex $v \in V(\Omega)$, such that f_v outputs a rational number for each assignment $x \in \{0,1\}^{I(v)}$ to the edges incident with v. Given Ω as input, the task of a Holant problem is then to evaluate $\mathrm{Holant}(\Omega)$, a weighted sum over the assignments $x \in \{0,1\}^{E(\Omega)}$, where an assignment x is weighted by the product of the evaluations $f_v(x)$ for $v \in V(\Omega)$.

In Section 2.1, we formalize this definition, show exemplarily how the counting problems PerfMatch, MatchSum and #SAT can be reformulated as Holant problems, and describe some useful operations on signature graphs. In Section 2.2, we introduce the notion of *matchgates*, which play a central role throughout the thesis, and we proceed to show that they enable a uniform reduction from *every* Holant problem to PerfMatch. This yields a flexible way for proving intractability of PerfMatch in various restricted cases. Finally, in Section 2.3, we introduce *combined signatures*, our main conceptual contribution to the theory of Holant problems, which adds a bridge towards parameterized counting complexity. This last section is joint work with Mingji Xia.

2.1. Basic definitions

In the following, we give a concise introduction to what we call the *Holant framework*, a toolbox that borrows from [Val08, CL07, CLX08] and resembles the framework of *tensor networks* studied in physics [Lan12]. Recall that, given a graph G and $v \in V(G)$, we denote the edges incident with v by $I(v)$.

Definition 2.1. A *signature graph* is an edge-weighted graph Ω, which may explicitly feature parallel edges, and which has a *vertex function* $f_v : \{0,1\}^{I(v)} \to \mathbb{Q}$ associated with each vertex $v \in V(\Omega)$.

[1] The term *Holant*, which was coined by Valiant, is not a geographic tribute, but should rather be seen as a contraction of the words "holographic" and "permanent".

The *Holant* of Ω is a particular sum over its edge assignments $x \in \{0,1\}^{E(\Omega)}$. Given an assignment $x \in \{0,1\}^{E(\Omega)}$, we say that an edge $e \in E(\Omega)$ is *active in* x if $x(e) = 1$ holds, otherwise e is *inactive in* x. As mentioned in the preliminaries, we tacitly identify x with the set of active edges in x. Given a subset $S \subseteq E(\Omega)$, we write $x|_S$ for the restriction of x to S, which is the unique assignment in $\{0,1\}^S$ that agrees with x on S.

Definition 2.2 (adapted from [Val08])**.** Let Ω be a signature graph with edge weights $w : E(\Omega) \to \mathbb{Q}$ and a vertex function $f_v : \{0,1\}^{I(v)} \to \mathbb{Q}$ for each $v \in V(\Omega)$. Furthermore, let $x \in \{0,1\}^{E(\Omega)}$ be an assignment to the edges of Ω. Then we define

$$\operatorname{val}_\Omega(x) := \prod_{v \in V(\Omega)} f_v(x|_{I(v)}), \tag{2.1}$$

$$w_\Omega(x) := \prod_{e \in x} w(e), \tag{2.2}$$

and we say that x *satisfies* Ω if $\operatorname{val}_\Omega(x) \neq 0$ holds. Furthermore, we define

$$\operatorname{Holant}(\Omega) := \sum_{x \in \{0,1\}^{E(\Omega)}} w_\Omega(x) \cdot \operatorname{val}_\Omega(x). \tag{2.3}$$

For $x \in \{0,1\}^{E(\Omega)}$ and $v \in V(\Omega)$, it is sufficient to know $x|_{I(v)}$ in order to evaluate f_v on x. This is of particular interest for signature graphs of bounded maximum degree: In such graphs, each vertex function f_v depends only on $O(1)$ edge values.

We will sometimes consider *unweighted* signature graphs; for such graphs, we can omit the factor $w_\Omega(x) = 1$ appearing in the definition of $\operatorname{Holant}(\Omega)$. In fact, we will later observe that every weighted signature graph can be easily transformed to an unweighted version while preserving Holants and modifying Ω only slightly. Some arguments are however simpler when carried out on edge-weighted signature graphs.

A particularly useful type of vertex functions is that of *Boolean functions*, whose ranges are restricted to $\{0,1\}$ rather than $\mathbb{Q}$. If all signatures appearing in a signature graph Ω' are Boolean, then $\operatorname{Holant}(\Omega')$ simply sums over those assignments $x \in \{0,1\}^{E(\Omega')}$ that pass all constraints imposed by the vertex functions, and each x is weighted by $w_{\Omega'}(x)$. That is,

$$\operatorname{Holant}(\Omega') = \sum_{\substack{x \in \{0,1\}^{E(\Omega')} \\ \forall v\colon f_v(x)=1}} w_{\Omega'}(x). \tag{2.4}$$

We use such a Boolean function to reformulate PerfMatch as a Holant problem.

Example 2.3. Recall that $\operatorname{hw}(x)$ denotes the Hamming weight of x. Given an edge-weighted graph G, let $f_v : \{0,1\}^{I(v)} \to \{0,1\}$ for $v \in V(G)$ be the vertex function

$$f_v(x) = \begin{cases} 1 & \text{if } \operatorname{hw}(x) = 1, \\ 0 & \text{otherwise,} \end{cases} \tag{2.5}$$

and let Ω denote the signature graph obtained from G by associating f_v with v, for all $v \in V(G)$. Note that, while the definition of the vertex functions f_v and $f_{v'}$ for distinct vertices $v, v' \in V(\Omega)$ in (2.5) agrees, their domains $\{0,1\}^{I(v)}$ and $\{0,1\}^{I(v')}$ may differ.

Since all vertex functions in Ω are Boolean, we see with (2.4) that Holant(Ω) ranges over those assignments $x \in \{0,1\}^{E(\Omega)}$ in which each vertex is incident with exactly one active edge. Each such x is weighted by $w_\Omega(x) = \prod_{e \in x} w(e)$. This is precisely the expression of PerfMatch(G), as defined in (1.7). □

2.1.1. Signatures

To allow for attaching the same vertex function to different vertices, as seen in Example 2.3, we use *signatures*, which are functions over the abstract domain $\{0,1\}^{[d]}$ for $d \in \mathbb{N}$.

Given a signature graph Ω and a vertex $v \in V(\Omega)$ of degree $d \in \mathbb{N}$, assume that $I(v)$ is ordered by a bijection $\sigma_v : [d] \to I(v)$. In the following, such bijections will always be specified by the context, and they will typically be one of the following:

- If $E(\Omega)$ is ordered in some canonical way, then we may define σ_v for $v \in V(\Omega)$ as the restriction of this ordering to $I(v)$.

- If Ω is given as a plane graph, then we may define σ_v for $v \in V(\Omega)$ by choosing $\sigma_v(1)$ arbitrarily and then defining σ_v as the clockwise order of $I(v)$ around v, starting with $\sigma_v(1)$.

When such an ordering σ_v of $I(v)$ is fixed, then we can equivalently consider $f_v : \{0,1\}^{I(v)} \to \mathbb{Q}$ as a function of the type $\{0,1\}^{[d]} \to \mathbb{Q}$. We formalize this in the following definition:

Definition 2.4. For $d \in \mathbb{N}$, a *signature of arity* d is a function $s : \{0,1\}^{[d]} \to \mathbb{Q}$. Let Ω be a signature graph, let $v \in V(\Omega)$ be a vertex of degree $d \in \mathbb{N}$, and let $\sigma_v : [d] \to I(v)$ be a bijective function.

Given an assignment $x \in \{0,1\}^{I(v)}$, we write $\sigma_v x \in \{0,1\}^{[d]}$ for the assignment that maps each $i \in [d]$ to $x(\sigma_v(i))$, and we define a function $\sigma_v f_v : \{0,1\}^{[d]} \to \mathbb{Q}$ that satisfies

$$\forall x \in \{0,1\}^{I(v)} : \ (\sigma_v f_v)(\sigma_v x) = f_v(x).$$

Then we say that $\sigma_v f_v$ is the *signature attached to* v *under* σ_v. Note that, if σ_v is bijective, then $\sigma_v f_v$ is indeed a function, and it is defined on the entire set $\{0,1\}^{[d]}$.

It should be noted that will rarely treat signatures as formally as in Definition 2.4. The following simple signatures will however occur often throughout this thesis.

Example 2.5. We consider the following (families of) signatures of arity $k \in \mathbb{N}$. On inputs $x \in \{0,1\}^{[k]}$ with $x = (x_1, \ldots, x_k)$, they are defined as

$$\begin{aligned} \texttt{EQ} &: x \mapsto [x_1 = \ldots = x_k] \\ \texttt{HW}_{=1} &: x \mapsto [\mathrm{hw}(x) = 1] \\ \texttt{HW}_{\leq 1} &: x \mapsto [\mathrm{hw}(x) \leq 1] \\ \texttt{ODD} &: x \mapsto x_1 \oplus \ldots \oplus x_k \\ \texttt{EVEN} &: x \mapsto 1 \oplus x_1 \oplus \ldots \oplus x_k. \end{aligned}$$

Occasionally, we use subscripts to make the arity of a signature explicit, e.g., for $k \in \mathbb{N}$, we may write $\texttt{EQ}_k$ for the signature of arity k in the family $\texttt{EQ}$.

To define the signature attached at v in Definition 2.4, we assumed an ordering σ_v of $I(v)$. For *symmetric* vertex functions, this is not necessary; such functions f_v satisfy $f_v(x) = f_v(x')$ for all assignments $x, x' \in \{0,1\}^{I(v)}$ that can be obtained from another by permutations. If f_v is symmetric, then its output depends only on the Hamming weight of its input, and in particular, the signatures $\sigma_v f_v$ under *all* orderings σ_v are equal, so we usually omit the orderings σ_v in such cases. Note that all signatures from Example 2.5 are symmetric; we will however also consider non-symmetric signatures later.

2.1.2. More examples for Holant problems

To illustrate the expressive power of the Holant framework, we revisit some problems from Chapter 1 and show that they can be reformulated as the problem of evaluating Holant(Ω) for a given signature graph. These examples will be used throughout the thesis.

MatchSum as a Holant problem

As a first example, we consider the evaluation of the polynomial MatchSum on vertex-weighted graphs.

Example 2.6. Let G be a graph with vertex weights $w : V(G) \to \mathbb{Q}$. Then we can express MatchSum(G) as a Holant problem by using the following symmetric signature $\texttt{VTX}_w$ for $w \in \mathbb{Q}$.

$$\texttt{VTX}_w : \quad x \mapsto \begin{cases} w & \text{if } \mathrm{hw}(x) = 0, \\ 1 & \text{if } \mathrm{hw}(x) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

To this end, we construct a signature graph Ω from G by attaching, at each $v \in V(G)$, the signature $\texttt{VTX}_{w(v)}$. Note that no ordering of $I(v)$ needs to be specified, since $\texttt{VTX}_{w(v)}$ is symmetric. In every satisfying assignment $x \in \{0,1\}^{E(\Omega)}$, each vertex $v \in V(\Omega)$ is incident with at most one active edge, so x is a (not necessarily perfect) matching, and $\mathrm{val}_\Omega(x)$ is the product of the following factors:

- If v is incident with exactly *one* active edge, then v contributes 1 to $\mathrm{val}_\Omega(x)$.
- If v is incident with *no* active edges, so $v \in \mathrm{usat}(G, x)$, then v contributes $w(v)$.

Hence, it holds that $\mathrm{val}_\Omega(x) = \prod_{v \in \mathrm{usat}(G,x)} w(v)$. Summing over all matchings, we obtain $\mathrm{Holant}(\Omega) = \mathrm{MatchSum}(G)$ by identifying terms. □

#SAT as a Holant problem

Holant problems also capture problems outside the realm of graph theory. This includes the problem #SAT, as we show in the following example.

Example 2.7. For $n, m, d \in \mathbb{N}$, let φ be a d-CNF formula with the variables $x_1, \ldots, x_n$ and the clauses $c_1, \ldots, c_m$. We construct a signature graph Ω with

$$\#\mathsf{SAT}(\varphi) = \mathrm{Holant}(\Omega) \tag{2.6}$$

such that Ω has $n + m$ vertices, dm edges, and $\Delta(\Omega)$ is bounded by the maximum of d and $\max_{i \in [n]} r(i)$, where $r(i)$ denotes the number of occurrences of x_i as a positive or negative literal in φ. This signature graph Ω is merely the variable-clause graph of φ, with appropriate signatures:

- For each $i \in [n]$, we create a *variable vertex* v_i in Ω, with signature $\mathtt{EQ}_{r(i)}$.
- For each $j \in [m]$, we consider the clause c_j as a Boolean function of arity d over the variables $x_{i_1}, \ldots, x_{i_d}$ it depends upon. We create a *clause vertex* w_j in Ω, and for $\kappa \in [d]$, we add the edge $w_j x_{i_\kappa}$ as the κ-th edge in the ordering of $I(w_j)$. Then we attach the signature c_j to w_j.

By the $\mathtt{EQ}$ signatures at variable vertices, every satisfying assignment $x \in \{0,1\}^{E(\Omega)}$ corresponds to a unique binary assignment $x' : \{x_1, \ldots, x_n\} \to \{0,1\}$ to the variables of φ. Furthermore, for all $j \in [m]$, the signature c_j at the clause vertex w_j ensures that x' satisfies clause c_j, so altogether x' satisfies φ. Likewise, every satisfying assignment to φ induces such a satisfying assignment $x \in \{0,1\}^{E(\Omega)}$, thus proving (2.6). □

Removing edge weights from signature graphs

We proceed to show how edge-weighted Holant problems can be reduced to unweighted ones while keeping the structure of the involved graphs intact up to subdivisions.

Lemma 2.8. *Let Ω be a signature graph and let Ω' be defined by subdividing each edge $e \in E(\Omega)$ once, assigning weight 1 to the obtained subdivision edges, and equipping the obtained subdivision vertex with the symmetric signature $\mathtt{EDGE}_{w(e)}$ of arity 2, where*

$$\mathtt{EDGE}_w : \quad x \mapsto \begin{cases} w & \text{if } x = 11, \\ 0 & \text{if } x \in \{01, 10\}, \\ 1 & \text{if } x = 00. \end{cases}$$

Then Ω' features only the edge-weight 1, and we have $\mathrm{Holant}(\Omega) = \mathrm{Holant}(\Omega')$.

Proof. The satisfying assignments $x \in \{0,1\}^{E(\Omega)}$ stand in bijection with those of Ω': Every such x can be transformed to a satisfying assignment $x' \in \{0,1\}^{E(\Omega')}$ by assigning, for each $e \in E(\Omega)$, the value $x(e)$ to both edges e_1, e_2 obtained in Ω' from subdividing e.

Likewise, every such $x' \in \{0,1\}^{E(\Omega')}$ can be "contracted" to a unique satisfying assignment $x \in \{0,1\}^{E(\Omega)}$, since $x'(e_1) = x'(e_2)$ holds for every edge pair e_1,e_2 replacing an original edge $e \in E(\Omega)$. We observe that $w_\Omega(x) \cdot \mathrm{val}_\Omega(x) = w_{\Omega'}(x') \cdot \mathrm{val}_{\Omega'}(x')$ holds, which shows the claim. □

2.1.3. Gates and matchgates

Given a signature graph Ω, we can replace any vertex set $S \subseteq V(\Omega)$ by a single vertex v_S with an appropriate signature, while preserving the Holant of Ω. In other words, we may cut out an induced subgraph from Ω and contract it to a single vertex that simulates the entire subgraph.

For the following definition, we use the notion of *dangling edges*, which are "edges" that feature only one endpoint. In other words, there is precisely one vertex incident with such a dangling edge.[2] This notion is borrowed from [CLX08], and the *gates* we define in the following are parallel to the so-called $\mathcal{F}$-gates introduced in that paper.

Definition 2.9. A *gate* is a signature graph Γ, possibly containing a set $D \subseteq E(\Gamma)$ of dangling edges, all of which have edge-weight 1.

For disjoint sets A and B, and for assignments $x \in \{0,1\}^A$ and $y \in \{0,1\}^B$, we write $xy \in \{0,1\}^{A \cup B}$ for the assignment that agrees with x on A, and with y on B. We also say that the assignment xy *extends* x.

Then the *signature of* Γ is the function $\mathrm{Sig}(\Gamma) : \{0,1\}^D \to \mathbb{Q}$ that maps x to

$$\mathrm{Sig}(\Gamma, x) = \sum_{\substack{xy \in \{0,1\}^{E(\Gamma)}: \\ y \in \{0,1\}^{E(\Gamma) \setminus D}}} w_\Gamma(xy) \cdot \mathrm{val}_\Gamma(xy). \tag{2.7}$$

We also say that Γ *realizes* $\mathrm{Sig}(\Gamma)$.

That is, given an assignment $x \in \{0,1\}^D$ to the dangling edges of Γ, the value $\mathrm{Sig}(\Gamma, x)$ is a restriction of $\mathrm{Holant}(\Gamma)$ that sums only over assignments $y \in \{0,1\}^{E(\Gamma)}$ with $y|_D = x$.

Contracting and inserting gates

In the following, we formalize the operation of *contracting* a gate, mentioned already before, and its dual operation of *inserting* a gate into a signature graph.

[2] If the reader feels uncomfortable with this notion, a "virtual" and irrelevant vertex v_0 outside of G may be considered, and dangling edges can be made incident with v_0.

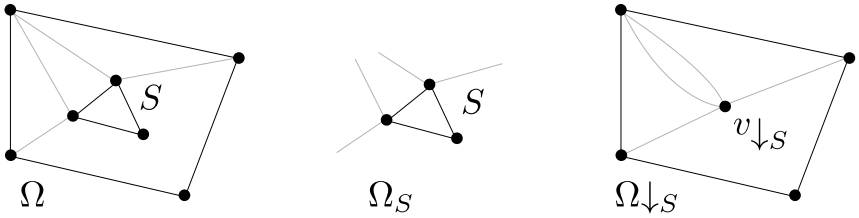


Figure 2.1.: For $S \subseteq V(\Omega)$, the gate Ω_S and the graph $\Omega{\downarrow_S}$ from Definition 2.10 are shown. Edges in the cut $E(S, \overline{S})$ are gray and appear as dangling edges in Ω_S. Note that $\Omega{\downarrow_S}$ features parallel edges. When viewing this from left to right, Ω_S is contracted in Ω. From right to left, Ω_S is inserted at the vertex $v{\downarrow_S}$.

Definition 2.10. Let Ω be a signature graph. For $S \subseteq V(\Omega)$, let $E(S, \overline{S})$ denote the set of edges $uv \in E(\Omega)$ with $u \in S$ and $v \in V(\Omega) \setminus S$, that is, edges crossing the cut S in Ω.

- We write Ω_S for the *induced gate* obtained from the induced subgraph $\Omega[S]$ by adding the edges in $E(S, \overline{S})$ as dangling edges that only have an endpoint in S.
- The *contraction* $\Omega{\downarrow_S}$ is defined by replacing Ω_S in Ω with a single vertex $v{\downarrow_S}$ that is incident to $E(S, \overline{S})$ and features the vertex function $\mathrm{Sig}(\Omega_S)$. For every edge $e \in E(S, \overline{S})$, replace its endpoint in S by $v{\downarrow_S}$. See Figure 2.1 for an example.
- Conversely, given a set $D \subseteq E(\Omega)$ of dangling edges and a gate Γ containing the same dangling edges, we can *insert* Γ *at* D by placing a copy of Γ into Ω and identifying each dangling edge $e \in D$ across Γ and Ω. That is, if e is a dangling edge with endpoint u in Ω, and a dangling edge with endpoint v in Γ, then we consider e as an edge uv in the resulting graph. If $v \in V(\Omega)$ is a vertex with $I(v) = D$, then we can also *insert* Γ *at* v by deleting v first, keeping $I(v)$ as dangling edges, and then inserting Γ at $I(v)$.

Note that contractions of gates are similar to the contractions used in graph minor theory. A simple calculation shows that contracting a gate preserves Holants.

Lemma 2.11. *Let Ω be a signature graph and let $S \subseteq V(\Omega)$ be such that all edges in $E(S, \overline{S})$ have unit weight. Then we have* $\mathrm{Holant}(\Omega) = \mathrm{Holant}(\Omega{\downarrow_S})$.

Proof. Note that $E(\Omega)$ is partitioned into the subsets

$$M = E(S, \overline{S}), \quad X = E(\Omega[S]), \quad Y = E(\Omega - S).$$

Every vertex $v \in S$ is only incident with edges from $M \cup X$, while every vertex $v \notin S$ is only incident with edges from $M \cup Y$. Writing $\overline{S} = V(\Omega) \setminus S$ and $v^* = v{\downarrow_S}$, we recall that $V(\Omega{\downarrow_S}) = \{v^*\} \cup \overline{S}$ and $E(\Omega{\downarrow_S}) = M \cup Y$.

Let $w : E(\Omega) \to \mathbb{Q}$ denote the weight function of Ω and let f_v for $v \in V(\Omega)$ denote the vertex function at v. For sets $A \subseteq E(G)$, let us write $w(A) = \prod_{e \in A} w(e)$. Since $w(e) = 1$ for $e \in M$ by assumption, we may add edges from M to a set A without changing $w(A)$.

Then we obtain

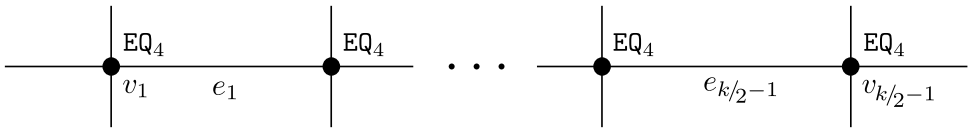
$$\begin{aligned}
\mathrm{Holant}(\Omega) &= \sum_{x\in\{0,1\}^{E(\Omega)}} w(x)\cdot\left(\prod_{v\in S} f_v(x|_{M\cup X})\right)\left(\prod_{v\in\overline{S}} f_v(x|_{M\cup Y})\right)\\
&= \sum_{z\in\{0,1\}^{M}}\left(\sum_{x\in\{0,1\}^{X}} w(zx)\cdot\prod_{v\in S} f_v(zx)\right)\left(\sum_{y\in\{0,1\}^{Y}} w(zy)\cdot\prod_{v\in\overline{S}} f_v(zy)\right)\\
&= \sum_{z\in\{0,1\}^{M}} f_{v^*}(z)\cdot\sum_{y\in\{0,1\}^{Y}} w(zy)\cdot\prod_{v\in\overline{S}} f_v(zy)\\
&= \sum_{zy\in\{0,1\}^{M\cup Y}}\prod_{v\in V(\Omega\downarrow_S)} w(zy)\cdot f_v(zy)\\
&= \mathrm{Holant}(\Omega\downarrow_S).
\end{aligned}$$

In the third equation, we used (2.7) from Definition 2.9. □

Realizing vertex functions by gates

We will often insert gates at vertices to simulate complicated signatures by simpler signatures. For instance, we can realize $\texttt{EQ}_k$ for any even arity $k \in \mathbb{N}$ by a gate whose vertices feature only the signature $\texttt{EQ}_4$. This will be useful in Section 2.2.

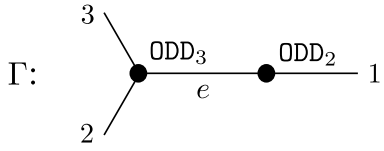
Example 2.12. For all even $k \geq 4$, there exists a gate $\Gamma_{\texttt{EQ}}$ with $\mathrm{Sig}(\Gamma_{\texttt{EQ}}) = \texttt{EQ}_k$. This gate consists of vertices $v_1, \ldots, v_{k/2-1}$, each equipped with $\texttt{EQ}_4$, internal edges $e_1, \ldots, e_{k/2-2}$ with $e_i = v_i v_{i+1}$ for all i, and k additional dangling edges.



To see $\mathrm{Sig}(\Gamma_{\texttt{EQ}}) = \texttt{EQ}_k$, let x be any satisfying assignment to $\Gamma_{\texttt{EQ}}$. By $\texttt{EQ}_4$ at v_1, the edges incident with v_1 (including e_1) all feature the same value in x, say $t \in \{0, 1\}$. Since e_2 is incident with v_2, and by $\texttt{EQ}_4$ at v_2, all edges incident with v_2 have value t in x. By induction, the same applies to all edges of $\Gamma_{\texttt{EQ}}$, and to its dangling edges in particular. □

Next we show that, given signatures $\texttt{ODD}_3$ and $\texttt{ODD}_2$, we can simulate $\texttt{EVEN}_3$.

Example 2.13. We can realize $\texttt{EVEN}_3$ as a gate Γ featuring signatures $\texttt{ODD}_3$ and $\texttt{ODD}_2$:



Note that indeed $\mathrm{Sig}(\Gamma) = \texttt{EVEN}_3$: If dangling edge 1 is active, then e is not active, and exactly one of 2 and 3 must be active. If edge 1 is not active, then e is active, and either both of 2 and 3 are active, or none are. This gives the satisfying assignments $101, 110, 011, 000$, which are precisely the bitstrings of length 3 with even Hamming weight. □

Using this, we can realize the signatures $\texttt{ODD}_k$ and $\texttt{EVEN}_k$ for any arity $k \geq 3$ from $\texttt{ODD}_3$ and $\texttt{ODD}_2$, noted in a similar way in [Val08, Theorem 3.3]. This will be required in the next section, and ultimately in Chapter 7.

Example 2.14. For all $k \geq 3$, there exists a gate $\Gamma_{\texttt{EVEN}}$ with $\mathrm{Sig}(\Gamma_{\texttt{EVEN}}) = \texttt{EVEN}_k$. It consists of vertices $v_1, \ldots, v_{k-2}$ equipped with $\texttt{EVEN}_3$, edges $e_1, \ldots, e_{k-3}$ connecting these vertices, and dangling edges $[k]$.

Let x be a satisfying assignment to $\Gamma_{\texttt{EVEN}}$. By $\texttt{EVEN}_3$ at v_1, we have $x(e_1) = x(1) \oplus x(2)$, where $\oplus$ denotes addition in $\mathbb{Z}/2\mathbb{Z}$. Likewise, we have $x(e_2) = x(e_1) \oplus x(3)$, so we obtain inductively that

$$x(e_{k-3}) = \bigoplus_{t=1}^{k-2} x(t). \tag{2.8}$$

Then $\texttt{EVEN}_3$ at v_{k-2} implies that

$$\begin{aligned} x(e_{k-3}) \oplus x(k-1) \oplus x(k) &\underset{(2.8)}{=} \left(\bigoplus_{t=1}^{k-2} x(t) \right) \oplus x(k-1) \oplus x(k) \\ &= \bigoplus_{t=1}^{k} x(t) = 0. \end{aligned}$$

Likewise, we can construct $\texttt{ODD}_k$ for all $k \geq 3$ by following the construction above, but replacing the signature $\texttt{EVEN}_3$ at v_{k-2} by $\texttt{ODD}_3$. □

2.2. Matchgates

Valiant's paper on holographic algorithms and Holants [Val08] focused on reductions from Holant problems to evaluations of PerfMatch(G) on graphs G obtained from Ω. This is interesting for at least two reasons: Firstly, if we can compute PerfMatch(G) efficiently, e.g., by invoking the FKT method on planar G, then we can also compute Holant(Ω) efficiently. Secondly, if we can show that the evaluation of Holant(Ω) is hard, then we obtain the same for PerfMatch(G), and structural properties of Ω might carry over to G.

Of course, the evaluation of Holant(Ω) could be reduced to other counting problems, such as counting cliques, vertex covers, or the evaluation of the Tutte polynomial. However, PerfMatch is a particularly good target, since it is a Holant problem itself, as shown in Example 2.3, and because nontrivial algorithms are known for PerfMatch. In later chapters, we will also reduce to other targets, namely to counting (not necessarily perfect) matchings and to counting colorful matchings in edge-colored graphs.

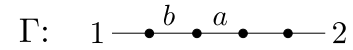
2.2.1. Basic definitions and first examples

The reduction to PerfMatch mentioned above proceeds via *matchgates*, i.e., gates that only feature the perfect matching signature $\mathtt{HW}_{=1}$ from Example 2.5.

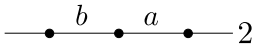
Definition 2.15 (adapted from [Val08])**.** A *matchgate* Γ (of arity $k \in \mathbb{N}$) is a gate with dangling edges D whose vertices use only the signature $\mathtt{HW}_{=1}$. We consider $D = [k]$. Note that Γ may be edge-weighted, but dangling edges must receive weight 1 by Definition 2.9. Also observe that, being a gate, Γ realizes a signature $\mathrm{Sig}(\Gamma) : \{0,1\}^{[k]} \to \mathbb{Q}$.

Consider the following matchgates from [Val08, Proposition 6.1] as first examples.

Example 2.16. For $a, b \in \mathbb{Q}$, we define the following matchgates Γ and Γ' of arity 2 that realize the signatures $\mathrm{Sig}(\Gamma)$ and $\mathrm{Sig}(\Gamma')$ of type $\{0,1\}^{[2]} \to \mathbb{Q}$.

Γ: 

$$\mathrm{Sig}(\Gamma, x) = \begin{cases} a & \text{if } x = 11, \\ b & \text{if } x = 00, \\ 0 & \text{otherwise.} \end{cases}$$

Γ': 

$$\mathrm{Sig}(\Gamma', x) = \begin{cases} b & \text{if } x = 01, \\ a & \text{if } x = 10, \\ 0 & \text{otherwise.} \end{cases}$$

The signatures can be computed easily by hand. For instance, we have $\mathrm{Sig}(\Gamma, 11) = a$ since there is only one assignment $y \in \{0,1\}^{E(\Gamma)}$ that extends $x = 11$: In this assignment, the edge of weight a and the dangling edges 1 and 2 are active. The other values can be verified likewise.

Since matchgates are simply gates, we can use them to simulate signatures, as seen in Section 2.1.3. The benefit of matchgates as opposed to general gates lies in the fact that they enable a simple reduction to PerfMatch.

Fact 2.17 (Realizing signature graphs using matchgates)**.** *Let Ω be a signature graph. If there is a matchgate Γ_v with $\mathrm{Sig}(\Gamma_v) = f_v$ for every vertex $v \in V(\Omega)$, then we can insert Γ_v as a gate at v, as specified in Definition 2.10, and we obtain a signature graph that uses only edge-weights and the signature $\mathtt{HW}_{=1}$.*

In other words, we obtain a graph $G = G(\Omega)$ on $\sum_v |V(\Gamma_v)|$ vertices and $\sum_v |E(\Gamma_v)|$ edges that realizes *Ω, that is,*

$$\mathrm{Holant}(\Omega) = \mathrm{PerfMatch}(G(\Omega)).$$

The signature of a matchgate Γ of arity $d \in \mathbb{N}$ can be computed by counting perfect matchings in induced subgraphs of Γ: For an assignment $x : \{0,1\}^{[d]} \to \mathbb{Q}$, if a vertex $v \in V(\Gamma)$ is matched by an active dangling edge in x, then it cannot be matched by a non-dangling edge of Γ, so we could as well delete v when computing $\mathrm{Sig}(\Gamma, x)$.

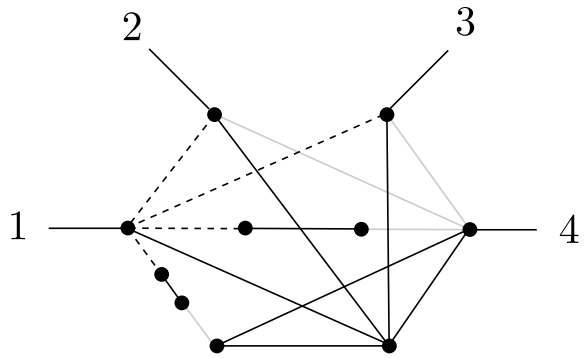


Figure 2.2.: The equality matchgate $\Gamma_=$ that realizes $\texttt{EQ}_4$. Edges of weight -1, $\frac{1}{2}$ and 1 are shown gray, dashed and black, respectively. A similar matchgate is shown in [BD07], but it realizes only a weighted version of the signature $\texttt{EQ}_4$.

Fact 2.18. *Let Γ be a matchgate of arity $d \in \mathbb{N}$. Given $x \in \{0,1\}^{[d]}$, let $S(x) \subseteq V(\Gamma)$ denote the vertices of Γ that are incident with active dangling edges in x. Write Γ' for the graph obtained from Γ by deleting all dangling edges. Then* $\mathrm{Sig}(\Gamma, x) = \mathrm{PerfMatch}(\Gamma' - S(x))$.

Since only graphs with an even number of vertices admit perfect matchings, Fact 2.18 implies the following parity condition on signatures of matchgates.

Fact 2.19 (Parity Condition)**.** *If a signature $f : \{0,1\}^{[d]} \to \mathbb{Q}$ can be realized by a matchgate, then at least one of the following holds:*

- *For all $x \in \{0,1\}^{[d]}$ with odd* $\mathrm{hw}(x)$*, we have $f(x) = 0$. Then we call f* even.
- *For all $x \in \{0,1\}^{[d]}$ with even* $\mathrm{hw}(x)$*, we have $f(x) = 0$. Then we call f* odd.

As seen for gates in Examples 2.12 and 2.14, we can use matchgates for simple signatures to construct matchgates for more complex signatures. In the following, we use this observation to realize the signature $\texttt{EQ}_k$ for *even* $k \in \mathbb{N}$ from the matchgate $\Gamma_=$ for $\texttt{EQ}_4$ shown in Figure 2.2. We discovered $\Gamma_=$ using a computer algebra system; this process is described in more detail in Appendix C. Note that $\texttt{EQ}_k$ for odd $k \in \mathbb{N}$ does not satisfy the parity condition and can hence not be realized by matchgates.

Lemma 2.20. *For all even $k \in \mathbb{N}$, there is a matchgate realizing $\texttt{EQ}_k$, which features only $-1, 1, \frac{1}{2}$ as edge-weights. This matchgate has $\mathcal{O}(k)$ vertices and edges.*

Proof. For $k = 2$, we use Example 2.16 with $a = b = 1$. For $k = 4$, it is verified in Appendix C.1 that the matchgate $\Gamma_=$ in Figure 2.2 realizes $\texttt{EQ}_4$. For $k > 4$, we use the gate from Example 2.12 and realize each occurrence of $\texttt{EQ}_4$ by $\Gamma_=$. □

We first focus on *planar* matchgates, which are relevant in Part I. In Section 2.2.3, we then prove that *every* signature satisfying the parity condition can be realized by (not necessarily planar) matchgates. This will yield a new #P-completeness proof of PerfMatch.

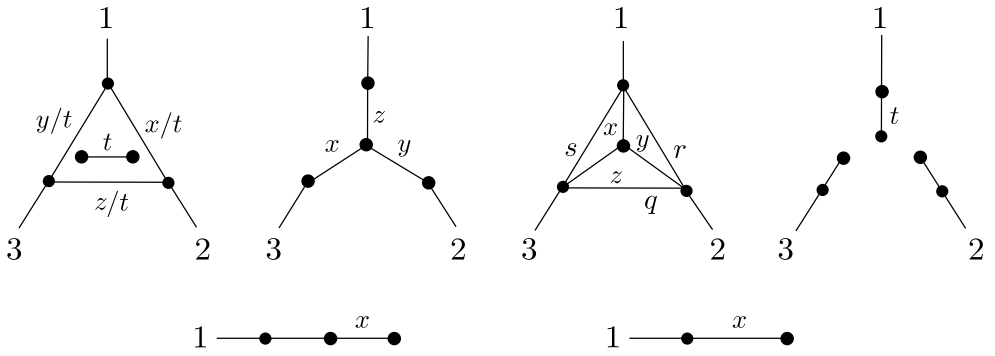


Figure 2.3.: The matchgates from Propositions 6.1 and 6.2 in [Val08], used in Lemma 2.23.

2.2.2. Planar matchgates

In the paper that introduced Holant problems [Val08], Valiant focused on *planar* matchgates; this allowed him to invoke the FKT method for planar PerfMatch, seen in Theorem 1.27, to find unexpected polynomial-time algorithms for counting problems.

Definition 2.21. A matchgate Γ is *planar* if it admits a planar drawing with all of its dangling edges on the outer face. Furthermore, a clockwise traversal of the outer face starting at dangling edge 1 must encounter the dangling edges $1, \dots, d$ in this order. We call a signature f planar if it can be realized by a planar matchgate.

Note that Valiant uses the notion of matchgates to refer to *planar* matchgates, whereas we do not impose this restriction upon general matchgates. If a signature graph Ω is planar and contains only planar signatures, then we can find a planar graph $G(\Omega)$ realizing Ω. This observation clearly generalizes to signature graphs of bounded genus. For the following lemma, please recall the notion $\sigma_v f_v$ from Definition 2.4.

Lemma 2.22. *Let Ω be a signature graph that admits a drawing π on a surface S of genus $\gamma \in \mathbb{N}$. For each $v \in V(\Omega)$, let σ_v denote the clockwise order of $I(v)$ around v, as specified by π, and assume that there exists a planar matchgate Γ_v realizing $\sigma_v f_v$. Then the graph $G(\Omega)$ that realizes Ω as in Fact 2.17 admits a drawing on the same surface S.*

Proof. We extend π and the drawings of the matchgates $\{\Gamma_v \mid v \in V(\Omega)\}$ to a drawing of $G(\Omega)$. At $v \in V(\Omega)$, the embedding of Ω can be locally extended by inserting Γ_v at v: If v is incident with the edges $e_1, \dots, e_t$, as ordered by the embedding π of Ω, then the insertion of Γ_v attaches e_i to the i-th dangling edge of Γ_v. Since Γ_v admits a planar drawing in which the dangling edges $1, \dots, t$ appear on the outer face in that order, these extensions are compatible with the drawing of Ω. □

It was shown in [Val08] that every signature of arity ≤ 3 that satisfies the parity condition is planar. We will later see that there exist non-planar signatures of arity 4.

Lemma 2.23 ([Val08])**.** *If $f : \{0,1\}^{[d]} \to \mathbb{Q}$ with $d \leq 3$ is even or odd, then f is planar.*

Proof sketch. Figure 2.3, taken from [Val08], shows several matchgates, each drawn as a plane graph. If f is even or odd and has arity 1 or 3, then it can be verified that at least one planar matchgate Γ satisfies $\mathrm{Sig}(\Gamma) = f$ after appropriate substitution of edge-weights. As an example, for $x, y, z, t \in \mathbb{Q}$ with $t \neq 0$, the signature

$$f(x) = \begin{cases} z & \text{if } x = 001, \\ y & \text{if } x = 010, \\ x & \text{if } x = 100, \\ t & \text{if } x = 111. \end{cases} \tag{2.9}$$

is realized by the upper left matchgate. If $t = 0$, then the matchgate to the right of it applies. If f has arity 2, then a matchgate from Example 2.16 applies. □

Using this lemma, we can also construct planar matchgates for certain signatures of arbitrarily large arity, as the following lemma shows for ODD and EVEN, noted also in [Val08]. This will be relevant for Sections 2.3 and 4.3.

Lemma 2.24. *For all $k \in \mathbb{N}$, the signatures $\mathtt{ODD}_k$ and $\mathtt{EVEN}_k$ are planar. Furthermore, both signatures can be realized by planar matchgates that feature no edge-weights.*

Proof. Using Example 2.12, we can realize $\mathtt{ODD}_k$ and $\mathtt{EVEN}_k$ as planar gates containing only the signatures $\mathtt{ODD}_3$ and $\mathtt{ODD}_2$, both of which are planar by Lemma 2.23.

Observe that the signature $\mathtt{ODD}_3$ is realized by the matchgate shown on the top left of Figure 2.3 after substitution of 1 for all edge-weights. The signature $\mathtt{ODD}_2$ is realized with the matchgate from Example 2.16 without edge-weights. □

To conclude this section, we mention a known non-planar signature, namely the Boolean signature CROSS of arity 4, which is defined by

$$\mathtt{CROSS}: \quad (x_1, x_2, x_3, x_4) \mapsto \begin{cases} 1 & \text{if } x_1 = x_3 \wedge x_2 = x_4, \\ 0 & \text{otherwise.} \end{cases}$$

This signature is not planar, although it admits a matchgate that can be drawn in the plane with all dangling edges on the outer face, e.g., by using two copies of the matchgate Γ from Example 2.16 that realizes $\mathtt{EQ}_2$. However, in a planar drawing of this resulting matchgate, the dangling edges are not in the correct order. In fact, there are two good reasons for CROSS not to be planar according to our definition:

1. It is known that a signature f is planar if and only if f satisfies the so-called *matchgate identities* [CG14]. For signatures f with the support of CROSS, these can be stated succinctly as
$$f(0000) \cdot f(1111) = -f(0101) \cdot f(1010). \tag{2.10}$$
This is clearly not satisfied by CROSS, so the signature is not planar.

2. The signature CROSS can be used to transform non-planar signature graphs Ω to planar versions Ω' with $\text{Holant}(\Omega) = \text{Holant}(\Omega')$. Given a drawing of Ω with crossings, we place a new vertex at each crossing, incident with the four edges involved in the crossing in clockwise order. Attach CROSS to this vertex to obtain Ω'. If CROSS were planar, this would allow for a polynomial-time reduction from PerfMatch to planar PerfMatch, but since this latter problem can be solved by the FKT method (Theorem 1.27) in polynomial time, we would have established $\mathsf{FP} = \#\mathsf{P}$.

2.2.3. Matchgates for all possible signatures

We turn our attention back to matchgates that are not necessarily planar. The parity condition from Fact 2.19 tells us that every signature of a matchgate must be even or odd. It is natural to ask whether, conversely, every even or odd signature admits a matchgate. The positive answer to this question we give in this subsection does not only shed light upon the expressivity of general matchgates versus planar matchgates, but it also allows us later to reduce *all* Holant problems uniformly to PerfMatch.

Lemma 2.25. *Let $f : \{0,1\}^{[d]} \to \mathbb{Q}$ be a signature of arity $d \in \mathbb{N}$ that is even or odd. Then there is a matchgate Γ that realizes f. Furthermore,*

- *Γ has $\mathcal{O}(|\text{supp}(f)| \cdot d)$ vertices and edges, and satisfies $\Delta(\Gamma) \leq |\text{supp}(f)| + \mathcal{O}(1)$,*
- *the edge-weights of Γ are contained in $\text{im}(f) \cup \{-1, 1/2, 1\}$, where $\text{im}(f)$ denotes the image of f, and*
- *given as input $\{(x, f(x)) \mid x \in \text{supp}(f)\}$, we can construct Γ in time $\mathcal{O}(|\text{supp}(f)| \cdot d)$.*

Note that $|V(\Gamma)|$ depends polynomially upon d and $|\text{supp}(f)|$, but since $|\text{supp}(f)|$ may be as large as 2^{d-1}, we may obtain matchgates on $\Omega(2^d d)$ vertices. We also observe that Γ features the additional weights -1 and $\frac{1}{2}$, and we will later see why they are necessary.

The construction of Γ resembles the construction of a formula in disjunctive normal form from a given Boolean function. For each element $x \in \text{supp}(f)$, we create an assignment gate L_x that tests whether the dangling edges of Γ are assigned x. This will ensure that $\text{Sig}(\Gamma, x) = f(x)$ for all $x \in \text{supp}(f)$. Furthermore, we ensure that $\text{Sig}(\Gamma, x) \neq 0$ holds only if exactly one of these tests succeeds, that is, if $x \in \text{supp}(f)$.

Proof of Lemma 2.25. For this proof, we assume that $d - \text{hw}(x)$ is odd for every $x \in \text{supp}(f)$, which implies that exactly one of d and f is even, while the other one is odd. If $d - \text{hw}(x)$ is even, the proof proceeds similarly.

We first define the following gate Γ' on dangling edges $[d]$, shown exemplarily in Figure 2.4. The matchgate Γ is then obtained by realizing all signatures appearing in Γ' by matchgates: This is trivial for $\text{HW}_{=1}$, and concerning EQ, we will ensure that all vertices featuring EQ have even degree, so we can realize these signatures with Lemma 2.20.

1. Create vertices $O = \{o_1, \ldots, o_d\}$ with signature $\text{HW}_{=1}$, and for $i \in [d]$, add the dangling edge i and make it incident to o_i.

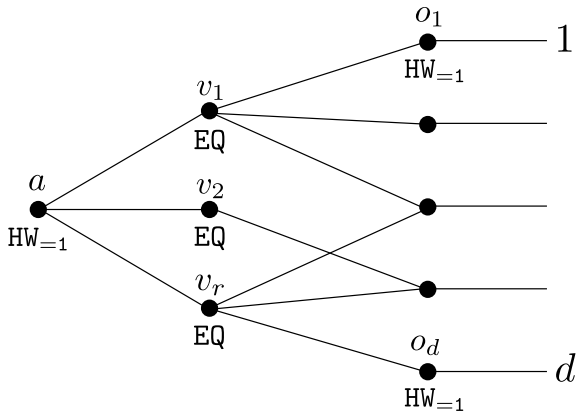


Figure 2.4.: The gate Γ' constructed in the proof of Lemma 2.25. In this example, Γ' realizes the signature f that maps $\{00011, 11101, 11000\}$ to 1 and all other inputs to 0.

2. Create a vertex a with signature $\mathtt{HW}_{=1}$.

3. Let $\mathrm{supp}(f) = \{x_1, \ldots, x_r\}$ for some $r \in \mathbb{N}$. For each $\kappa \in [r]$, let

$$S_\kappa = O \setminus \{o_i \mid i \in x_\kappa\}.$$

Note that $|S_\kappa|$ is odd by assumption. We perform the following steps.

a) Create a vertex v_κ with signature $\mathtt{EQ}_{|S_\kappa|+1}$ and make it adjacent to all vertices in S_κ. Note that $|S_\kappa| + 1$ is even, so $\mathtt{EQ}_{|S_\kappa|+1}$ can be realized by a matchgate by Lemma 2.25.

b) Draw an edge of weight $f(x_\kappa)$ from v_κ to a.

We prove that Γ' realizes f. Let $y \in \{0,1\}^{E(\Gamma')}$ be a satisfying assignment. By $\mathtt{HW}_{=1}$ at the vertex a and $\mathtt{EQ}$ at v_κ for $\kappa \in [r]$, there is exactly one $\kappa \in [r]$ such that all edges of $I(v_\kappa)$ are active under y, while all edges in $I(v_{\kappa'})$ for $\kappa' \neq \kappa$ are inactive. In particular, we then have

$$\mathrm{val}_{\Gamma'}(y) \cdot w_{\Gamma'}(y) = f(x_\kappa). \tag{2.11}$$

Let $x = y|_{[d]}$ be the restriction of y to the dangling edges of Γ'. We observe that, if the edges in $I(v_\kappa)$ are active under y, for $\kappa \in [r]$, then $x = x_\kappa$: Since y is satisfying, by $\mathtt{HW}_{=1}$ at O, every $o_i \in O$ for $i \in [d]$ is incident with exactly one active edge, and this edge must be dangling if $x(i) = 1$, or contained in $I(v_k)$ if $x(i) = 0$.

Hence, for every $x \in \{0,1\}^{[d]}$, there is a satisfying assignment y of Γ' that extends x if and only if $x = x_\kappa$ for some $\kappa \in [s]$. Furthermore, in this case, y is unique and satisfies (2.11). We conclude that $\mathrm{Sig}(\Gamma', x) = f(x)$. □

Remark 2.26. For a general signature of arity d, the construction above yields a matchgate on up to $\Omega(2^d d)$ vertices. In some applications, this exponential size is prohibitive, but smaller matchgates might still exist.

Finally, using Lemma 2.25, we can obtain a simple uniform method for reducing arbitrary Holant problems to PerfMatch. To this end, we simply "double" signatures that are not even, in the sense specified below.

Definition 2.27. For $d \in \mathbb{N}$, let $f : \{0,1\}^{[d]} \to \mathbb{Q}$ be a signature. Then we define $f_{double} : \{0,1\}^{[2d]} \to \mathbb{Q}$ as the following signature:

$$f_{double}(x) = \begin{cases} f(x_1 \dots x_d) & \text{if } x = x_1 x_1 x_2 x_2 \dots x_d x_d \text{ for } x_1, \dots, x_d \in \{0,1\}, \\ 0 & \text{otherwise.} \end{cases}$$

It is obvious that f_{double} is an even signature, for every choice of f.

Theorem 2.28. *Every signature graph Ω with vertex functions $\{f_v\}_{v \in V(\Omega)}$ can be realized by a graph G (whose vertices all feature the signature $\mathtt{HW}_{=1}$) on*

$$\mathcal{O}\left(\sum_{v \in V(\Omega)} |\mathrm{supp}(f_v)| \cdot \deg(v)\right)$$

vertices and edges. Furthermore, the maximum degree $\Delta(G)$ of G is bounded by the maximum of a universal constant, and $|\mathrm{supp}(f_v)|$, and $\deg(v)$ for all $v \in V(\Omega)$. The edge-weights of G are

$$\{1, -1, 1/2\} \cup \bigcup_{v \in V(\Omega)} \mathrm{im}(f_v),$$

and given Ω as input, we can construct G in time $\mathcal{O}(|V(G)| + |E(G)|)$.

Proof. We first construct a derived signature graph Ω' with $\mathrm{Holant}(\Omega) = \mathrm{Holant}(\Omega')$ that features only even vertex functions:

1. Subdivide each edge $e \in E(\Omega)$ into a subdivision vertex s_e and two subdivision edges. Replace each subdivision edge by two parallel edges to obtain edges $e^{(i)}$ for $i \in [4]$. Then place $\mathtt{EQ}_4$ at s_e. If e has weight $w(e)$ in $E(\Omega)$, then assign this weight to an arbitrary edge among $e^{(i)}$, say $e^{(1)}$.

2. For every non-subdivision vertex $v \in V(\Omega')$, order $I_{\Omega'}(v)$ such that the two edges replacing an edge $e \in E(\Omega)$ from $I_\Omega(v)$ appear consecutively, and at the place specified by the order of e in $I_\Omega(v)$. Then replace $f = f_v$ with f_{double}.

All signatures in Ω' are even and can thus be realized by matchgates via Lemma 2.25. It remains to prove $\mathrm{Holant}(\Omega) = \mathrm{Holant}(\Omega')$.

For all satisfying assignments $x' \in \{0,1\}^{E(\Omega')}$ and all $e \in E(\Omega)$, the four values assigned to $e^{(i)}$ for $i \in [4]$ under x' agree by $\mathtt{EQ}$ at the subdivision vertex s_e, and we can hence "contract" x' to a uniquely corresponding assignment $x \in \{0,1\}^{E(\Omega)}$. By the definition of f_{double} at non-subdivision vertices, and by our choice of edge-weights in Ω', we have that $\mathrm{val}_{\Omega'}(x') \cdot w_{\Omega'}(x') = \mathrm{val}_\Omega(x) \cdot w_\Omega(x)$. This proves the claim. □

Since we observed in Example 2.7 that #SAT is a Holant problem, we can reduce it to the evaluation of PerfMatch and hence obtain #P-hardness of the latter problem.

Lemma 2.29. *Let φ be a d-CNF formula with n variables and m clauses such that every variable occurs in at most r clauses, for $n, m, d, r \in \mathbb{N}$. Then we can construct a graph G on $\mathcal{O}(rn + 2^d dm)$ vertices and edges, with maximum degree $\max\{r, 2^d - 1, \mathcal{O}(1)\}$, and edge-weights $\{-1, \frac{1}{2}, 1\}$ such that*

$$\#\mathsf{SAT}(\varphi) = \mathrm{PerfMatch}(G).$$

Proof. As in Example 2.7, we construct a signature graph Ω on $n + m$ vertices and dm edges with $\#\mathsf{SAT}(\varphi) = \mathrm{Holant}(\Omega)$. Each of the m clause signatures has degree $\leq d$ and support $2^d - 1$. Each variable vertex has degree $\leq r$, the sum of degrees of the variable signatures in Ω is dm, and each signature has support 2. Using Theorem 2.28, we obtain a graph G' that realizes Ω and satisfies the claimed size and degree bounds. □

By reduction from 3-$\#\mathsf{SAT}$, we obtain $\#\mathsf{P}$-hardness for $\mathrm{PerfMatch}^{-1,\frac{1}{2},1}$, and a lower bound under $\#\mathsf{ETH}$. For the lower bound, note that Lemma 2.29 maps formulas in 3-CNF with n variables and m clauses to graphs with $\mathcal{O}(n + m)$ vertices and edges.

Corollary 2.30. *The problem $\mathrm{PerfMatch}^{-1,\frac{1}{2},1}$ is $\#\mathsf{P}$-complete under parsimonious reductions, as introduced in Definition 1.2. Furthermore, it admits no $2^{o(m)}$ time algorithm on graphs with m edges, unless $\#\mathsf{ETH}$ fails.*

Using removal of the weight $\frac{1}{2}$, we can reduce $\mathrm{PerfMatch}^{-1,\frac{1}{2},1}$ to $\mathrm{PerfMatch}^{-1,1}$. To this end, we use Remark 1.30 to transform the graph G obtained from Lemma 2.29 to a graph G' with edge-weights -1 and 1 such that

$$\#\mathsf{SAT}(\varphi) = \mathrm{PerfMatch}(G) = 2^{-\frac{|V(G)|}{2}} \cdot \mathrm{PerfMatch}(G'). \tag{2.12}$$

Corollary 2.31. *The problem $\mathrm{PerfMatch}^{-1,1}$ is $\#\mathsf{P}$-complete under weakly parsimonious polynomial-time reductions. Furthermore, it admits no $2^{o(m)}$ time algorithm on graphs with m edges, unless $\#\mathsf{ETH}$ fails.*

Together with univariate interpolation as in Lemma 1.37, this implies the following.

Corollary 2.32. *The problem $\mathrm{PerfMatch}^{0,1}$ on unweighted graphs is $\#\mathsf{P}$-complete under polynomial-time Turing reductions that use $\mathcal{O}(n)$ oracle calls and incur quadratic blowup.*

In Chapter 7, we will discuss a novel and more efficient way of getting from Corollary 2.31 to Corollary 2.32, which has further implications to structural complexity theory.

Why the equality matchgate needs weights

We conclude Section 2.2 with an observation about the edge-weights $1/2$ and -1 added by Theorem 2.28, which ultimately stem from the matchgate $\Gamma_=$ for $\mathtt{EQ}_4$ from Figure 2.2. When reducing $\mathrm{Holant}(\Omega)$ for a signature graph Ω with Boolean signatures to $\mathrm{PerfMatch}(G)$, they introduce weights into G that are not present in Ω, and this might seem like a nuisance. However, both weights are required under standard assumptions in complexity theory. To see this, consider the reduction from $\#\mathsf{SAT}$ to PerfMatch from Lemma 2.29.

- Unless $\mathsf{P} = \oplus\mathsf{P}$, the edge-weight $1/2$ or the simulation of the weight $1/2$ by a factor $\frac{1}{2^t}$ as in Remark 1.30 is necessary: Since (2.12) can be rewritten as

$$2^{\frac{|V(G)|}{2}} \cdot \#\mathsf{SAT}(\varphi) = \mathrm{PerfMatch}(G), \tag{2.13}$$

  an algorithm for $\oplus$PerfMatch does not imply an algorithm for $\oplus\mathsf{SAT}$. This makes sense since $\oplus$PerfMatch can be reduced to the determinant, as mentioned in Section 1.3, while $\oplus\mathsf{SAT}$ is $\oplus\mathsf{P}$-complete. In fact, even a matchgate for $\mathtt{EQ}_4$ where the edge-weight $1/2$ is replaced with $1/3$ would imply $\mathsf{P} = \oplus\mathsf{P}$ as well.

- The weight -1 is needed unless $\mathsf{P} = \mathsf{NP}$: If only the weight $1/2$ were present, then (2.13) would imply a weakly parsimonious reduction from $\#\mathsf{SAT}$ to PerfMatch on unweighted graphs, and $\mathsf{SAT} \in \mathsf{P}$ would follow as in Section 1.2.

2.3. Combined signatures

In the final section of this chapter, we introduce the notion of *combined signatures*, an extremely simple idea with surprisingly useful applications to parameterized counting complexity that will we used in all parts of this thesis. A "combined" signature is nothing but a signature that can be expressed as a point-wise linear combination of other signatures.

Definition 2.33. Let $f = c_1 \cdot f_1 + \ldots + c_t \cdot f_t$ be a signature, where $c_1, \ldots, c_t \in \mathbb{Q}$ are coefficients and $f_1, \ldots, f_t$ are signatures, and the linear combination is to be understood point-wise. Then we say that f is t-combined from the constituents $f_1, \ldots, f_t$.

Combined signatures will be useful in the following setting: Assume we would like to realize a signature f in a signature graph Ω via matchgates or other gates, but (must) fail to do so, e.g., because f does not satisfy the parity condition. Instead, we can express f as a combined signature whose constituents do admit matchgates, and then use a simple observation to compute $\mathrm{Holant}(\Omega)$ as a linear combination of Holants where all involved signatures admit matchgates.

Lemma 2.34. *Let Ω be a signature graph and let $w \in V(\Omega)$ be an arbitrary fixed vertex with signature f_w. Let $g_1, \ldots, g_t$ be signatures and $c_1, \ldots, c_t \in \mathbb{Q}$ be coefficients such that*

$$f_w = \sum_{i=1}^{t} c_i \cdot g_i.$$

For $i \in [t]$, let Ω_i be the signature graph obtained from Ω by replacing f_w with g_i. Then

$$\mathrm{Holant}(\Omega) = \sum_{i=1}^{t} c_i \cdot \mathrm{Holant}(\Omega_i).$$

Proof. By elementary manipulations, we have

$$\begin{aligned}\operatorname{Holant}(\Omega) &= \sum_{x\in\{0,1\}^{E(\Omega)}} f_w(x) \cdot \prod_{v\in V(\Omega)\setminus\{w\}} f_v(x) \\ &= \sum_{x\in\{0,1\}^{E(\Omega)}} \left(\sum_{i=1}^{t} c_i \cdot g_i(x)\right) \cdot \prod_{v\in V(\Omega)\setminus\{w\}} f_v(x) \\ &= \sum_{i=1}^{t} \sum_{x\in\{0,1\}^{E(\Omega)}} c_i \cdot g_i(x) \cdot \prod_{v\in V(\Omega)\setminus\{w\}} f_v(x) \\ &= \sum_{i=1}^{t} c_i \cdot \operatorname{Holant}(\Omega_i).\end{aligned}$$

This proves the claim. □

In Lemma 2.34, we considered only signature graphs featuring *one* combined signature; this can of course be extended to the case of several combined signatures.

Lemma 2.35 (Combined Signature Lemma)**.** *Let Ω be a signature graph, let $k, t \in \mathbb{N}$ and let $w_1, \dots, w_k$ be distinct vertices of Ω such that the following holds: For all $\kappa \in [k]$, the signature f_κ at w_κ admits coefficients $c_{\kappa,1}, \dots, c_{\kappa,t} \in \mathbb{Q}$ and signatures $g_{\kappa,1}, \dots, g_{\kappa,t}$ such that*

$$f_\kappa = \sum_{i=1}^{t} c_{\kappa,i} \cdot g_{\kappa,i}.$$

Given a tuple $\theta \in [t]^k$, let Ω_θ be defined by replacing, for each $\kappa \in [k]$, the vertex function f_κ at w_κ with $g_{\kappa,\theta(\kappa)}$. Then

$$\operatorname{Holant}(\Omega) = \sum_{\theta\in[t]^k} \left(\prod_{\kappa=1}^{k} c_{\kappa,\theta(\kappa)}\right) \cdot \operatorname{Holant}(\Omega_\theta). \tag{2.14}$$

Proof. Apply Lemma 2.34 inductively for $w_1, \dots, w_k$. Each step reduces the number of combined signatures by one, and elementary algebraic manipulations imply (2.14). □

When using Lemma 2.35 for positive results, then the right-hand side of (2.14) is "easy", in the sense that the values $\operatorname{Holant}(\Omega_\theta)$ for all θ can be obtained, e.g., by reduction to planar PerfMatch and the FKT method. In the same way, Lemma 2.35 also allows us to prove hardness results under Turing reductions: In this case, the left-hand side is "hard" and could be computed from oracle access to the values $\operatorname{Holant}(\Omega_\theta)$ for all θ.

Remark 2.36. For later use, we remark that Lemma 2.35 generalizes to restricted versions of Holants that sum only over assignments from a specified set: Let Ω be a signature graph,

let $X \subseteq \{0,1\}^{E(\Omega)}$, and define

$$\mathrm{Holant}_X(\Omega) = \sum_{a \in X} w_\Omega(a) \cdot \mathrm{val}_\Omega(a).$$

Then Lemma 2.34, and hence Lemma 2.35, still hold if we replace every occurrence of $\mathrm{Holant}(\Omega)$ by $\mathrm{Holant}_X(\Omega)$. We will use this later in the context of *edge-colorful* Holants.

2.3.1. Combined signatures with planar constituents

As mentioned before, we are particularly lucky if the constituents of a combined signature do not only admit matchgates, but these are even planar: If a signature graph Ω is planar and contains only k non-planar signatures, which are combined from t (explicitly known) planar signatures, then we can reduce $\mathrm{Holant}(\Omega)$ to t^k instances of planar PerfMatch. This generalizes naturally to graphs of bounded genus.

Lemma 2.37. *Let Ω be a signature graph of genus γ with planar vertex functions, except for k functions that are combined from t planar signatures, for some $k, t \in \mathbb{N}$. Let the coefficients and constituents of these combined signatures be given as part of the input. Then* $\mathrm{Holant}(\Omega)$ *can be computed in time $t^k 4^\gamma n^{\mathcal{O}(1)}$.*

Proof. Using Lemma 2.35, reduce $\mathrm{Holant}(\Omega)$ to a linear combination of $\mathrm{Holant}(\Omega_\theta)$ for $\theta \in [t]^k$. Each Ω_θ has genus γ and features only planar signatures, so it can be realized by a graph G_θ of the same genus by Lemma 2.22. We determine $\mathrm{PerfMatch}(G_\theta)$ in time $4^\gamma n^{\mathcal{O}(1)}$ via Theorem 1.28, obtaining the claimed total runtime. □

In the following, we give an example for a signature with planar constituents, which will be used later to reduce a certain parameterized version of MatchSum on planar graphs to planar PerfMatch. Recall that MatchSum is #P-hard on planar graphs by Theorem 1.24.

Lemma 2.38. *For $t \in \mathbb{N}$ and $\mathbf{c} \in \mathbb{Q}^t$, define the t-ary signature $\mathtt{PROD}_\mathbf{c}$ by*

$$\mathtt{PROD}_\mathbf{c} : \quad x \mapsto \prod_{i \in x} c_i.$$

This signature is 2-combined from planar constituents. In particular, by choosing $\mathbf{c}$ as the all-ones vector $\mathbf{1}$ of length t, we observe that the t-ary signature

$$\mathtt{OMIT}_t := \mathtt{PROD}_\mathbf{1}$$

is 2-combined from planar constituents. Note that OMIT *yields 1 on all inputs.*

Proof. Recall that $\mathtt{ODD}_t$ and $\mathtt{EVEN}_t$ are planar by Lemma 2.24. It is clear that

$$\mathtt{OMIT}_t = \mathtt{EVEN}_t + \mathtt{ODD}_t.$$

To obtain $\mathtt{PROD}_\mathbf{c}$, we construct the following gate Γ: Create a vertex v that is equipped with $\mathtt{OMIT}_t$ and connected to vertices $w_1, \ldots, w_t$, where w_i for $i \in [t]$ is equipped with

$\texttt{EDGE}_{c(i)}$, as defined in Lemma 2.8, and is incident with dangling edge i. It is easily seen that $\mathrm{Sig}(\Gamma) = \texttt{PROD}_{\mathbf{c}}$. To obtain two planar matchgates from Γ, substitute $\texttt{OMIT}_t$ once by $\texttt{EVEN}_t$ and once by $\texttt{ODD}_t$ and realize the resulting gates via Lemma 2.24. □

The signature OMIT on its own is rather useless as it cannot discriminate between inputs. However, we observe in the following that OMIT can prove useful when it comes to "absorbing" incoming active edges.

2.3.2. From MatchSum to PerfMatch

To provide a first algorithmic application of combined signatures, we show how to compute $\mathrm{MatchSum}(G)$ for planar vertex-weighted graphs G that contain k distinguished faces such that every vertex of G with non-zero weight is contained in at least one of these faces. In other words, we count matchings in which only vertices in the distinguished faces may be unmatched. For later use, we first prove an auxiliary lemma where faces are replaced by arbitrary subsets.

Lemma 2.39. *Let G be a vertex-weighted graph and let $V_1, \dots, V_k \subseteq V(G)$ be such that for every $v \in V(G)$ with $w(v) \neq 0$, there is exactly one $i \in [k]$ with $v \in V_i$. Define a signature graph $\Phi = \Phi(G)$ as follows:*

1. *For each $i \in [k]$, add a* sink vertex *s_i to G and connect it to the vertices $v_1, \dots, v_t$ of V_i, in this order.*
2. *Attach $\texttt{PROD}_{\mathbf{c}}$ to s_i, where $\mathbf{c} = (w(v_1), \dots, w(v_t))$.*
3. *Remove all vertex weights and place $\texttt{HW}_{=1}$ at all non-sink vertices.*

Then we have $\mathrm{MatchSum}(G) = \mathrm{Holant}(\Phi)$.

Proof. For every matching $M \in \mathcal{M}[G]$, let $w(M) = \prod_{v \in \mathrm{usat}(M)} w(v)$. If $w(M) \neq 0$, then we have $\mathrm{usat}(M) \subseteq V_1 \cup \ldots \cup V_k$. Thus, we can uniquely extend M to an assignment $x = x(M)$ with $x \in \{0,1\}^{E(\Phi)}$ and $\mathrm{val}_\Phi(x) = w(M)$: For every unmatched vertex $v \in \mathrm{usat}(M)$ with $v \in V_i$ for some $i \in [k]$, include the edge from v to the sink vertex s_i. Then $\texttt{HW}_{=1}$ at all non-sink vertices yields value 1, while the signatures PROD at sink vertices together yield the value $w(M)$.

Likewise, when restricted to non-sink vertices, every satisfying assignment $x \in \{0,1\}^{E(\Phi)}$ induces a unique matching $M = M(x)$ with $w(M) \neq 0$. □

For use in Chapter 7, we note the following:

Remark 2.40. Let $\Phi = \Phi(G)$ be as in Lemma 2.39. With Lemma 2.35, we obtain

$$\mathrm{Holant}(\Phi) = \sum_{\theta \in [2]^k} \mathrm{Holant}(\Phi_\theta), \tag{2.15}$$

where Φ_θ is obtained from Φ by realizing copies of PROD by its constituents. Recall the constituents ODD and EVEN of OMIT, used by PROD in Lemma 2.38, and let us say that

$\theta \in [2]^k$ is *odd* if Φ_θ features an odd number of ODD signatures. Likewise, we call θ *even* if Φ_θ features an even number of ODD signatures.

If $|V(G)|$ is even and θ is odd, then we can observe that $\text{Holant}(\Phi_\theta) = 0$: The proof of Lemma 2.39 shows that $\text{Holant}(\Phi_\theta)$ ranges over the matchings $M \in \mathcal{M}[G]$ with an odd number of defects. But since $|V(G)|$ is even, no such matchings exist. A symmetric argument applies in the case of odd $|V(G)|$.

Hence, if $|V(G)|$ is even, we may safely restrict the right-hand side of (2.15) to even $\theta \in [2]^k$. If $|V(G)|$ is odd, then we may restrict the sum to odd $\theta \in [2]^k$.

We conclude this subsection with an fpt-algorithm for evaluating MatchSum on planar graphs, parameterized by the number of faces that contain vertices of non-zero weight. This will be useful in Section 4.3.

Theorem 2.41. *Let G be a vertex-weighted graph that is given together with a drawing on a surface of genus γ, and assume that there are faces $F_1, \dots, F_k$ that contain all vertices with non-zero weight. Then we can compute* $\text{MatchSum}(G)$ *in time $\mathcal{O}(2^k \cdot 4^\gamma \cdot n^{1.5})$.*

Proof. Let B_i denote the boundary of face F_i in the drawing of G. We first create a partition $\mathcal{B} = \{B'_1, \dots, B'_k\}$ of $\bigcup_{i\in[k]} B_i$ such that $B'_i \subseteq B_i$ for $i \in [k]$ and $B'_i \cap B'_j = \emptyset$ for $i \neq j$. This can be achieved trivially by assigning each vertex that occurs in several sets B_i to exactly one B_i in an arbitrary fashion.

From Lemma 2.39, we obtain a signature graph $\Phi = \Phi(G)$ with sink vertices $s_1, \dots, s_k$ corresponding to the sets $B_1, \dots B_k$, which satisfies

$$\text{MatchSum}(G) = \text{Holant}(\Phi).$$

We extend the drawing of G to one of Φ by drawing each sink vertex s_i for $i \in [k]$ into the face F_i. Note that this is possible since s_i is only adjacent with B_i. Since Φ has genus γ and features only k occurrences of PROD, which is 2-combined from planar constituents, we can compute $\text{Holant}(\Phi)$ in time $2^k \cdot 4^\gamma \cdot n^{\mathcal{O}(1)}$ by Corollary 2.37. □

Concluding notes

Holant problems, signature graphs [Val08, CL07] and planar matchgates [CG14] are well-studied objects, see also [CLX09b, KC10, CHL10, CLX11]. Outside of computer science, they were studied in the literature on quantum physics and linear algebra under the name of *tensor networks* [Lan12]. For instance, the problem #SAT is reformulated as a tensor network in [BMT14] in a way that parallels our Example 2.7. An algorithm for PerfMatch with running time $\mathcal{O}(4^\gamma n^3)$ on graphs of genus γ, similar to Theorem 1.28, was also shown in that framework [Bra08].

Our notion of gates parallels that of *$\mathcal{F}$-gates* [CLX08], from which we also borrow the notion of dangling edges. In the lingo of tensor networks, dangling edges are dangling *wires*, and contractions are a very prominent operation.

Section 2.3 is based on joint work with Mingji Xia from 2013. To the best of the author's knowledge, the equality matchgate from Figure 2.2 and the content of Sections 2.2.3 and

2.3 are novel, with the exception of the following, pointed out by Tyson Williams:

- A trick similar to "doubling signatures" as in Definition 2.27 was already considered for a different application [GW13] under the name of "domain pairing".
- Linear combinations of signatures were already considered for other purposes in [GLV13], and in particular, Lemma 6.2 in that paper parallels our Lemma 2.35. However, connections to parameterized complexity are not studied in that paper.

Signature graphs are called signature *grids* in the literature on Holant problems. We decided to depart from this term, because we will later occasionally work with signature graphs that are *actual* grids, and we see some potential for confusion here.[3]

After submission of this thesis, the concepts introduced in this chapter gave rise to further applications (apart from those shown in the later parts of the thesis):

- Combined signatures have been used to obtain a simplified presentation of the $\mathcal{O}(4^\gamma n^3)$ time algorithm for PerfMatch on graphs of genus γ, as well as a $\oplus\mathsf{W}[1]$-hardness proof for the permanent modulo 2^k. Both results will appear in [CX].
- In unpublished joint work with Dániel Marx, the results of Section 2.2.3 were used to prove lower bounds for PerfMatch under various parameters, assuming #ETH and #SETH.

[3]Since Holant problems were initially only studied on *planar* signature graphs, it made more sense to speak of signature grids at that time.

Part I.

Counting perfect matchings in H-minor-free graphs

Introduction to Part I

In this part, we study the problem of counting perfect matchings in a graph G, parameterized by structural parameters of G, namely the Hadwiger number and variants thereof. Recall from Section 1.5.3 that the Hadwiger number $\mathrm{hadw}(G)$ of a graph G denotes the size of a maximum clique minor in G.

The relevance of counting perfect matchings and of evaluating the permanent was already discussed in earlier chapters, and we have reproven a #P-hardness result for PerfMatch in Section 2.2.3. Despite this hardness result, which actually holds even on bipartite graphs of maximum degree 3 by Theorem 1.3, there are nontrivial "tractable" graph classes $\mathcal{C}$ such that $\mathrm{PerfMatch}(G)$ can be evaluated in polynomial time on graphs $G \in \mathcal{C}$. Let us briefly recapitulate the positive results known in the literature, some of which were already mentioned in Section 1.3.2. To provide some historical context, we list the results with their publication dates.

1961/67: It was shown in the area of statistical physics that PerfMatch can be evaluated in time $\mathcal{O}(n^{1.5})$ for planar graphs, see Theorem 1.27. This algorithm was later dubbed the *FKT method*, for their inventors Fisher, Kasteleyn and Temperley [Kas61, TF61, Kas67]. The algorithm first applied only to certain grids, and was only later extended to *arbitrary* planar graphs.

1974/89: The FKT method was generalized to graphs excluding the minor $K_{3,3}$. These do not necessarily have to exclude K_5 as well, as is the case for planar graphs. The first algorithm for $K_{3,3}$-free graphs was shown by Little [Lit74], and it was later parallelized to an NC-algorithm by Vazirani [Vaz89].

1998: Another generalization of the FKT method was shown [GL98], which is orthogonal to the $K_{3,3}$-free case: The problem PerfMatch admits an $\mathcal{O}(4^\gamma n^3)$ algorithm on graphs of genus γ. We address the notion of orthogonality later.

folklore: Classical dynamic programming on the tree decompositions gives an $2^{\mathcal{O}(k)}n$ time algorithm on graphs of treewidth k, as seen in Corollary 1.45.

2014: Very recently, and independently of this work, a parallel polynomial-time algorithm for computing PerfMatch on graphs excluding the minor K_5 was obtained in [STW14]. This generalizes the FKT method and is orthogonal to the bounded-genus and the $K_{3,3}$-free cases.

2006: After submitting this thesis, an algorithm by Makowsky et al. [MRAG06] came to the author's attention. This algorithm evaluates the matching polynomial

$\mu(\xi)$ at arbitrary $\xi \in \mathbb{Q}$ in time $n^{\mathcal{O}(k)}$ on graphs of clique-width k. The clique-width is a graph parameter that generalizes treewidth, and in particular, complete graphs have clique-width 2. On *unweighted* graphs G, this algorithm can be used to compute $\mu(G;0) = \text{PerfMatch}(G)$. It does however not apply for general edge-weighted graphs (and could be used to prove $\mathsf{FP} = \#\mathsf{P}$ otherwise).

Note that algorithms for evaluating PerfMatch on restricted graph classes have a history that dates back over 50 years and predates the field of computational complexity. We observe that, apart from the algorithm for graphs of bounded clique-width, all of the positive results produced in this time are related by a common theme that was seemingly not remarked before:

Almost every *known* tractable graph class for PerfMatch excludes *some* fixed minor.

Indeed, the planar graphs exclude both $K_{3,3}$ and K_5. The $K_{3,3}$-free graphs and the K_5-free graphs exclude a minor by definition. The graphs of treewidth k exclude the minor K_{k+2} by Lemmas 1.47 and 1.48, and by the same argument, they actually even exclude the $k \times k$ square grid. Finally, the graphs of genus at most k exclude a sufficiently large clique, and actually even a fixed 1-apex graph H, that is, a graph that can be drawn in the plane after removal of a single vertex [Epp00].

Addressing the "almost" in the above statement, note that the class of complete graphs excludes no fixed minor (every graph is the subgraph of some complete graph), but the number of perfect matchings in complete graphs admits a closed formula. Since complete graphs have cliquewidth 2, this is however subsumed by the algorithm on graphs of bounded clique-width from [MRAG06]. Furthermore, we recall that the general problem PerfMatch on *weighted* graphs from such classes does not necessarily admit a polynomial-time algorithm. We will therefore treat the classes of bounded clique-width as an exception.

In the following, we will focus on graph classes that exclude a fixed minor. For such classes, all known algorithms for evaluating PerfMatch can be seen to crucially exploit building blocks from the Graph Structure Theorem introduced in Section 1.5:

- For graphs of bounded treewidth or genus, this is clear.
- Concerning the graphs excluding $K_{3,3}$ or $K_{5,5}$, it was shown in [Wag37] that such graphs admit tree decompositions whose torsos are either planar or have bounded size, and this can be interpreted as a precursor to the Graph Structure Theorem. Algorithmically, this decomposition was used in the $K_{3,3}$-free case to extend Pfaffian orientations of the torsos to an orientation of the entire graph, and hence reduce PerfMatch to the determinant as in the FKT method. Recall the proof of Theorem 1.27 for the notion of Pfaffian orientations.
- Interestingly, the K_5-free case can *not* be solved via Pfaffian orientations, since $K_{3,3}$ is K_5-free, but does not admit a Pfaffian orientation [MRST97]. To obtain an algorithm, the authors of [STW14] introduce a different trick, which we independently discovered and present in Chapter 3. Our approach also yields a more general result.

Having identified the importance of graph minors for the evaluation of PerfMatch, it is very natural to ask whether we can make the jump from evaluating PerfMatch for classes excluding *specific* fixed minors, such as $K_{3,3}$ or K_5, to classes excluding *arbitrary* fixed minors. This is precisely what we attempt to do in this part, and we formulate the following research question:

Problem 2.1. For every *fixed* graph H, is there a polynomial-time algorithm for PerfMatch on graphs from Excl$[H]$? Slightly stronger, is PerfMatch/hadw contained in XP?

Conversely, can we prove lower bounds on the running times of such algorithms, or can we find graphs H such that PerfMatch is #P-complete on $\mathcal{C}[H]$?

In this part, we will present partial answers to these questions, but let us first make two remarks: Firstly, we consider *perfect* matchings rather than matchings that are not necessarily perfect. By Theorem 1.24, counting matchings is already #P-complete on planar graphs, and this rules out an algorithm on *general* H-minor free graphs.

Secondly, we could also consider classes excluding a fixed *topological* minor H, which is a weaker requirement than excluding H as a *minor*. However, this is not useful: If $\Delta(H) \leq 3$, then any graph G contains H as a minor iff G contains H as a topological minor. Otherwise, if $\Delta(H) \geq 4$, then all graphs of maximum degree 3 exclude H as a topological minor, and we recall that PerfMatch is #P-complete on such graphs by Theorem 1.3. Hence, every positive result we could obtain on classes excluding *topological* minors H, namely if $\Delta(H) \leq 3$, is already covered by results on classes excluding *minors*.

Single-crossing minors

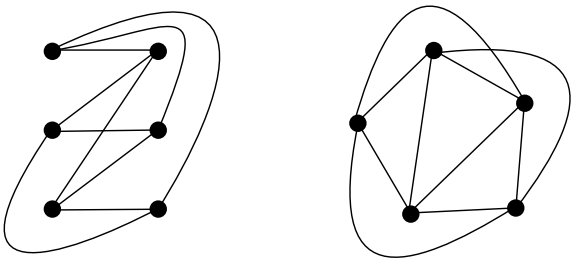
In Chapter 3, we give a partial answer to the algorithmic question posed in Problem 2.1. Let us call a graph H *single-crossing* if H can be drawn in the plane with at most one crossing. We show that, for every fixed single-crossing graph H, the problem PerfMatch can be solved in polynomial time on graphs from Excl$[H]$.

Theorem 3.1. *For any fixed single-crossing graph H, we can compute* PerfMatch(G) *in time $\mathcal{O}(n^4)$ on graphs $G \in$ Excl$[H]$.*

It should be stressed that the excluded minor H rather than G is required to be single-crossing: An algorithm for PerfMatch(G) on single-crossing graphs G would be rather underwhelming as it could be derived easily by reduction to the planar case.

Theorem 3.1 unifies and generalizes the algorithms for bounded-treewidth graphs, $K_{3,3}$-free graphs and K_5-free graphs: The graphs $K_{3,3}$ and K_5 are single-crossing by the drawings below, and as mentioned before, the graphs of treewidth k exclude the $k \times k$ grid, which

is even a planar graph. In fact, our algorithm uses an algorithm for bounded-treewidth graphs as a subroutine.



Graphs excluding a fixed single-crossing minor H have already been studied in other contexts, and our proof of Theorem 3.1 relies on some of these results: By a theorem of Robertson and Seymour [RS93], the graphs in Excl[H] for single-crossing H can be decomposed into planar graphs and graphs of bounded treewidth, and it was shown by Demaine et al. [DHN$^+$04] how to compute such decompositions. The same authors also described approximation algorithms for the treewidth and other invariants of such graphs [DHT02, DHN$^+$04, DHT05], and Chambers and Eppstein [CE13] developed an $\mathcal{O}(n \log n)$ algorithm for computing maximum flows on such graphs.

We also observe that Theorem 3.1 is *orthogonal* to the bounded-genus case from Theorem 1.28: The graph consisting of n disjoint copies of K_5 can be seen to have unbounded genus, but it excludes $K_{3,3}$ as a minor. Thus, Theorem 3.1 applies for this graph, while the algorithm for bounded-genus graphs does not. Conversely, the class of torus-embeddable graphs contains all single-crossing graphs, as we can always embed one of the crossing edges on the "handle". Thus, the algorithm for bounded-genus graphs applies here, while Theorem 3.1 does not.

Our algorithm for PerfMatch on graphs that exclude a fixed single-crossing minor proceeds by reduction to PerfMatch on graphs that are planar or have bounded treewidth, treating these problems as black boxes. In particular, the theory of Pfaffian orientations will not be exploited *within* our algorithm, as opposed to the approaches undertaken in [Lit74, Vaz89] for $K_{3,3}$-free and in [GL98, Tes00] for bounded-genus graphs. If an alternative algorithm for planar PerfMatch were found that circumvents the theory of Pfaffian orientations, then the same would follow for our algorithm. This allows for a simple algorithm that fits nicely into the framework introduced in Section 2.

Lower bounds

In view of the Graph Structure Theorem for not necessarily single-crossing graphs H, it is natural to ask whether our approach for PerfMatch can be extended to such graphs as well. To repeat the algorithmic question in Problem 2.1, is there, for *every* fixed H, a polynomial-time algorithm for PerfMatch on graphs from Excl[H]?

We could not find an answer to this general question yet, due to our failure to handle the *vortices* arising in the Graph Structure Theorem. In Chapter 4, we will however see that, if such polynomial-time algorithms exist (and can be uniformly constructed) for every

fixed H, then their exponent *must* depend on H, thus providing a lower bound in the sense of Problem 2.1. In other words, we prove that PerfMatch/hadw is #W[1]-hard.

To show this, we actually prove the stronger statement that PerfMatch/apex is #W[1]-hard. Recall that apex(G), first introduced in Section 1.2.2, denotes the minimum size of a set $A \subseteq V(G)$ such that $G - A$ is planar. Apices are one of the several building blocks allowed by the Graph Structure Theorem, and we have mentioned in Remark 1.53 that hadw(G) can be bounded by a function of apex(G) for all graphs G. Hence, a hardness result for PerfMatch under the apex number also implies one under the Hadwiger number.

Theorem 4.1. *The problem* PerfMatch^{0,1}/apex *is* #W[1]*-hard.*

Corollary 4.1. *The problem* PerfMatch^{0,1}/hadw *is* #W[1]*-hard.*

Note that this #W[1]-hardness result contrasts Theorem 3.1, where an $f(H)n^4$ algorithm was obtained for each fixed single-crossing graph H. We will revisit this soon and define an appropriate parameter that allows us to express that PerfMatch is fixed-parameter tractable when parameterized by excluded single-crossing minors.

However, let us first note that we strengthen Theorem 4.1 in Section 4.2 by proving that counting k-defect matchings in planar graphs is #W[1]-hard. Recall that the k-defect matchings in a graph G are those matchings that leave exactly k vertices unmatched. It is easily observed that these matchings can be counted in time $n^{k+\mathcal{O}(1)}$ on planar graphs by summing over the quantities PerfMatch($G - S$) for all k-subsets S of $V(G)$, where each term can be computed by the FKT method (see Theorem 1.27). The following result complements this naive algorithm:

Theorem 4.10. *Given a planar graph G and a number $k \in \mathbb{N}$ as inputs, the problem* #PlanarDualMatch *of counting the k-defect matchings in G is* #W[1]*-hard.*

Note that the parameterized complexity of counting k-defect matchings is not interesting on *general* graphs: Already for $k = 0$, this amounts to the #P-complete problem of counting perfect matchings on general graphs. On a different note, in planar graphs and in graphs of constant maximum degree, the dual problem of counting k-matchings, with k *edges* rather than defects, is fixed-parameter tractable by a general meta-theorem for counting models of first-order logic formulas on graphs of bounded local treewidth [FG01, Fri04].

Generalized Hadwiger numbers

After the short digression into implications of Theorem 4.1, we return to our main problem of evaluating PerfMatch in graphs excluding fixed minors. Note that, in Corollary 4.2, we no

longer considered the evaluation of PerfMatch on classes Excl[H] for fixed H, but we rather went one step further and asked about the *parameterized* complexity of PerfMatch/hadw. This can be put into a more general framework: Rather than considering PerfMatch only on classes Excl[H] for fixed H, we can also consider its parameterized complexity under generalized versions of the Hadwiger number.

Definition. Given an infinite graph class $\mathcal{D}$ and a graph G, let

$$\mathrm{hadw}_{\mathcal{D}}(G) = \min\{|V(H)| \mid H : H \in \mathcal{D} \wedge H \not\preceq G\}.$$

Given another graph class $\mathcal{D}'$, let us write $\mathcal{D} \preceq \mathcal{D}'$ if, for all $H \in \mathcal{D}$, there is some $H' \in \mathcal{D}'$ with $H \preceq H'$.

The relation $\preceq$ on graph classes defined above is clearly transitive and reflexive, and it induces the following ordering on the parameters $\mathrm{hadw}_{\mathcal{D}}$:

Lemma. *If $\mathcal{D}, \mathcal{D}'$ are recursively enumerable graph classes with $\mathcal{D} \preceq \mathcal{D}'$, then there is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that*

$$\mathrm{hadw}_{\mathcal{D}'}(G) \leq f(\mathrm{hadw}_{\mathcal{D}}(G)).$$

Proof. Let $\mathrm{hadw}_{\mathcal{D}}(G) = k$, so there exists some $H \in \mathcal{D}$ on k vertices with $H \not\preceq G$. Since $\mathcal{D} \preceq \mathcal{D}'$, there is some graph $H' \in \mathcal{D}'$ satisfying $H \preceq H'$, which implies that $H' \not\preceq G$. The graph H' can be found by enumerating all $H' \in \mathcal{D}'$ and testing whether $H \preceq H'$ holds by brute force, and hence, we have $|V(H')| \leq f(k)$ for a computable function f. This implies the bound $\mathrm{hadw}_{\mathcal{D}'}(G) \leq |V(H')| \leq f(k)$. □

That is, if two graph classes $\mathcal{D}$ and $\mathcal{D}'$ satisfy $\mathcal{D} \preceq \mathcal{D}'$, then tractability results for $\mathrm{hadw}_{\mathcal{D}'}$ are stronger than results for $\mathrm{hadw}_{\mathcal{D}}$. A maximal element among these classes is $\mathcal{K}$, the set of complete graphs: Since $\mathcal{D} \preceq \mathcal{K}$ holds for all classes $\mathcal{D}$, tractability results for $\mathrm{hadw}_{\mathcal{K}}$ are the strongest results that could hypothetically be obtained, but they are ruled out for PerfMatch by Corollary 4.2 unless $\mathsf{FPT} = \#\mathsf{W}[1]$.

We can thus only hope for tractability results under the parameter $\mathrm{hadw}_{\mathcal{D}}$ for classes $\mathcal{D}$ with $\mathcal{K} \not\preceq \mathcal{D}$. One such parameter could be $\mathrm{hadw}_{\mathcal{CR}1}$, where $\mathcal{CR}1$ denotes the class of single-crossing graphs, and indeed, we can state Theorem 3.1 in a slightly stronger form.

Theorem 3.2. *The problem* PerfMatch/$\mathrm{hadw}_{\mathcal{CR}1}$ *is fpt.*

We would like to know where $\#\mathsf{W}[1]$-hardness sets in between the parameters $\mathrm{hadw}_{\mathcal{CR}1}$ and $\mathrm{hadw}_{\mathcal{K}}$, and hence ask whether there is a class $\mathcal{D}$ with $\mathcal{K} \not\preceq \mathcal{D}$ such that the problem PerfMatch/$\mathrm{hadw}_{\mathcal{D}}$ is $\#\mathsf{W}[1]$-complete. It will turn out that our proof of Theorem 4.1 fails to provide such a class $\mathcal{D}$: More precisely, if $\mathcal{D}$ denotes the class of output graphs produced by the reduction, then we will explicitly note that $\mathcal{K} \preceq \mathcal{D}$ holds.

In an attempt to extend the algorithm from Theorem 3.2, we complement Theorem 4.1 by an algorithm on certain restricted k-apex graphs: If G is a graph with k apices that see at most s faces of the underlying planar graph, then we can determine PerfMatch(G) in time $f(k,s) \cdot n^{\mathcal{O}(1)}$. In fact, the same holds when the underlying graph has bounded genus.

Theorem 4.21. *Given as input a graph G, a set $A \subseteq V(G)$, and a drawing of $G - A$ on a surface of genus γ such that A is adjacent to at most s faces, we can compute* PerfMatch(G) *in time $f(|A|, \gamma, s) \cdot n^3$ for a computable function f.*

This setting is motivated by a decomposition theorem for graphs G excluding a fixed 1-apex minor H: As shown in [DHK09], such graphs G admit decompositions similar to those described by the Graph Structure Theorem, with the additional restriction that apices of torsos can connect only to the underlying faces that contain vortices, under a slightly generalized notion of vortices. Since each torso features only $f(H)$ vortices, the apices of G can hence see only $f(H)$ faces, for some computable function f. With a leap of faith, and assuming that we can eventually "solve vortices" in fixed-parameter time, we can hence interpret Theorem 4.21 as suggesting an fpt-algorithm for PerfMatch/hadw$_{\mathcal{A}1}$, where $\mathcal{A}1$ is the class of 1-apex graphs.[4]

Notes

The proof of Theorem 4.1 from Section 4.1 is joint work with Mingji Xia, partly carried out when the author was visiting him at ISCAS Beijing in 2013. This work is to appear in [CX]. All other results were obtained independently.

The author conjectured Theorem 4.10 already in 2010, and a connection to graph minors was suggested to the author in the same year by Dániel Marx and Holger Dell at the Dagstuhl Seminar 10481 *"Computational Counting"*.

The algorithm for PerfMatch on graphs excluding a K_5-minor from [STW14] uses ideas that are similar to our approach in Theorem 3.2. Our work was obtained independently, as stated in the conclusion section of that paper, and we have published our result shortly after as an arXiv preprint [Cur14]. Note that our results generalize those in [STW14].

The immense use of the problem #GridTiling for reductions to planar-ish problems was only clear to the author after visiting Dániel Marx in 2013 at MTA SZTAKI, Budapest.

[4] While this is not explained in the present thesis, as long as vortices are not present, clique-sums of bounded adhesion can be performed efficiently for the problem PerfMatch.

3. Excluding a single-crossing minor

In this chapter, we prove Theorem 3.1 by presenting, for every fixed single-crossing graph H, a polynomial-time algorithm for PerfMatch(G) on graphs $G \in \text{Excl}[H]$.

Theorem 3.1 (restated from page 81)**.** *For any fixed single-crossing graph H, we can compute* PerfMatch(G) *in time $\mathcal{O}(n^4)$ on graphs $G \in \text{Excl}[H]$.*

In fact, we prove the stronger statement of Theorem 3.2, which states that the problem PerfMatch/hadw$_{\mathcal{CR}1}$ is fpt. Our algorithm essentially proceeds by standard dynamic programming, augmented with a trick that allows to store the results of the subproblems needed to solve G within G itself as "remnant" vertices. A similar trick was used for computing maximum flows in graphs exlcuding a fixed single-crossing minor [CE13].

Theorem 3.2 (restated from page 84)**.** *The problem* PerfMatch/hadw$_{\mathcal{CR}1}$ *is fpt.*

A crucial ingredient in the algorithm is a variant of Theorem 1.52 that describes the structure of graphs $G \in \text{Excl}[H]$ when H is single-crossing. This variant, shown in [RS91], asserts that all such graphs G can be decomposed into planar torsos and torsos of bounded treewidth, using tree decompositions of adhesion at most 3. Please recall Definition 1.51 for these notions. Furthermore, it was shown in [DHN$^+$04] that such decompositions can be found in time $f(H)n^4$ for a computable function f. For our purposes, we need to refine these decompositions slightly by imposing a certain *triangle condition* upon planar torsos, in a way that is similar to [CE13].

Theorem 3.3. *There is a computable function f such that, on input a single-crossing graph H and a graph $G \in \text{Excl}[H]$, we can compute a tree decomposition $\mathcal{T} = (T, \mathcal{B})$ of G with $\mathcal{B} = (B_t)_{t \in V(T)}$ in time $f(H)n^4$ that satisfies the following properties: The adhesion of $\mathcal{T}$ is bounded by* 3*, and for each node $t \in V(T)$, the torso G_t satisfies*

- $\text{tw}(G_t) \leq f(H)$*, or*
- *G_t is given as a plane graph with the following* triangle condition*: If s is a neighbor of t in T and $|B_s \cap B_t| = 3$ holds, then $B_s \cap B_t$ bounds a face in G_t.*

Proof. By [DHN$^+$04, Theorem 2], we can find a tree decomposition $\mathcal{T}$ of G in time $f(H) \cdot n^4$ that satisfies the conditions, except that planar torsos do not need to obey the triangle condition. We split such torsos (given as plane graphs) along triangles to fix $\mathcal{T}$.

Let $t \in V(T)$ be a node with planar torso G_t and let s be a neighbor of t, with $K = B_s \cap B_t$ of size 3 that does *not* bound a face in the plane graph G_t. Then the interior of K in the drawing of G_t contains an induced subgraph F of G_t such that $F - K$ is non-empty.

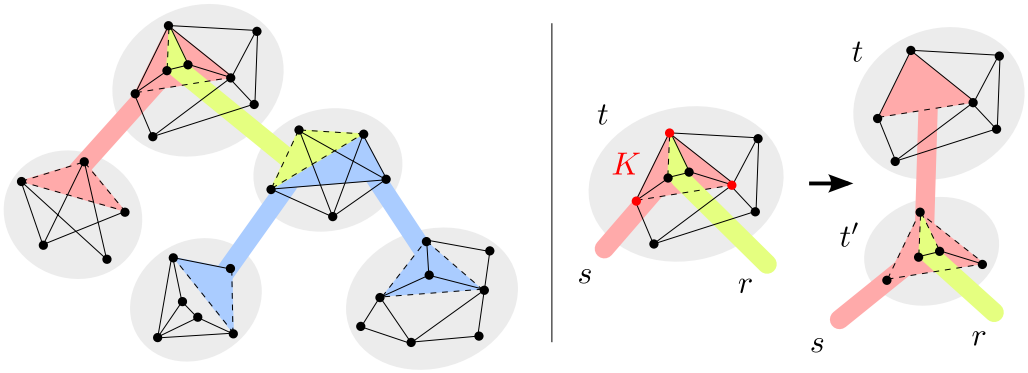


Figure 3.1.: *(left)* A tree decomposition $\mathcal{T}$ of G. At $t \in V(T)$, the torso G_t is shown. Edges that are not present in G itself are dashed. Subsets $B_s \cap B_t$ for $st \in E(T)$ are colored. *(right)* The splitting operation used for Theorem 3.3.

See the right part of Figure 3.1 for an example, where F is drawn red. Create a node t' in T, declare $B_{t'} = V(F)$ and add the edge tt' to T. For every neighbor r of t with $B_t \cap B_r \subseteq V(F)$, replace the edge tr in T by $t'r$. Then delete $V(F) \setminus K$ from B_t. It is easily checked that the resulting tree decomposition is still a tree decomposition of G.

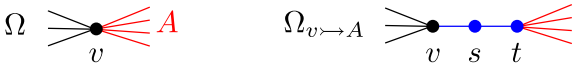
Repeat this process as long as there is a planar torso G_t that violates the triangle condition. Each step decreases the *total* number of triangles that violate the triangle condition (over all torsos G_t) by 1. Since planar graphs have $\mathcal{O}(n)$ triangles by Euler's formula, this process stops in time $\mathcal{O}(|V(T)| \cdot n)$. □

We can view [DHN+04, Theorem 2], which yields our initial unrefined tree decomposition, as a successor to Wagner's theorem [Wag37], which gives a similar result for the K_5-free graphs, and which in turn may be considered a follow-up result on Kuratowski's theorem [Kur30]. It should also be noted that, for specific single-crossing minors such as $K_{3,3}$ or K_5, the initial decomposition $\mathcal{T}$ required in the proof of Theorem 3.3 can be found in time $\mathcal{O}(n)$ instead of $\mathcal{O}(n^4)$, as shown in [Asa85] for $K_{3,3}$ and [RL08] for K_5.

We use Theorem 3.3 as follows: Given a graph G with $\mathrm{hadw}_{\mathcal{CR}1}(G) = k$, for $k \in \mathbb{N}$, start enumerating the single-crossing graphs H by non-decreasing number of vertices. For each graph H, test whether $H \preceq G$ holds via Theorem 1.39. Once the first test fails, stop and observe that $|V(H)| = k$ holds. Then invoke Theorem 3.3 on H and G to obtain the tree decomposition $\mathcal{T}$ of G in time $g(k) \cdot n^4$ for some computable function g.

Let us consider G as an edge-weighted signature graph Ω that features the signature $\mathtt{HW}_{=1}$ at each vertex. Of course, the tree decomposition $\mathcal{T}$ of G also applies to Ω. As in Example 2.3, we obtain $\mathrm{PerfMatch}(G) = \mathrm{Holant}(\Omega)$, and we will work with this signature graph Ω from now on. In particular, our algorithm will rely upon the following two elementary operations on signature graphs.

Fact 3.4. *Given a vertex $v \in V(\Omega)$ with signature $\mathtt{HW}_{=1}$ and a subset $A \subseteq I(v)$, let $\Omega_{v \mapsto A}$ be defined from Ω as shown below: First, add vertices s and t with signatures $\mathtt{HW}_{=1}$ to Ω, and edges vs and st, shown blue. Secondly, replace each edge vw in A, for $w \in V(\Omega)$, by the edge tw. Then* $\mathrm{Holant}(\Omega) = \mathrm{Holant}(\Omega_{v \mapsto A})$.



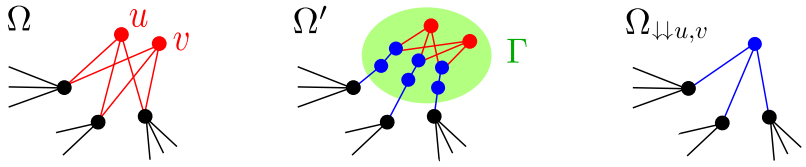
Proof. Consider the gate induced by $\{v, s, t\}$ in $\Omega' = \Omega_{v \mapsto A}$. Its signature is $\mathtt{HW}_{=1}$, similar to Example 2.16. We have $\Omega' \downarrow_{\{v,s,t\}} = \Omega$, so Proposition 2.11 implies the claim. □

The operation described in Fact 3.4 can be used to "collapse" two vertices to one.

Fact 3.5. *Let $u, v \in V(\Omega)$ feature arbitrary signatures, but no incident parallel edges, and assume that all neighbors of u and all neighbors of v feature the signature $\mathtt{HW}_{=1}$.*

Then we can replace u, v by a new vertex w, adjacent to $X = N(u) \cup N(v)$, without incident parallel edges, and we can compute a signature for w in time $2^{\mathcal{O}(|X|)}$ to obtain a signature graph $\Omega_{\downarrow\downarrow u,v}$ with $\mathrm{Holant}(\Omega) = \mathrm{Holant}(\Omega_{\downarrow\downarrow u,v})$.

Proof. Proceed as in the following picture, where X is black and u, v are red.



That is, for each $x \in X$, let $J(x) := \{xu, xv\} \cap E(\Omega)$, and successively replace Ω by $\Omega_{x \mapsto J(x)}$. This adds vertices to Ω, shown blue, and does not change $\mathrm{Holant}(\Omega)$ by Fact 3.4.

Consider the gate Γ induced by u, v and the added vertices. Since no parallel edges are present, we can compute $\mathrm{Sig}(\Gamma)$ by brute force in time $2^{\mathcal{O}(|X|)}$. Define $\Omega_{\downarrow\downarrow u,v} = \Omega \downarrow_\Gamma$, declare w as the vertex that replaces Γ in $\Omega \downarrow_\Gamma$, and observe that Lemma 2.11 about gate contractions implies $\mathrm{Holant}(\Omega) = \mathrm{Holant}(\Omega_{\downarrow\downarrow u,v})$. □

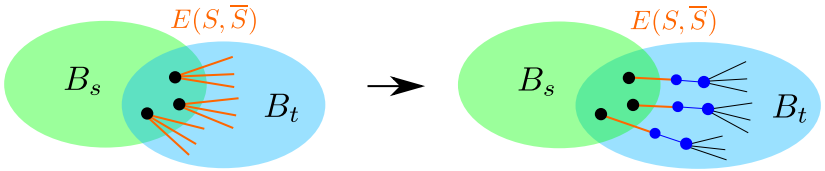
In order to compute $\mathrm{Holant}(\Omega)$, we present an algorithm that modifies Ω and $\mathcal{T}$ iteratively in a bottom-up manner. At each step, we delete a leaf t from $\mathcal{T}$ by contracting its associated bag in Ω to a so-called *remnant vertex*: This is simply a vertex of degree at most 3 that may feature a signature other than $\mathtt{HW}_{=1}$ that can be realized by a matchgate. At any given time in the execution, we denote the set of such remnant vertices by $\mathcal{R} \subseteq V(\Omega)$, and we note that $\mathcal{R} = \emptyset$ holds prior to execution. Furthermore, we maintain the following four invariants during execution of the algorithm:

1. All modifications made on Ω preserve $\mathrm{Holant}(\Omega)$.
2. After each step, $\mathcal{T}$ is a tree decomposition of $\Omega - \mathcal{R}$ satisfying the conditions of Theorem 3.3. In particular, $|B_s \cap B_t| \leq 3$ holds for all $st \in E(T)$.
3. The neighborhood of each $r \in \mathcal{R}$ is contained in $B_s \cap B_t$ for some $st \in E(T)$. No parallel edges are incident with r. This implies $\deg(r) \leq 3$ by Invariant 2.
4. All vertices in $\Omega - \mathcal{R}$ feature the signature $\mathtt{HW}_{=1}$, while vertices $r \in \mathcal{R}$ may feature arbitrary signatures that can be realized by *matchgates*.

All invariants hold trivially before execution of the algorithm. We then repeat the following steps as long as $|V(T)| \geq 1$ holds.

1. Pick a leaf $t \in V(T)$ and let s denote its parent. In the following steps, we will process t in a way that will ultimately lead to its deletion. Let $S = B_t \setminus B_s$, and if t is the only node left in $\mathcal{T}$, let $S = B_t$.

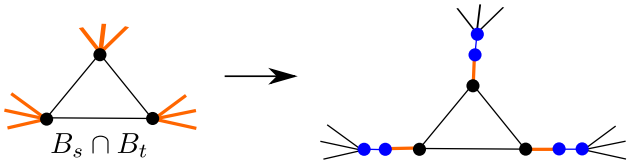
2. Let $E(S, \overline{S}) = \{uv \in E(\Omega) \mid u \in S, v \notin S\}$. By Lemma 1.46, we actually have that $E(S, \overline{S}) = \{uv \in E(\Omega) \mid u \in S, v \in B_s \cap B_t\}$. We wish to contract S to a remnant vertex, but to ensure Invariant 3, we first need to preprocess Ω and S to avoid introducing parallel edges.



   a) For each $x \in B_s \cap B_t$, write $D(x) = I(x) \cap E(S, \overline{S})$ and replace Ω by $\Omega_{x \rightarrowtail D(x)}$ from Fact 3.4. See the picture above. This adds a set N of at most 6 vertices to Ω and satisfies Invariant 1 by Fact 3.4. Furthermore, $|E(S, \overline{S})| \leq 3$ holds.

   b) Put N into B_t to obtain a tree decomposition of the resulting graph Ω. Recall that we write Ω_t for the torso of node t. We verify Invariant 2 by the following observations:

      - If $\mathrm{tw}(\Omega_t - \mathcal{R}) \leq c$ before Step 2.a, then $\mathrm{tw}(\Omega_t - \mathcal{R}) \leq c + 6$ now.
      - If $\Omega_t - \mathcal{R}$ was a plane graph obeying the triangle condition before Step 2.a, then the drawing can be extended as below. Note that, if $|B_s \cap B_t| = 3$, then the triangle condition yields that all edges incident with $B_s \cap B_t$ are contained in a *single* region bounded by $B_s \cap B_t$.



3. After Step 2, the set $S = B_t \setminus B_s$ induces a gate Γ with at most 3 dangling edges $E(S, \overline{S})$ in Ω, all of which lead to $B_s \cap B_t$. We want to contract Γ to a remnant vertex, but this requires us to compute $\mathrm{Sig}(\Gamma)$. Here, we will use that $\Omega_t - \mathcal{R}$ is either planar or has bounded treewidth by Invariant 2, but we still need to handle the remnant vertices present in Γ. To this end, we realize remnant vertices by planar matchgates via Lemma 2.23 after a preprocessing step.

   a) If Ω_t contains a clique with two attaching remnant vertices $u, v \in \mathcal{R}$, replace Ω by the graph $\Omega_{\downarrow\downarrow u,v}$ from Fact 3.5. This requires time $\mathcal{O}(1)$ and satisfies Invariant 1. Repeat this until, for every clique K in Ω_t, at most one remnant vertex in Ω_t attaches to K. Now observe the following:

i. We have $\mathrm{tw}(\Omega_t - \mathcal{R}) = \max\{\mathrm{tw}(\Omega_t), 4\}$: By Invariant 2, every remnant vertex $r \in \mathcal{R}$ attaches to some clique K in $\Omega_t - \mathcal{R}$. By Lemma 1.47, the tree decomposition of $\Omega_t - \mathcal{R}$ has a node b whose bag contains K. Add a child b' to b, with bag $K \cup \{r\}$, to obtain a tree decomposition of Ω_t.

ii. If $\Omega_t - \mathcal{R}$ is planar, then so is Ω_t: By Invariant 2, every remnant vertex $r \in \mathcal{R}$ has neighborhood $K_r = B_t \cap B_b$ for some neighbor b of t. By Step 3.a, no other remnant vertex has neighborhood K_r. By the triangle condition, K_r bounds a face of $\Omega_t - \mathcal{R}$, so we can draw the single remnant vertex r with neighborhood K_r in this face.

b) Replace each remnant vertex $v \in V(\Gamma) \cap \mathcal{R}$ by a planar matchgate from Lemma 2.23 that realizes the signature of v. This is possible by Invariant 4: Since v has the signature of a matchgate, this signature obeys the parity condition, and can hence be realized by Lemma 2.23. Note that Γ now is a matchgate itself, and similar to Steps 3.a.i and 3.a.ii, it is either planar or has bounded treewidth. Hence, we can compute $\mathrm{Sig}(\Gamma)$ with at most 2^3 calls to PerfMatch in planar graphs or bounded-treewidth graphs via Fact 2.18 and the algorithms from Theorem 1.27 for planar graphs and Corollary 1.45 for bounded-treewidth graphs.

c) Contract Γ to a remnant vertex w_t with signature $\mathrm{Sig}(\Gamma)$. This preserves $\mathrm{Holant}(\Omega)$ by Lemma 2.11. Place w_t into $\mathcal{R}$ and B_s, then delete the node t from T. We check the four invariants:

Inv. 1	All modifications were explicitly noted to preserve $\mathrm{Holant}(\Omega)$.
Inv. 2,4	The vertex w_t is obtained from contracting Γ, which eventually was a matchgate. We added w_t to B_s and $\mathcal{R}$, and we only deleted other vertices, but never modified their signatures.
Inv. 3	All dangling edges of Γ lead to $B_s \cap B_t$.

Repeat these steps until the root of $\mathcal{T}$ is reached, when Ω is contracted to an isolated remnant vertex. By Invariant 1, its signature of arity 0 is equal to $\mathrm{PerfMatch}(G)$.

The running time for each step is bounded by $\mathcal{O}(n^{1.5})$ arithmetic operations in the planar case (Theorem 1.27), or by $f(k) \cdot n$ in the bounded-treewidth case (Corollary 1.45). The number of executed steps is equal to the number of nodes in $\mathcal{T}$, which can easily be ensured to be $\mathcal{O}(n)$. Hence, the overall running time spent, including that for Theorem 3.3, is $g(H) \cdot n^4$ for some computable function g. This proves Theorem 3.2.

Remark 3.6. The signatures of contracted gates (and consequently, the edge-weights appearing in the matchgates realizing them) may feature numbers with at most $\mathcal{O}(n)$ bits. Performing arithmetic operations on such numbers however does not change the overall running time.

4. Apices and planar k-defect matchings

In this chapter, we consider PerfMatch on graphs with small apex number. In Section 4.1, we prove Theorem 4.1, which asserts that PerfMatch/apex is #W[1]-hard, from which we derive #W[1]-hardness for PerfMatch/hadw as Corollary 4.2. In Section 4.2, we extend this to show that the problem #PlanarDualMatch of counting k-defect matchings is #W[1]-hard, as claimed in Theorem 4.10. These results are complemented by an fpt-algorithm for graphs containing certain restricted apices in Theorem 4.21, see Section 4.3.

4.1. Perfect matchings on k-apex graphs

In this section, we prove Theorem 4.1 and thus present the first major application of our framework of combined signatures.

Theorem 4.1 (restated from page 83)**.** *The problem* PerfMatch^{0,1}/apex *is* #W[1]*-hard.*

By Remark 1.53, and as discussed before, this yields the following corollary.

Corollary 4.2 (restated from page 83)**.** *The problem* PerfMatch^{0,1}/hadw *is* #W[1]*-hard.*

Our reduction is from the problem #GridTiling of counting planar grid tilings, which was introduced as Problem 1.12 on page 32. In this problem, we are given a function $\mathcal{T} : [k]^2 \to 2^{[n]^2}$, and we wish to count the consistent grid tilings $a : [k]^2 \to [n]^2$ of $\mathcal{T}$, as specified in Problem 1.12. By Lemma 1.13, this problem is #W[1]-hard, even when $\mathcal{T}$ is balanced, in the sense that we are given some $T \in \mathbb{N}$ such that, for any choice of $\kappa \in [k]^2$ and $v \in [n]$, the set $\mathcal{T}(\kappa)$ features exactly T pairs of the form $(\star, v)$.

From a very high level, our reduction from #GridTiling could be compared to, say, the reduction in [Mar12] for planar multiway cut: In Section 4.1.1, we first express the number of consistent grid tilings as Holant(G) for a $k \times k$ grid G with appropriate signatures. This is possible since the consistency of grid tilings can be checked locally at each cell. Then we realize the signatures of G in Section 4.1.2. At this point however, we need to invoke the framework of combined signatures, described in Section 2.3, and this is where we depart from the usual reductions from #GridTiling. In particular, we need to invoke 2^{k^2} oracle calls to the target problem PerfMatch/apex, and thus need to exploit the full power of Turing fpt-reductions, whereas one oracle call sufficed in [Mar12].

4.1.1. Global construction

Let $n, k, T \in \mathbb{N}$ and let $\mathcal{T} : [k]^2 \to 2^{[n]^2}$ be an instance to #GridTiling as above. First, we show how to reformulate $\mathcal{T}$ as the Holant of a signature graph $G = G(\mathcal{T})$. This graph G

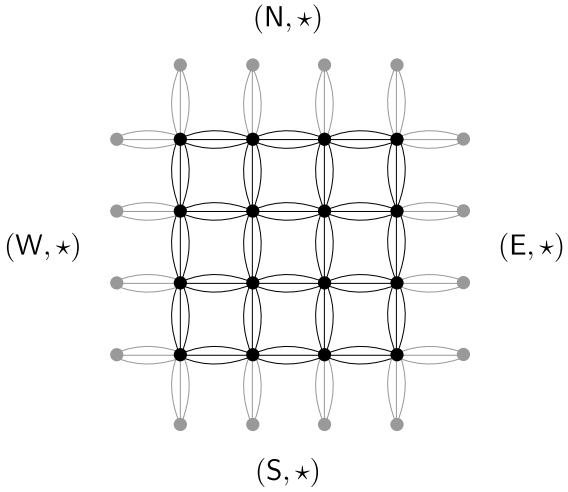


Figure 4.1.: The signature graph $G(\mathcal{T})$. Border vertices c_κ for $\kappa \in \{\mathsf{N}, \mathsf{W}, \mathsf{S}, \mathsf{E}\} \times [k]$ are shown gray, along with their incident edges. Cell vertices c_κ for $\kappa \in [k]^2$ are shown black. Horizontally or vertically adjacent edges are connected by an edge bundle of n parallel edges.

consists of a $k \times k$ square grid of *cells,* and $4k$ additional *border vertices* adjacent to the borders of the grid. Note that G is planar. We denote its vertices by c_κ for $\kappa \in \Xi$, where

$$\Xi := [k]^2 \ \cup \ \{\mathsf{N}, \mathsf{W}, \mathsf{S}, \mathsf{E}\} \times [k].$$

Vertices $\kappa, \kappa' \in [k]^2$ are called horizontally or vertically adjacent as specified in Section 1.1. Additionally, any index (N, i) for $i \in [k]$ is declared to be vertically adjacent to $(1, i)$, and (S, i) is vertically adjacent to (k, i). Likewise, (W, i) is horizontally adjacent to $(i, 1)$, and (E, i) is horizontally adjacent to (i, k), see Figure 4.1. We refer to the neighbors of any index/vertex using cardinal directions in the obvious way, e.g., we may speak of the northern neighbor of a vertex. Between any pair of vertices c_κ and $c_{\kappa'}$ with adjacent indices κ and κ', we place a set $E_{\kappa,\kappa'}$ of n parallel edges, which we call an *edge bundle.*

We proceed to define the signatures of G. In the assignments $a \in \{0,1\}^{E(G)}$ we are interested in, each edge bundle features exactly one active edge, which is used to encode a number from $[n]$. At border vertices, we place the signature $\mathtt{HW}_{=1}$ to ensure this. The signatures of cells c_κ, for $\kappa \in [k]^2$, will be defined so that each cell propagates the number $x_W \in [n]$ encoded by its western incident edge bundle to the east, and its northern number $x_N \in [n]$ to the south while checking along the way whether $(x_W, x_N) \in \mathcal{T}(\kappa)$ holds.

In this section, we adhere to the following notational conventions:

- We often identify the string $0^{i-1}10^{n-i} \in \{0,1\}^n$ with the number i, for all $i \in [n]$, when it is clear from the context which of these two objects we refer to.
- The $4n$ incident edges of each vertex c_κ, for $\kappa \in [k]^2$, are ordered such that all northern edges appear first, in a block of length n, followed by the n eastern, the n southern, and finally the n western edges.

- We also implicitly consider all strings $x \in \{0,1\}^{4n}$ to be decomposed into $x = x_N x_E x_S x_W$ with $x_N, x_E, x_S, x_W \in \{0,1\}^n$,

Using these conventions, we define the predicates

$$\begin{aligned} \varphi_{one}(x) &\equiv \mathrm{hw}(x_N) = 1 \wedge \mathrm{hw}(x_W) = 1, \\ \varphi_{prop}(x) &\equiv x_N = x_S \wedge x_W = x_E. \end{aligned}$$

If a vertex function f satisfies $\varphi_{prop}(x)$ for each $x \in \mathrm{supp}(f)$, then we call f *propagating*. By placing at each cell vertex c_κ, with $\kappa \in [k]^2$, a specific propagating signature f_κ, which we define in the following, we can complete the grid G to a signature graph whose satisfying assignments correspond bijectively to the valid grid tilings of $\mathcal{T}$.

Definition 4.3. For $\kappa \in [k]^2$, let f_κ be any function $f_\kappa : \{0,1\}^{4n} \to \{0,1\}$ such that, for all $x \in \{0,1\}^{4n}$ satisfying the predicate $\varphi_{one}(x)$, we have

$$f_\kappa(x) = \begin{cases} 1 & \text{if } \varphi_{prop}(x) \wedge (x_W, x_N) \in \mathcal{T}(\kappa), \\ 0 & \text{otherwise.} \end{cases}$$

Note that no requirement is imposed upon $f_\kappa(x)$ on those $x \in \{0,1\}^{4n}$ that fail to satisfy the predicate $\varphi_{one}(x)$.

We attach a signature f_κ to the cell c_κ, for all $\kappa \in [k]^2$. Recall that border vertices are assigned $\mathtt{HW}_{=1}$. In the following, we show that $G = G(\mathcal{T})$ indeed encodes $\mathcal{T}$ properly.

Lemma 4.4. *The consistent grid tilings of $\mathcal{T}$ correspond bijectively to the satisfying assignments in $x \in \{0,1\}^{E(G)}$, and each satisfying assignment x has $\mathrm{val}_G(x) = 1$.*

Proof. Every consistent grid tiling $a : [k]^2 \to [n]^2$ can be transformed into an assignment $x(a) \in \{0,1\}^{E(G)}$ as follows: For each $\kappa \in [k]^2$, with $a(\kappa) = (i,j)$, declare the i-th edge in the western edge bundle of c_κ and the j-th edge in the northern edge bundle of c_κ to be active. At vertices $c_{(k,\star)}$, copy the assignment from northern edges to southern edges, and at $c_{(\star,k)}$, copy the assignment from western edges to eastern edges. Declare all other edges to be inactive. It follows from the definition of f_κ at all $\kappa \in [k]^2$ that $\mathrm{val}_G(x(a)) = 1$.

For the converse direction, we show that every satisfying assignment $x \in \{0,1\}^{E(G)}$ can be written as $x = x(a)$ for some consistent grid tiling a, where $x(a)$ is defined as in the previous paragraph. Note that this also implies $\mathrm{val}_G(x) = 1$. By the signature $\mathtt{HW}_{=1}$, every border vertex is incident with exactly one active edge in x. Hence, the restriction of x to $I(c_{1,1})$ satisfies φ_{one}; call this restricted assignment y. Since $f_{1,1}(y) = 1$, and since $f_{1,1}$ is propagating, we also have $\varphi_{prop}(y)$. Furthermore, we have $(y_W, y_N) \in \mathcal{T}(1,1)$ by definition of $f_{1,1}$. By induction along rows and columns, we obtain, for every $\kappa \in [k]^2$, that the partial assignment y at $I(c_\kappa)$ satisfies $\varphi_{prop}(y)$ and $(y_W, y_N) \in \mathcal{T}(\kappa)$. Hence $x = x(a)$ holds for a unique consistent grid tiling a. □

In the next subsection, we realize each signature f_κ in G as a linear combination of two matchgate signatures, where one matchgate is planar, and the other has maximum apex

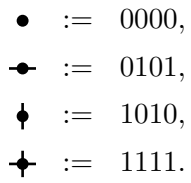
number 2. Note that the remaining signatures $\mathtt{HW}_{=_1}$ occurring in G are planar. Since G itself is planar and features only k^2 signatures f_κ, the graphs realizing G will feature at most $2k^2$ apices, and we will use this to obtain the desired parameterized reduction.

4.1.2. Realizing cell signatures

Using arguments as in Section 2.2.2, it can be *unconditionally* shown that there are non-planar signatures f_κ. From the viewpoint of complexity theory, if all signatures f_κ in G were planar and we knew explicit planar matchgates, then we could count consistent grid tilings by reduction to planar PerfMatch, and thus show $\mathsf{FP} = \#\mathsf{P}$ by the FKT method.[1]

Rather than trying to use planar matchgates, we show that each signature f_κ can be realized as a specific *combination* of the signatures of one planar and one 2-apex matchgate. Note that at least one non-planar constituent is necessary in such a combination, as we could otherwise solve #GridTiling by reduction to 2^{k^2} instances of planar PerfMatch via Lemma 2.35, solve each planar instance by the FKT method, and thus show $\mathsf{FPT} = \#\mathsf{W}[1]$.

In the remainder of this section, we construct the constituents for f_κ. We consider $\kappa \in [k]^2$ to be fixed, we write $A = \mathcal{T}(\kappa)$ and we recall that $A \subseteq [n]^2$. For clarity of presentation, we abbreviate certain bitstrings of length 4:



The constituents for f_κ will be the signatures of two gates Φ and $\Phi'(A)$ that we construct in the following. These gates in turn use as building blocks the signatures PASS and ACT, which we will show to be realized by "essentially planar" matchgates.

$$\mathtt{PASS}(x) = \begin{cases} -1 & \text{if } x = \text{⊕}, \\ 1 & \text{if } x \in \{\, \bullet, \text{—•—}, \text{•—•} \,\}, \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathtt{ACT}(x) = \begin{cases} \mathtt{PASS}(y) & \text{if } x = y00 \text{ for } y \in \{0,1\}^4, \\ 1 & \text{if } x \in \{\, \text{•—•}11, \text{⊕}11 \}, \\ 0 & \text{otherwise.} \end{cases}$$

Note that the signature PASS agrees with the signature CROSS from Section 2.2.2 on all inputs, with the exception of ⊕, where

$$\mathtt{PASS}(\text{⊕}) = -1 = -\mathtt{CROSS}(\text{⊕}).$$

[1] Note that Lemma 1.11 and 1.13 imply #P-hardness of #GridTiling since the initial reduction source #Clique is easily seen to be #P-complete and the reductions are polynomial-time reductions.

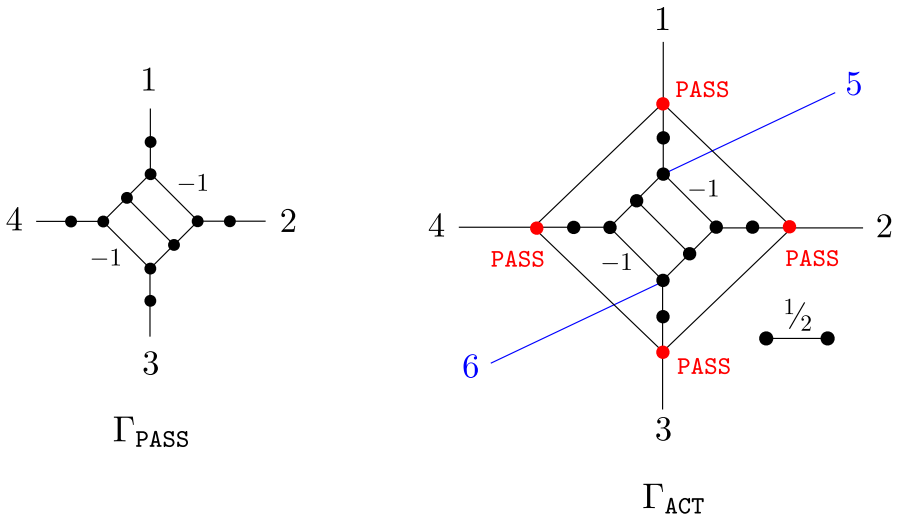


Figure 4.2.: The matchgates $\Gamma_{\texttt{PASS}}$ and $\Gamma_{\texttt{ACT}}$ realizing PASS and ACT. All black vertices are assigned $\texttt{HW}_{=1}$. In the drawing of $\Gamma_{\texttt{ACT}}$, all red vertices are assigned PASS. The matchgate $\Gamma_{\texttt{PASS}}$ was inspired by [CG14, Figure 6].

This allows PASS to satisfy the matchgate identities on on page 65, which are violated by CROSS, and hence PASS admits a planar matchgate $\Gamma_{\texttt{PASS}}$, shown in Figure 4.2 and verified in Lemma 4.5

The signature ACT has arity 6, and we consider its last two inputs as "switches", which will later connect to apices. It is crucial to observe that

$$\texttt{ACT}(x00) = \texttt{PASS}(x) \quad \forall x \in \{0,1\}^4.$$

Intuitively, if the two switch inputs are off, then ACT behave exactly like PASS on its non-switch inputs. If both switches are on, then some differences occur, namely, the restriction to non-switch inputs must be in state $\bullet$ or $\bullet\!\!-\!\!\bullet$ for ACT to yield a nonzero value. Furthermore, if only one of the two switches is on, then ACT yields value zero.

Lemma 4.5. *It holds that* $\mathrm{Sig}(\Gamma_{\texttt{PASS}}) = \texttt{PASS}$ *and* $\mathrm{Sig}(\Gamma_{\texttt{ACT}}) = \texttt{ACT}$.

Proof. Deferred to Section 4.1.3. □

Recall that we wanted to define matchgates Φ and $\Phi'(A)$ using the signatures PASS and ACT. To obtain Φ, we arrange PASS in a square grid as in Figure 4.3 on the following page and realize each occurrence by $\Gamma_{\texttt{PASS}}$. A similar construction yields $\Phi'(A)$, with the addition of apex vertices a_1 and a_2, and the signature ACT replacing PASS at all indices $\kappa \in A$, which is realized by $\Gamma_{\texttt{ACT}}$. Note that $\Phi'(A)$ is not necessarily planar, but it is obviously planar after removal of a_1 and a_2. We give more formal definitions in the following.

Definition 4.6. Let Φ denote the gate consisting of an $n \times n$ square grid of vertices featuring PASS, with $4n$ dangling edges, as shown in Figure 4.3.

Given $A \subseteq [n]^2$, let the gate $\Phi'(A)$ be defined as an $n \times n$ square grid of vertices v_τ for $\tau \in [n]^2$. The signature at v_τ is defined as ACT if $\tau \in A$, and as PASS if $\tau \notin A$. Add two

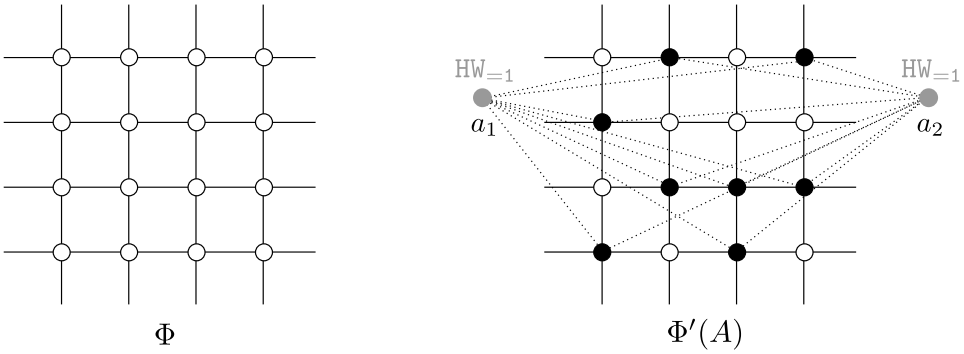


Figure 4.3.: The gates Φ and $\Phi'(A)$. White vertices are assigned PASS, black vertices are assigned ACT. Gray vertices in $\Phi'(A)$ are apex vertices. Edges from apices are drawn dashed to avoid visual cluttering. By the balance property of $\mathcal{T}$, every column has the same number T of signatures ACT.

apex vertices a_1 and a_2 with signature $\mathtt{HW}_{=1}$. For all $\tau \in A$, add the edges $a_1 v_\tau$ and $a_2 v_\tau$ and declare these to be the last two edges in the edge ordering of $I(v_\tau)$.

By realizing PASS and ACT by the matchgates $\Gamma_{\mathtt{PASS}}$ and $\Gamma_{\mathtt{ACT}}$, respectively, we can view Φ and $\Phi'(A)$ as matchgates as well.

By the balance property of our instance to #GridTiling, as ensured in Lemma 1.13, there is some $T \in \mathbb{N}$ such that we have $|A \cap (\star, v)| = T$ for all $v \in [n]$. That is, in Figure 4.3, we may assume that every column features the same number of vertices with signature ACT. In the following, we use this to compute the signatures of Φ and $\Phi'(A)$, and we show how to combine them to realize the cell signature f_κ.

Lemma 4.7. *Recall the definition of the predicates φ_{one} and φ_{prop} on page 95. Let $x \in \{0,1\}^{4n}$ be an assignment that satisfies the predicate φ_{one}. Then*

$$\mathrm{Sig}(\Phi, x) = \begin{cases} 0 & \text{if } \neg\varphi_{prop}(x), \\ -1 & \text{if } \varphi_{prop}(x). \end{cases} \tag{4.1}$$

$$\mathrm{Sig}(\Phi'(A), x) = \begin{cases} 0 & \text{if } \neg\varphi_{prop}(x) \\ \begin{cases} -T & \text{if } (x_W, x_N) \notin A \\ -T+2 & \text{if } (x_W, x_N) \in A \end{cases} & \text{if } \varphi_{prop}(x). \end{cases} \tag{4.2}$$

For $x \in \{0,1\}^{4n}$ satisfying φ_{one}, and for $\kappa \in [k]^2$, let us write $A = \mathcal{T}(\kappa)$. Then we have

$$f_\kappa(x) = \frac{1}{2}\mathrm{Sig}(\Phi'(A), x) - \frac{T}{2}\mathrm{Sig}(\Phi, x). \tag{4.3}$$

In Section 4.1.3, we prove Lemma 4.7 by inspecting the possible satisfying assignments to Φ and Φ'. Before doing this, let us first show how Lemma 4.7 implies Theorem 4.1:

Proof of Theorem 4.1. By Lemma 4.4, we know that Holant(G) counts the consistent grid tilings of $\mathcal{T}$. Using the linear combination (4.3) and Lemma 4.5, the Combined Signature

Lemma (Lemma 2.35) yields

$$\mathrm{Holant}(G) = \frac{1}{2^{k^2}} \sum_{\omega:[k]^2\to[2]} (-T)^{d(\omega)} \cdot \mathrm{PerfMatch}(H_\omega), \tag{4.4}$$

where, for $\omega : [k]^2 \to [2]$, we define $d(\omega) \in \mathbb{N}$ and the graph H_ω in the following way:

- $d(\omega)$ is the number of 2-entries in ω, and
- H_ω is obtained as follows:
  - For $\kappa \in [k]^2$ with $\omega(\kappa) = 1$, insert the matchgate $\Phi'(\mathcal{T}(\kappa))$ at the cell vertex c_κ.
  - For $\kappa \in [k]^2$ with $\omega(\kappa) = 2$, insert the matchgate Φ at the cell vertex c_κ.

Since G is planar, and since Φ and $\Phi'(\mathcal{T}(\kappa))$ for $\kappa \in [k]^2$ both have at most 2 apices, it follows that the graphs H_ω have maximum apex number $2k^2$.

Note that, by construction of the matchgates $\Gamma_{\texttt{PASS}}$ and $\Gamma_{\texttt{ACT}}$, every graph H_ω features only edge-weights from the set $\{-1, 1/2, 1\}$. Furthermore, non-unit edge-weights in H_ω appear only at edges $uv \in E(H_\omega)$ where neither of u, v are apices. We can hence use Lemma 1.37 and Remark 1.30 to remove the edge-weights -1 and $1/2$, while maintaining the apex number, and thus obtain #W[1]-completeness of $\mathrm{PerfMatch}^{0,1}$/apex. □

Remark 4.8. For later use, we remark the following: By construction, the apices in the reduction image H_ω form an independent set, for any $\omega : [k]^2 \to [2]$, and each non-apex vertex in H_ω is incident with at most one apex. This last condition holds because the matchgate $\Gamma_{\texttt{ACT}}$ has no vertex with two incident dangling edges.

Before proving Lemmas 4.5 and 4.7, we show that our proof of Theorem 4.1 fails to yield #W[1]-hardness for PerfMatch/hadw$_\mathcal{D}$ for a graph class $\mathcal{D}$ with $\mathcal{K} \not\preceq \mathcal{D}$. Recall that hadw$_\mathcal{D}$ was introduced in the definition on page 84. Let $\mathcal{D}$ denote the class of reduction images constructed by Theorem 4.1, i.e., the graphs $H = H_\omega$ for $\omega : [k]^2 \to 2$ over all possible instances $\mathcal{T}$ to the problem #GridTiling. Then we observe that $\mathcal{D}$ contains all complete graphs as minors.

Fact 4.9. *For every $t \in \mathbb{N}$, there is a graph $H \in \mathcal{D}$ such that $K_t \preceq H$.*

Proof sketch. Consider a rectilinear drawing of the complete graph K_t on the integer lattice such that every crossing of edges involves only two edges. To realize such a drawing, we might need to draw vertices as squares rather than points, and it can be verified that the complete drawing of K_t can be fitted into a square of side-length $\ell = t^{\mathcal{O}(1)}$.

Let $\mathcal{T}$ denote an instance to #GridTiling with $\ell \times \ell$ cells, universe size n, and $\mathcal{T}(\kappa) = [n]^2$ for all $\kappa \in [\ell]^2$. Consider the signature graph $G = G(\mathcal{T})$ defined in Section 4.1.1 and the reduction image $H = H_{\omega^*}$ with $\omega^* = (1, \ldots, 1)$ obtained from G, where every signature f_κ is replaced by the 2-apex matchgate $\Phi'([n]^2)$.

Then we have $K_t \preceq H$: Starting from the drawing of K_t on the integer lattice, we can embed each line segment of the drawing between adjacent lattice points in the planar part of $\Phi'([n]^2)$. Since each cell has two apices, we can embed the crossing lines using paths over the apex vertices. □

4.1.3. Deferred proofs

It remains to prove Lemmas 4.5 and 4.7, which is the goal of the following subsection.

Lemma 4.5: Realizing ACT and PASS

We compute mechanically in Appendix C.2 that

$$\mathrm{Sig}(\Gamma_{\texttt{PASS}}) = \texttt{PASS}.$$

Concerning $\Gamma_{\texttt{ACT}}$, we decompose $\Gamma_{\texttt{ACT}}$ into a matchgate Ψ, and an additional part, as shown in Figure 4.4 on the next page. In Appendix C.2, we compute mechanically that

$$\mathrm{Sig}(\Psi, xy) = \begin{cases} \texttt{ACT}(xy) & \text{if } \mathrm{hw}(x) \text{ even,} \\ \text{arbitrary} & \text{otherwise.} \end{cases} \tag{4.5}$$

Intuitively, the ring of PASS signatures around Ψ in $\Gamma_{\texttt{ACT}}$ acts as an "even filter" that ensures the following, for all $x \in \{0,1\}^4$ and $y \in \{0,1\}^2$:

- If $\mathrm{hw}(x)$ is not even, then $\mathrm{Sig}(\Gamma_{\texttt{ACT}}, xy) = 0$, regardless of the value of Ψ on xy.
- If $\mathrm{hw}(x)$ is even, then $\mathrm{Sig}(\Gamma_{\texttt{ACT}}, xy) = \mathrm{Sig}(\Psi, xy)$. Observe that, by (4.5), this also implies $\mathrm{Sig}(\Gamma_{\texttt{ACT}}, xy) = \texttt{ACT}(xy)$.

Since $\texttt{ACT}(xy) \neq 0$ implies $x \in \{$  $\}$, which in turn implies that $\mathrm{hw}(x)$ is even, we obtain that indeed

$$\mathrm{Sig}(\Gamma_{\texttt{ACT}}) = \texttt{ACT}.$$

To determine $\mathrm{Sig}(\Gamma_{\texttt{ACT}}, xy)$ for $x \in \{0,1\}^4$ and $y \in \{0,1\}^2$, we consider the satisfying assignments $w \in \{0,1\}^{E(\Gamma_{\texttt{ACT}})}$ that extend xy. The dummy edge of weight $1/2$ is present in any assignment w and contributes a factor $1/2$ to $\mathrm{val}(w)$.[2] At each red vertex, the signature PASS ensures that opposing edges have the same assignment under w. This fixes the value of all black edges and ensures that $\mathrm{val}(w)$ contains the factor $\mathrm{Sig}(\Psi, xy)$, contributed from the green vertex with signature $\mathrm{Sig}(\Psi)$.

It remains to assign values to the red edges: Due to the signature PASS at red vertices, this is possible with at most two satisfying assignments $w_1, w_2 \in \{0,1\}^{E(\Gamma_{\texttt{ACT}})}$:

w_1 : All red edges are active. Then every red vertex in state  yields a factor $\texttt{PASS}($$) = -1$, while all other red vertices are in one of the states  or  and yield value 1. The number of red vertices in state is $\mathrm{hw}(x)$, so the value of $\Gamma_{\texttt{ACT}}$ on w_1 is

$$\mathrm{val}(w_1) = \frac{1}{2} \cdot (-1)^{\mathrm{hw}(x)} \cdot \mathrm{Sig}(\Psi, xy).$$

[2] In the following, let us write $\mathrm{val}(w)$ instead of $\mathrm{val}_{\Gamma_{\texttt{ACT}}}$ to avoid the double index.

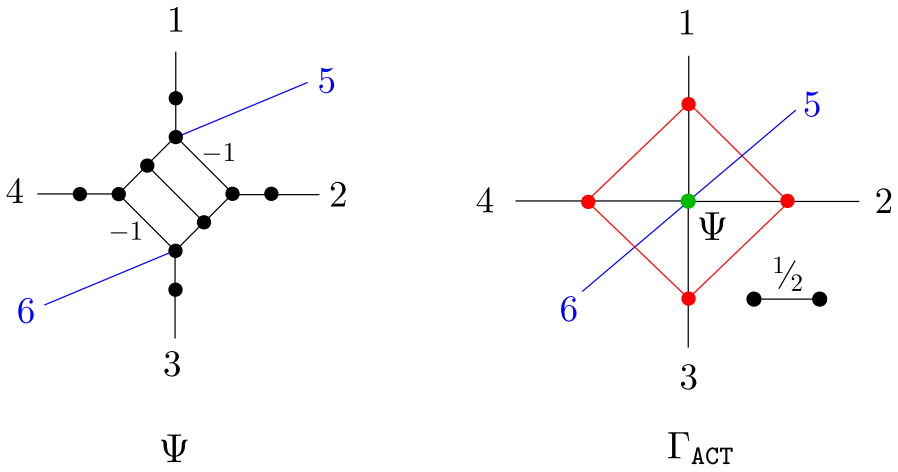


Figure 4.4.: The matchgate Ψ and the construction of $\Gamma_{\texttt{ACT}}$ from Ψ. All black vertices are assigned $\texttt{HW}_{=1}$, all red vertices are assigned $\texttt{PASS}$, and the green vertex in the middle of the drawing is assigned $\mathrm{Sig}(\Psi)$.

w_2 : No red edges are active. Then every red vertex is in one of the states  or  and hence yields value 1. Thus, the value of $\Gamma_{\texttt{ACT}}$ on w_2 is

$$\mathrm{val}(w_2) = \frac{1}{2} \cdot \mathrm{Sig}(\Psi, xy).$$

It follows that for all $x \in \{0,1\}^4$ and $y \in \{0,1\}$, we have

$$\begin{aligned}
\mathrm{Sig}(\Gamma_{\texttt{ACT}}, xy) &= \mathrm{val}(w_1) + \mathrm{val}(w_2) \\
&= \frac{1}{2} \cdot \Big((-1)^{\mathrm{hw}(x)} \cdot \mathrm{Sig}(\Psi, xy) + \mathrm{Sig}(\Psi, xy) \Big) \\
&= \begin{cases} \mathrm{Sig}(\Psi, xy) & \text{if } \mathrm{hw}(x) \text{ even,} \\ 0 & \text{otherwise.} \end{cases} \\
&= \texttt{ACT}(xy)
\end{aligned}$$

This proves Lemma 4.5.

Lemma 4.7: The signature of Φ

Let $x \in \{0,1\}^{4n}$ be an assignment to the dangling edges of Φ that satisfies the predicate $\varphi_{one}(x)$, and let $xy \in \{0,1\}^{E(\Phi)}$ be a satisfying assignment to the gate Φ that extends x. We show that, whenever x satisfies the predicate φ_{prop}, then y is unique and has value -1, so $\mathrm{Sig}(\Phi, x) = \mathrm{val}_\Phi(xy) = -1$. Furthermore, we show that, if x does not satisfy the predicate φ_{prop}, then no such y exists, and hence $\mathrm{Sig}(\Phi, x) = 0$.

Recall from on page 95 that we decompose x into x_N, x_E, x_S, x_W. Write $W \in [n]$ and $N \in [n]$ for the unique non-zero index in $x_W \in \{0,1\}^n$ and $x_N \in \{0,1\}^n$, respectively.

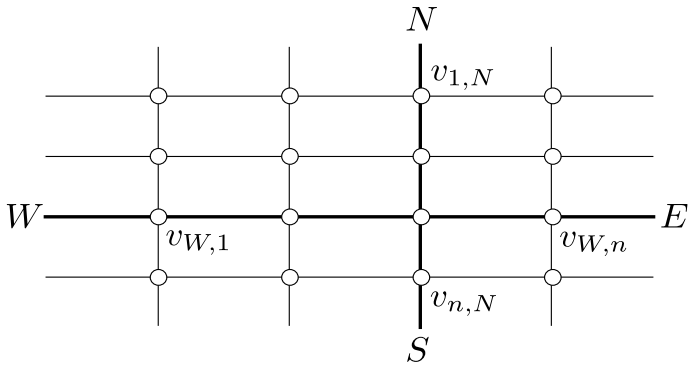


Figure 4.5.: The unique assignment y to $E(\Phi)$ that extends x.

These numbers are well-defined because x satisfies $\varphi_{one}(x)$ by assumption. Then all western and eastern edges of vertices in row $(W, \star)$ are active in xy: The western edge of $v_{W,1}$ is active by definition, and since xy satisfies Φ and PASS at $v_{W,1}$, the vertex $v_{W,1}$ must be in state ⊸ or ⊹, so its eastern edge is also active. The same follows inductively for all vertices in the row $(W, \star)$. By the same argument, rotated about 90 degrees, all northern and southern edges of vertices in row $(\star, N)$ are active in xy, see Figure 4.5.

By a similar argument, no other edges are active, and we conclude that y is uniquely determined by x. Furthermore, if E and S denote the active indices in x_E and x_S, then we observe that $W = E$ and $N = S$, since otherwise xy could not satisfy $v_{W,n}$ and $v_{n,N}$. Hence, xy satisfies Φ only if $\varphi_{prop}(x)$ holds. We obtain

$$\mathrm{Sig}(\Phi, x) = 0 \qquad \text{if } \neg\varphi_{prop}(x).$$

If $\varphi_{prop}(x)$ holds, then $v_{W,N}$ is in state ⊹ under xy, while the $n-1$ other vertices in row $(W, \star)$ are in state ⊸, the $n-1$ other vertices in column $(\star, N)$ are in state ╿, and the remaining $n^2 - 2n + 1$ vertices are in state •. In conclusion, $\varphi_{prop}(x)$ implies

$$\begin{aligned} \mathrm{Sig}(\Phi, x) &= \mathrm{val}(\Phi, xy) \\ &= \texttt{PASS}(\text{⊹}) \cdot \texttt{PASS}(\text{╿})^{n-1} \cdot \texttt{PASS}(\text{⊸})^{n-1} \cdot \texttt{PASS}(\bullet)^{n^2-2n+1} \\ &= -1. \end{aligned}$$

This proves (4.1).

Lemma 4.7: The signature of $\Phi'(A)$

Let $\Phi' = \Phi'(A)$ for some fixed $A \subseteq [n]^2$, let $D \subseteq E(\Phi')$ denote the dangling edges of Φ' and let $F = I(a_1) \cup I(a_2)$ denote the set of edges incident with either a_1 or a_2 in Φ'. Let $x \in \{0,1\}^{4n}$ be an assignment to D that satisfies the predicate $\varphi_{one}(x)$, and let

$xyz \in \{0,1\}^{E(\Phi')}$ be a satisfying assignment to the edges of Φ' that extends x, with

$$\begin{aligned} y &\in \{0,1\}^{E(\Phi')\setminus(F\cup D)}, \\ z &\in \{0,1\}^{F}. \end{aligned}$$

We consider the restriction of xyz to xy, that is, to edges not incident with any apex. By definition of `PASS` and `ACT`, we have, for every vertex $v \in V(\Phi')$, that

$$(xy)|_{I(v)} \in \{ \bullet, ┃, ━, ╋ \}. \tag{4.6}$$

Recall from the convention on page 95 that we decompose x into x_N, x_E, x_S, x_W. Write $W \in [n]$ and $N \in [n]$ for the unique non-zero index in $x_W \in \{0,1\}^n$ and $x_N \in \{0,1\}^n$. Since $(xy)|_{I(v)} \in \text{supp}(\texttt{PASS})$ holds by (4.6) and the definition of `PASS`, the same argument as in the previous subsection for Φ shows that the western and eastern edges of all vertices in row $(W, \star)$ are active under xy, as well as the northern and southern edges of all vertices in the column $(\star, N)$. Likewise, as seen in the previous subsection, it shows that no other edges in $E(\Phi') \setminus F$ are active, that y is unique if $\varphi_{prop}(x)$ holds, and that y does not exist otherwise. This last statement implies that

$$\text{Sig}(\Phi', x) = 0 \qquad \text{if } \neg\varphi_{prop}(x).$$

In the following, let $x \in \{0,1\}^D$ be an assignment to the dangling edges of Φ' that satisfies $\varphi_{prop}(x)$, and let $xy \in \{0,1\}^{E(\Phi')\setminus F}$ be its unique extension, as seen for Φ. We consider the possible assignments $z \in \{0,1\}^F$ to the apex edges such that xyz satisfies Φ'. Here, while the choice of y was unique, the choice of z is not unique.

By virtue of $\texttt{HW}_{=1}$ at the apex vertices a_1 and a_2, there are unique indices $\kappa, \kappa' \in A$ such that the edges $a_1 v_\kappa$ and $a_2 v_{\kappa'}$ are active in xy. By definition of `ACT`, we actually have $\kappa = \kappa'$, since all elements in supp(`ACT`) end on 00 or 11. We write $\kappa^* = \kappa = \kappa'$ for the unique "apex-matched" index and $c^* = c_{\kappa^*}$ for the unique "apex-matched" vertex. By definition of `ACT`, we have

$$(xyz)|_{I(c^*)} \in \{ ┃11, ╋11 \}.$$

It follows that the second component of κ^* must be equal to N, since only vertices in $(\star, N)$ have state ┃ or ╋ under xy. There are T vertices with signature `ACT` in row $(\star, N)$, by the balance property of our instance $\mathcal{T}$ to #GridTiling, and we can choose any of these vertices to be apex-matched. We distinguish two main cases.

$(W, N) \notin A$: The apex-matched vertex must be in state ┃11 under xyz. It cannot be in state ╋11, since only $v_{W,N}$ can have state ╋ among its first four edges, but $v_{W,N}$ has `PASS` assigned, since $(W, N) \notin A$. This gives T assignments z such that xyz satisfies Φ'. Each assignment xyz satisfies $\text{val}_{\Phi'}(xyz) = -1$, because there is (i) one vertex in state ╋00, which contributes a factor of -1 to $\text{val}_{\Phi'}(xyz)$, and (ii) some number of vertices in other states, which however all contribute a unit factor to $\text{val}_{\Phi'}(xyz)$. This implies that $\text{Sig}(\Phi', x) = -T$ if both $(W, N) \notin A$ and $\varphi_{prop}(x)$ hold.

$(W, N) \in A$: The apex-matched vertex may be in state $\bullet\!\!-11$ or $-\!\!\bullet\!\!-11$. We distinguish these two subcases:

$\bullet\!\!-11$**:** We proceed as in the case of $(W, N) \notin A$, but we have only $T - 1$ choices left for the apex-matched vertex, since $v_{W,N}$ must have state $-\!\!\bullet\!\!-$ among its first four edges and can thus not be in state $\bullet\!\!-11$. This gives $T - 1$ assignments z with $\mathrm{Sig}(\Phi', xyz) = -1$ for each z.

$-\!\!\bullet\!\!-11$: Since only $v_{W,N}$ can have state $-\!\!\bullet\!\!-$ among its first four edges, the apex-matched vertex must be $v_{W,N}$. This gives one assignment z, and $\mathrm{Sig}(\Phi', xyz) = 1$, because all vertices yield a unit factor.

In total, we obtain

$$\mathrm{Sig}(\Phi', xyz) = (T - 1) \cdot (-1) + 1 = -T + 2$$

if both $(W, N) \in A$ and $\varphi_{prop}(x)$ hold.

This proves (4.2), and thus Lemma 4.7, and consequently Theorem 4.1.

4.2. Planar k-defect matchings

In this section, we prove Theorem 4.10: We show that, given a *planar* graph G and $k \in \mathbb{N}$, it is #W[1]-hard to count the k-defect matchings of G. This amounts to computing the coefficient of x^k in the matching-defect polynomial $\mu(G)$.

Theorem 4.10 (restated from page 83)**.** *Given a planar graph G and a number $k \in \mathbb{N}$ as inputs, the problem* #PlanarDualMatch *of counting the k-defect matchings in G is* #W[1]*-hard.*

In other words, we show that PerfMatch/apex does not become easier when we ignore the apex adjacency structure and instead make each apex adjacent to *all* vertices of the base graph: Given a graph G containing an independent set A of k apices that connect to all vertices of $G - A$, each k-defect matching of $G - A$ corresponds to precisely $k!$ perfect matchings of G. Our #W[1]-hardness result for #PlanarDualMatch thus implies #W[1]-hardness of PerfMatch/apex on graphs G whose apex vertices are adjacent to all vertices of the base graph.

In the proof, we introduce an intermediate problem #RestrDualMatch:

Problem 4.11 (#RestrDualMatch)**.** Given as input a triple (G, S, k) where G is a planar graph, $S \subseteq V(G)$ and $k \in \mathbb{N}$, count those k-defect matchings M of G whose defects all avoid S, i.e., those k-defect matchings M with $S \cap \mathrm{usat}(M) = \emptyset$.

As in the previous paragraph, the problem #RestrDualMatch may be considered equivalent to PerfMatch/apex on graphs G whose apices A are all adjacent to a specific subset

$S \subseteq V(H)$ of the base graph $H = G - A$, and to no other vertices. Our overall reduction then proceeds along the chain

$$\text{PerfMatch}^{0,1}/\text{apex} \leq_{fpt}^{lin} \#\mathsf{RestrDualMatch} \leq_{fpt}^{lin} \#\mathsf{PlanarDualMatch}. \tag{4.7}$$

More precisely, in the first reduction, we reduce from the problem $\text{PerfMatch}^{0,1}/\text{apex}$ when restricted to instances satisfying Remark 4.8.

4.2.1. Hardness of restricted k-defect matchings

In the following, let $\mathcal{DM}_k[G]$ denote the set of k-defect matchings in G. The first reduction in (4.7) follows from an application of the inclusion-exclusion principle.

Lemma 4.12. *The problem* #RestrDualMatch *is* #W[1]*-hard.*

Proof. We reduce from the problem $\text{PerfMatch}^{0,1}/\text{apex}$ and wish to count perfect matchings in an unweighted graph G with apex set $A = \{a_1, \ldots, a_k\}$ and planar base graph $H = G - A$. By Remark 4.8 on page 99, we can assume that

1. A is given as part of the input, along with G, and
2. A is an independent set, and
3. $V(H)$ admits a partition into $V_1 \cup \ldots \cup V_k \cup W$ such that all vertices $v \in V_i$ for $i \in [k]$ are adjacent to the apex a_i and to no other apices, while no vertex $v \in W$ is adjacent to any apex.

By (3), each vertex $v \in V(H)$ can be colored by its unique adjacent apex, or by a neutral color if $v \in W$. We then call a k-defect matching $M \in \mathcal{DM}_k[H]$ *colorful* if $|\text{usat}(M) \cap V_i| = 1$ holds for all $i \in [k]$, and we write $\mathcal{C}$ for the set of all such M. Note that $\text{usat}(H, M) \cap W = \emptyset$ for $M \in \mathcal{C}$, since none of its k defects are left over for W.

We claim that $\mathcal{PM}[G] \simeq \mathcal{C}$: If $M \in \mathcal{PM}[G]$, then $N = M - A$ satisfies $N \in \mathcal{C}$. Conversely, every $N \in \mathcal{C}$ can be extended to a unique $M \in \mathcal{PM}[G]$ by matching the unique i-colored defect to its unique adjacent apex a_i.

Given oracle access to #RestrDualMatch, we can determine $\#\mathcal{C}$ by the inclusion-exclusion principle from Lemma 1.33: For $i \in [k]$, let $\mathcal{A}_i$ denote the set of $M \in \mathcal{DM}_k[H]$ whose defects avoid color i, i.e., that satisfy $\text{usat}(H, M) \cap V_i = \emptyset$. Then

$$\mathcal{C} = \mathcal{DM}_k[H] \setminus \bigcup_{i \in [k]} \mathcal{A}_i.$$

For $S \subseteq [k]$, write $\mathcal{A}_S = \bigcap_{i \in S} \mathcal{A}_i$ and note that we can compute $\#\mathcal{A}_S$ by an oracle call to #RestrDualMatch on the instance $(H, \bigcup_{i \in S} V_i, k)$. We can hence compute

$$\#\mathcal{C} = \#\mathcal{PM}[G]$$

via inclusion-exclusion (Lemma 1.33) and 2^k oracle calls to #RestrDualMatch. □

4.2.2. From restricted to unrestricted matchings

For the second reduction in (4.7), we wish to solve instances (G, S, k) to #RestrDualMatch when given only an oracle for counting k-defect matchings in planar graphs, *without* the ability of specifying the set S. Let G, S and k be fixed in the following. Our reduction involves some algebraic manipulations on polynomials, such as a truncated version of polynomial division:

Lemma 4.13. *Let X be an indeterminate, and let $p, q \in \mathbb{Z}[X]$ be polynomials*

$$p = \sum_{i=0}^{m} b_i X^i, \qquad q = \sum_{i=0}^{n} a_i X^i,$$

with $a_0 \neq 0$. For all $t \in \mathbb{N}$, we can compute $b_0, \ldots, b_t$ with $\mathcal{O}(t^2)$ arithmetic operations when given as input $a_0, \ldots, a_t$ and the first $t+1$ coefficients of the product pq.

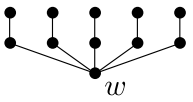
Proof. Let $c_0, \ldots, c_{n+m}$ enumerate the coefficients of the product pq. By elementary algebra, we have $c_i = \sum_{\kappa=0}^{i} a_\kappa b_{i-\kappa}$, which implies the linear system

$$\begin{pmatrix} a_0 & & \\ \vdots & \ddots & \\ a_t & \ldots & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ \vdots \\ b_t \end{pmatrix} = \begin{pmatrix} c_0 \\ \vdots \\ c_t \end{pmatrix}. \tag{4.8}$$

As this system is triangular with $a_0 \neq 0$ on its main diagonal, it has full rank and can be solved uniquely for $b_0, \ldots, b_t$ with $\mathcal{O}(t^2)$ arithmetic operations. □

Our proof also relies upon a gadget which will allow to distinguish S from $V(G) \setminus S$.

Definition 4.14. For $\ell \in \mathbb{N}$, an ℓ-rake R_ℓ is a matching M of size ℓ, together with an additional vertex w adjacent to one vertex of each edge in M:



Let $G_{S,\ell}$ be the graph obtained from attaching R_ℓ to each $v \in S$. This means adding a local copy of R_ℓ to v and identifying the copy of w with v. Please note that vertices $v \in V(G) \setminus S$ receive no attachments in $G_{S,\ell}$.

It is obvious that $G_{S,\ell}$ is planar if G is. Recall the defect-generating matching polynomial μ from Section 1.3.1. We first show that, for fixed $\ell \in \mathbb{N}$, the polynomial $\mu(G_{S,\ell})$ can be written as a weighted sum over matchings $M \in \mathcal{M}[G]$, where each M is weighted by an expression that depends on the number $|\mathrm{usat}(M) \cap S|$. Ultimately, we want to tweak these weights in such a way that only matchings with $|\mathrm{usat}(M) \cap S| = 0$ are counted.

Lemma 4.15. *Define polynomials $r, f_\ell \in \mathbb{Z}[X]$ and $s \in \mathbb{Z}[X, \ell]$ by*

$$\begin{aligned} r(X) &= 1 + X^2, \\ s(X, \ell) &= \ell + 1 + X^2, \\ f_\ell(X) &= (1 + X^2)^{|S|(\ell-1)}. \end{aligned}$$

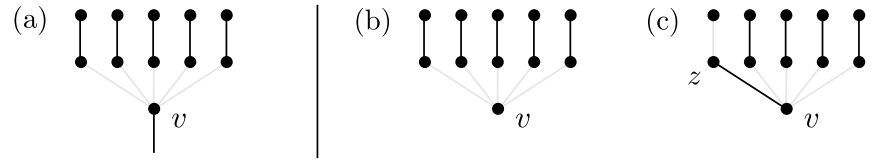


Figure 4.6.: Possible types of extensions of the rake at v. The left case corresponds to $v \notin \mathrm{usat}(M)$, and the two right cases correspond to $v \in \mathrm{usat}(M)$.

Then it holds that

$$\mu(G_{S,\ell}, X) = f_\ell \cdot \sum_{M \in \mathcal{M}[G]} X^{|\mathrm{usat}(M)|} \cdot r^{|S \setminus \mathrm{usat}(M)|} \cdot s^{|S \cap \mathrm{usat}(M)|}. \tag{4.9}$$

Proof. Every matching $M \in \mathcal{M}[G]$ induces a certain set $\mathcal{C}_M \subseteq \mathcal{M}[G_{S,\ell}]$ of matchings in $G_{S,\ell}$, where each matching $N \in \mathcal{C}_M$ consists of M together with an extension by rake edges. The family $\{\mathcal{C}_M\}_{M \in \mathcal{M}[G]}$ is easily seen to partition $\mathcal{M}[G_{S,\ell}]$, and we obtain

$$\mu(G_{S,\ell}, X) = \sum_{M \in \mathcal{M}[G]} \underbrace{\sum_{N \in \mathcal{C}_M} X^{|\mathrm{usat}(N)|}}_{=:e(M)}. \tag{4.10}$$

Every matching $N \in \mathcal{C}_M$ consists of M and rake edges, which are added independently at each vertex $v \in S$. Hence, the expression $e(M)$ in (4.9) can be computed from the product of the individual extensions at each $v \in S$.

To compute the factor obtained by extensions at $v \in S$, we have to distinguish whether v is unmatched in M or not. The possible extensions at v are also shown in Figure 4.6.

$v \notin \mathrm{usat}(M)$: We can extend M at v by any subset of the ℓ rake edges not adjacent to v, as shown in Figure 4.6.a. In total, these 2^ℓ extensions contribute the factor

$$(1 + X^2)^\ell = (1 + X^2)^{\ell-1} r.$$

$v \in \mathrm{usat}(M)$: We have two choices for extending, shown in the right part of Figure 4.6:

- We can extend as in the case $v \notin \mathrm{usat}(M)$, and then we obtain the factor $X(1+X^2)^\ell$. Here, the additional factor X corresponds to the unmatched vertex v. This situation is shown in Figure 4.6.b.
- We can match v to one of its ℓ incident rake edges, say to $e = vz$ for a rake vertex z, as in Figure 4.6.c. Then we can choose a matching among the $\ell - 1$ rake edges not incident with z. This gives a factor of $\ell X(1 + X^2)^{\ell-1}$. Note that v is matched, but the vertex adjacent to z is not, yielding a factor of X.

In total, if $v \in \mathrm{usat}(M)$, we obtain a factor of

$$X(1 + X^2)^\ell + \ell X(1 + X^2)^{\ell-1} = X(1 + X^2)^{\ell-1} s$$

In each matching $N \in \mathcal{C}_M$, every unmatched vertex in $\bar{S} = V(G) \setminus S$ contributes a factor X. By multiplying the contributions of all $v \in V(G)$, we have thus shown that

$$\begin{aligned} e(M) &= f_\ell(X) \cdot X^{|\bar{S} \cap \mathrm{usat}(M)|} \cdot r^{|S \setminus \mathrm{usat}(M)|} \cdot (Xs)^{|S \cap \mathrm{usat}(M)|} \\ &= f_\ell(X) \cdot X^{|\mathrm{usat}(M)|} \cdot r^{|S \setminus \mathrm{usat}(M)|} \cdot s^{|S \cap \mathrm{usat}(M)|} \end{aligned}$$

and together with (4.10), this proves the claim. □

Observe that, due to the factor f_ℓ, the expression $\mu(G_{S,\ell})$ is not a polynomial in the indeterminates X and ℓ. We define a new expression p, which is a polynomial $p \in \mathbb{Z}[X, \ell]$, by removing this factor.

$$p(X, \ell) := \sum_{M \in \mathcal{M}[G]} X^{|\mathrm{usat}(M)|} \cdot r^{|S \setminus \mathrm{usat}(M)|} \cdot s^{|S \cap \mathrm{usat}(M)|}. \tag{4.11}$$

Depending upon the concrete application, we will consider $p \in \mathbb{Z}[X, \ell]$ as a polynomial in the indeterminates ℓ and X, or as a polynomial $p \in (\mathbb{Z}[\ell])[X]$ in the indeterminate X with coefficients from $\mathbb{Z}[\ell]$. In this last case, we write

$$p = \sum_{i=0}^{n} a_i X^i$$

with coefficients $a_i \in \mathbb{Z}[\ell]$ for $i \in \mathbb{N}$ that are in turn polynomials. Then we define

$$[p]_k := \sum_{i=0}^{k} a_i X^i \tag{4.12}$$

as the restriction of p to its first $k+1$ coefficients. For later use, let us observe the following simple fact about $[p]_k$, considered as a polynomial $[p]_k \in \mathbb{Z}[X, \ell]$.

Fact 4.16. *For $i, j \in \mathbb{N}$, every monomial $\ell^i X^j$ appearing in $[p]_k$ satisfies $i \leq j \leq k$.*

Proof. Recall r and s from Lemma 4.15. The indeterminate ℓ appears in s with degree 1, but it does not appear in r. In the right-hand side of (4.11), every term containing a factor s^t, for $t \in \mathbb{N}$, also contains the factor X^t, because $|S \cap \mathrm{usat}(M)| \leq |\mathrm{usat}(M)|$ trivially holds. Hence, whenever $\ell^i X^j$ is a monomial in p, then $i \leq j$. Since the maximum degree of X in $[p]_k$ is k by definition, the claim follows. □

In the next lemma, we show that knowing the coefficients of $[p]_k$ allows to solve the instance (G, S, k) to #RestrDualMatch from the beginning of this subsection. After that, we will show how to compute $[p]_k$ with an oracle for #PlanarDualMatch.

Lemma 4.17. *Let $\mathcal{N}$ denote the set of (not necessarily k-defect) matchings in G with $\mathrm{usat}(M) \cap S = \emptyset$. For all $k \in \mathbb{N}$, we can compute the number of k-defect matchings in $\mathcal{N}$ in polynomial time when given the coefficients of $[p]_k$.*

Proof. For ease of presentation, assume first we knew *all* coefficients of p rather than only those of $[p]_k$. We will later show how to solve the problem when given only $[p]_k$.

Starting from p, we perform the substitution

$$\ell \leftarrow -(1+X^2) \tag{4.13}$$

to obtain a new polynomial $q \in \mathbb{Z}[X]$ from p. By definition of s (see Lemma 4.15), we have

$$s(X, -(1+X^2)) = 0, \tag{4.14}$$

so every matching $M \notin \mathcal{N}$ has zero weight in q. To see this, note that by (4.11), the weight of each matching $M \in \mathcal{M}[G]$ in p contains a factor $s^{|S \cap \mathrm{usat}(M)|}$. But due to (4.14), the corresponding term in q is non-zero only if $|S \cap \mathrm{usat}(M)| = 0$. We obtain

$$q = \sum_{M \in \mathcal{N}} X^{|\mathrm{usat}(M)|} \cdot (1+X^2)^{|S \setminus \mathrm{usat}(M)|}.$$

Since every $M \in \mathcal{N}$ satisfies $|S \setminus \mathrm{usat}(M)| = |S|$, this simplifies to

$$q = (1+X^2)^{|S|} \cdot \underbrace{\sum_{M \in \mathcal{N}} X^{|\mathrm{usat}(M)|}}_{=:q'} \tag{4.15}$$

and we can use standard polynomial division by $(1+X^2)^{|S|}$ to obtain

$$q' = q/(1+X^2)^{|S|}. \tag{4.16}$$

By (4.15), for all $k \in \mathbb{N}$, the coefficient of X^k in q' counts precisely the k-defect matchings in $\mathcal{N}$. This finishes the discussion of the idealized setting when all coefficients of p are known. Recall the three steps involved: The substitution in (4.13), the polynomial division in (4.16), and the extraction of the coefficient X^k from q'.

The full claim, when only $[p]_k$ rather than p is given, can be shown similarly, but some additional care has to be taken. First, we perform the substitution (4.13) on $[p]_k$ rather than p. This results in a polynomial $b \in \mathbb{Z}[X]$, for which we claim the following:

Claim 4.18. We have $[b]_k = [q]_k$.

Proof. Let $\Theta_{\leq i}$ for $i \in \mathbb{N}$ denote the set of monomials in p with degree $\leq i$ in X. The substitution (4.13) maps every monomial θ in the indeterminates X and ℓ to some polynomial $g_\theta \in \mathbb{Z}[X]$. Writing $a(\theta) \in \mathbb{Z}$ for the coefficient of θ in p, we obtain $q, b \in \mathbb{Z}[X]$ with

$$q = \sum_{\theta \in \Theta_{\leq n}} a(\theta) \cdot g_\theta, \tag{4.17}$$

$$b = \sum_{\theta \in \Theta_{\leq k}} a(\theta) \cdot g_\theta. \tag{4.18}$$

We can conclude that

$$[q]_k \underset{(4.17)}{=} \left[\sum_{\theta\in\Theta_{\leq n}} a(\theta)\cdot g_\theta\right]_k = \left[\sum_{\theta\in\Theta_{\leq k}} a(\theta)\cdot g_\theta\right]_k \underset{(4.18)}{=} [b]_k\,, \tag{4.19}$$

where the second identity holds since, whenever θ has degree i in X, for $i \in \mathbb{N}$, then g_θ contains a factor X^i. Hence, for $\theta \in \Theta_{\leq n} \setminus \Theta_{\leq k}$, no terms of the polynomial g_θ appear in $\left[\sum_{\theta\in\Theta_{\leq n}} a(\theta)\cdot g_\theta\right]_k$. □

Recall the polynomial q' from (4.16); it remains to apply polynomial division as in (4.16) to recover $[q']_k$ from $[b]_k$. To this end, we observe that the constant coefficient in $(1+X^2)^{|S|}$ is 1, and that all coefficients of $(1+X^2)^{|S|}$ can be computed by a closed formula. We can thus divide $[b]_k = [q]_k$ by $[(1+X^2)^{|S|}]_k$ via truncated polynomial division (Lemma 4.13) to obtain $[q']_k$, whose k-th coefficient counts the k-defect matchings in $\mathcal{N}$, as in the idealized setting discussed before. □

Using a combination of truncated polynomial division (Lemma 4.13) and interpolation, we compute the coefficients of $[p]_k$ with oracle access for #PlanarDualMatch. This completes the reduction from #RestrDualMatch to #PlanarDualMatch.

Lemma 4.19. *We can compute $[p]_k$ by a Turing fpt-reduction to* #PlanarDualMatch.

Proof. For ξ with $0 \leq \xi \leq k$, let $f_\xi \in \mathbb{Z}[X]$ be the evaluation of the expression f_ℓ defined in Lemma 4.15 at $\ell = \xi$. Define $p_\xi^{(k)} \in \mathbb{Z}[X]$ by

$$p_\xi^{(k)} := [\mu(G_{S,\xi})/f_\xi]_k\,. \tag{4.20}$$

Claim 4.20. We have $p_\xi^{(k)} = [p(\cdot,\xi)]_k = [p]_k(\cdot,\xi)$.

Proof. The first identity holds by the definition of p in (4.11), and by the definition of $p_\xi^{(k)}$. The second identity holds because, for all $t \in \mathbb{N}$, the coefficient of X^t in p is a polynomial in ℓ and does not depend on X. Hence we may arbitrarily interchange (i) the operation of substituting ℓ by expressions not depending on X (and by numbers $\xi \in \mathbb{N}$ in particular), and (ii) the operation of truncating to the first k coefficients. □

Recall that $a_t \in \mathbb{Z}[\ell]$ for $t \in \mathbb{N}$ denotes the coefficient of X^t in p, which has degree at most k (in the indeterminate ℓ) by Fact 4.16. Hence, for fixed $t \in \mathbb{N}$, if we knew the values

$$a_t(0), \ldots, a_t(k),$$

we could recover the coefficients of $a_t \in \mathbb{Z}[\ell]$ via univariate polynomial interpolation (Lemma 1.35). But for $0 \leq \xi, t \leq k$, we can obtain the value $a_t(\xi)$ as the coefficient of X^t in $p_\xi^{(k)}$. This follows from Claim 4.20. It remains to compute the polynomials

$$p_0^{(k)}, \ldots, p_k^{(k)}$$

with an oracle for #PlanarDualMatch: First, we observe that the constant coefficient in f_ξ is 1 for all $0 \leq \xi \leq k$, so we can apply the definition of $p_\xi^{(k)}$ from (4.20) and truncated polynomial division (Lemma 4.13) to compute $p_\xi^{(k)}$ from $[\mu(G_{S,\xi})]_k$ and f_ξ.

It remains only to compute $[\mu(G_{S,\xi})]_k$ and f_ξ. Note that the coefficients of f_ξ admit a closed expression by definition, and that $[\mu(G_{S,\xi})]_k$ can be computed by querying the oracle for #PlanarDualMatch to obtain the number of matchings in $G_{S,\xi}$ with $0, \ldots, k$ defects. □

We recapitulate the proof in the following.

Proof of Theorem 4.10. By Theorem 4.1, the problem PerfMatch^{0,1}/apex is #W[1]-hard, even on instances restricted as in Remark 4.8, and we have established the reduction chain

$$\text{PerfMatch}^{0,1}/\text{apex} \leq_{fpt}^{lin} \#\mathsf{RestrDualMatch} \leq_{fpt}^{lin} \#\mathsf{PlanarDualMatch} :$$

The first reduction was shown in Lemma 4.12. By Lemma 4.19, we can use oracle calls to #PlanarDualMatch with maximum parameter k to compute a polynomial $[p]_k$, and by Lemma 4.17, the coefficients of $[p]_k$ allow to recover the solution to #RestrDualMatch in polynomial time. These two steps establish the second reduction in the above chain. □

4.3. Apices with few adjacent faces

Our next result complements the #W[1]-hardness of PerfMatch/apex shown in Theorem 4.1 and is motivated by a refined structural decomposition of graphs $G \in \text{Excl}[H]$ for 1-apex graphs H. The decompositions guaranteed by [DHK09] for such graphs are similar to those for general H-minor-free graphs from the Graph Structure Theorem, with the exception of "quasi-vortices", which are vortices that only require a bound on the *treewidth* rather than on their pathwidth. (That is, the tree decompositions are not restricted to be paths.) However, any apex in the decomposition can only connect to the vertices of a quasi-vortex, so every apex attaches to at most $c = c(H)$ faces of the base graph.

We prove Theorem 4.21 and present an fpt-algorithm for a restricted version of the problem PerfMatch/apex in which every apex can see only a bounded number of faces.

Theorem 4.21 (restated from page 85)**.** *Given as input a graph G, a set $A \subseteq V(G)$, and a drawing of $G - A$ on a surface of genus γ such that A is adjacent to at most s faces, we can compute* PerfMatch(G) *in time $f(|A|, \gamma, s) \cdot n^3$ for a computable function f.*

Remark 4.22. We may assume that every edge $av \in E(G)$ with $a \in A$ and $v \in V(G) \setminus A$ has weight 1: Otherwise, replace av by a path ar_1r_2v with fresh vertices r_1, r_2, together with edges ar_1 and r_1r_2 of unit weight, and an edge r_2v of weight $w(e)$, as in Remark 1.30. This clearly preserves the apex number, the value of PerfMatch, and ensures that every apex is only incident with unweighted edges.

To proceed, we require a variant of multivariate polynomial interpolation (Lemma 1.38) that applies to a setting where we do not require the values of *all* coefficients, but rather only those in a "slice" of total degree k, for fixed $k \in \mathbb{N}$. Here, the polynomial p to be

interpolated features a distinguished indeterminate X, and we wish to extract the coefficient a_k of X^k, which is in turn a polynomial. Under certain restrictions, this can be achieved with $f(k) \cdot n$ evaluations, where n denotes the degree of X in p.

Lemma 4.23 (Sliced grid interpolation)**.** *Let $p \in \mathbb{Z}[X, \lambda]$ be a multivariate polynomial in the indeterminates X and $\lambda = (\lambda_1, \ldots, \lambda_t)$. Consider $p \in (\mathbb{Z}[\lambda])[X]$ and assume that*

- *p has degree n in X, and that*
- *for all $s \in \mathbb{N}$, the coefficient $a_s \in \mathbb{Z}[\lambda]$ of X^s in p has total degree at most s.*

Let $k \in \mathbb{N}$ be a given parameter, and let

$$\Xi = \Xi_0 \times \ldots \times \Xi_t \subseteq \mathbb{Q}^{t+1}$$

with $|\Xi_0| = n + 1$ and $|\Xi_i| = k + 1$ for all $i > 0$. Then we can compute the coefficients of the polynomial $a_k \in \mathbb{Z}[\lambda]$ with $\mathcal{O}(|\Xi|^3)$ arithmetic operations when given as input the set

$$\{(\xi, p(\xi)) \mid \xi \in \Xi\}.$$

Proof. We consider the grid Ξ' defined by removing the first component from Ξ:

$$\Xi' = \Xi_1 \times \ldots \times \Xi_t.$$

Let us observe that $p(\cdot, \xi') \in \mathbb{Z}[X]$ holds for $\xi' \in \Xi'$. Write $\Xi_0 = \{c_0, \ldots, c_n\}$ and observe that, for each fixed $\xi' \in \Xi'$, our input contains all evaluations

$$p(c_0, \xi'), \ldots, p(c_n, \xi'),$$

so we can use univariate interpolation (see Lemma 1.35) to determine the coefficient of X^k in $p(\cdot, \xi')$. This coefficient is equal to $a_k(\xi')$ by definition.

By performing this process for all $\xi' \in \Xi'$, we can evaluate $a_k(\xi')$ on all $\xi' \in \Xi'$, and hence interpolate the polynomial $a_k \in \mathbb{Z}[\lambda]$ via grid interpolation (Lemma 1.38). □

For our algorithm, we first consider the special case that A is an independent set; the full algorithm is then obtained by reduction to this case. The ideas used in the following lemma bear some similarity to those used in an algorithm for counting subgraphs of bounded vertex-cover number, which we will present in Section 6.1.

Lemma 4.24. *Let G be an edge-weighted graph, given as input together with*

- *an independent set $A \subseteq V(G)$ of size k,*
- *an embedding π of $H = G - A$ on a surface of genus γ, and faces $C_1, \ldots, C_s$ in π that contain all neighbors of A.*

Then we can compute $\mathrm{PerfMatch}(G)$ *in time $k^{\mathcal{O}(2^k)} \cdot 2^{\mathcal{O}(\gamma+s)} \cdot n^{\omega+1}$.*

Proof. Let $\mathcal{DM}_k[H]$ denote the set of all k-defect matchings of H. By Remark 4.22, we can assume that all edges incident with A have unit weight. Let

$$\mathcal{C} = \{M \in \mathcal{DM}_k[H] \mid \text{usat}(M) \subseteq N_G(A)\}.$$

Given any matching $M \in \mathcal{C}$, let $t(M)$ denote the *type* of N, which is defined as the following *multiset* with precisely k elements from 2^A:

$$t(M) = \{N_G(v) \cap A \mid v \in \text{usat}(M)\}.$$

For the set of all such types, we write

$$\mathcal{T} = \{t(M) \mid M \in \mathcal{C}\}$$

and observe that $|\mathcal{T}| \leq (2^k)^k = 2^{k^2}$. For $t \in \mathcal{T}$, define a graph F_t as follows: Create an independent set $[k]$, corresponding to A. Then, for each $N \in t$, create a vertex v_N that is adjacent to all of $N \subseteq [k]$. We note that every *perfect* matching $M \in \mathcal{PM}[G]$ can be decomposed uniquely as

$$M = B(M) \dot\cup I(M)$$

with a k-defect matching $B(M) \in \mathcal{C}$ and a perfect matching $I(M) \in \mathcal{PM}[F_{t(B(M))}]$ with

$$\begin{aligned} B(M) &= M - A, \\ I(M) &= M[A \cup \text{usat}(B(M))]. \end{aligned}$$

For $t \in \mathcal{T}$, let

$$\begin{aligned} \mathcal{C}_t &= \{M \in \mathcal{C} \mid t(M) = t\}, \\ P_t &:= \sum_{N \in \mathcal{C}_t} \prod_{e \in N} w(e). \end{aligned}$$

It is clear that $\{\mathcal{C}_t\}_{t \in \mathcal{T}}$ partitions $\mathcal{C}$, and this implies

$$\text{PerfMatch}(G) = \sum_{t \in \mathcal{T}} P_t \cdot \text{PerfMatch}(F_t). \tag{4.21}$$

To see this, note that each perfect matching of type t can be obtained by extending some matching $M \in \mathcal{C}_t$ (all of which have k defects) by a perfect matching from $\text{usat}(M)$ to A, which is precisely a perfect matching of F_t. Note that we require here that edges between $\text{usat}(M)$ and A have unit weight, otherwise the graphs F_t would have to be edge-weighted as well and might no longer depend on t only, but would also have to incorporate the edge-weights of G.

Since $|E(F_t)| \leq k^2$, we can compute $\text{PerfMatch}(F_t)$ in time $2^{\mathcal{O}(k^2)}$ by brute force for all $t \in \mathcal{T}$. Hence, we can use (4.21) to determine $\text{PerfMatch}(G)$ in time $|\mathcal{T}| \cdot 2^{\mathcal{O}(k^2)}$ if we know P_t for all $t \in \mathcal{T}$. In the remainder of this proof, we show how to compute P_t by using multivariate polynomial interpolation and the algorithm for MatchSum presented in

Theorem 2.41. To this end, define indeterminates corresponding to subsets of the apices:

$$\lambda = \{\lambda_S \mid S \subseteq A\}.$$

Let X denote an additional distinguished indeterminate, and define the following polynomial $p \in \mathbb{Z}[X, \lambda]$. In this definition, we abbreviate $w(M) := \prod_{e \in M} w(e)$.

$$p(X, \lambda) := \sum_{M \in \mathcal{C}} w(M) \cdot X^{|\mathrm{usat}(M)|} \cdot \prod_{v \in \mathrm{usat}(M)} \lambda_{N_G(v) \cap A}. \tag{4.22}$$

For each type $t \in \mathcal{T}$, say $t = \{N_1, \ldots, N_k\}$, we observe that the coefficient of

$$X^k \cdot \lambda_{N_1} \cdot \ldots \cdot \lambda_{N_k}$$

in p is equal to P_t. Hence, we can extract P_t for all $t \in \mathcal{T}$ from the coefficients of the monomials in p that have degree exactly k in X. Let us denote these monomials by $\mathfrak{N}$, and observe that each monomial $\nu \in \mathfrak{N}$ has total degree k in λ by definition of p in (4.22).

If we can evaluate p on the elements (r, ξ) from the grid

$$\Xi = [n+1] \times [k+1]^{2^{|A|}},$$

then we can compute the coefficients of all $\nu \in \mathfrak{N}$ in p, and thus P_t for all $t \in \mathcal{T}$, by sliced grid interpolation via Lemma 4.23. Note that $|\Xi| \in \mathcal{O}(n \cdot k^{2^k})$. We compute these evaluations $p(r, \xi)$ as

$$p(r, \xi) = \mathrm{MatchSum}(H'(r, \xi)),$$

where the vertex-weighted graph $H'(r, \xi)$ is obtained from H via the weight function

$$w(v) := \begin{cases} 0 & \text{if } v \notin N_G(A), \\ r \cdot \xi_{N_G(v) \cap A} & \text{otherwise.} \end{cases}$$

Since all vertices with non-zero weight in $H'(r, \xi)$ are contained in the faces $C_1, \ldots, C_s$, we can compute $\mathrm{MatchSum}(H')$ in time $\mathcal{O}(4^\gamma \cdot 2^s \cdot n^\omega)$ with Theorem 4.21. We obtain the values P_t for all $t \in \mathcal{T}$, so we obtain $\mathrm{PerfMatch}(G)$ via (4.21) in the required time. □

It remains to lift Lemma 4.24 to the case that A is not an independent set. This follows easily from the fact that, whenever $E(G) = E \dot\cup E'$, then every perfect matching $M \in \mathcal{PM}[G]$ must match every vertex $v \in V(G)$ into exactly one of the sets E or E'.

Proof of Theorem 4.21. Let $\mathcal{A} = \mathcal{M}[G[A]]$ denote the set of (not necessarily perfect) matchings of the induced subgraph $G[A]$, and note that $|\mathcal{A}| \leq 2^{k^2}$. For $M \in \mathcal{A}$, let

$$a_M = \mathrm{PerfMatch}(G_M),$$

where G_M is defined by keeping from A only $\mathrm{usat}(M)$, and then deleting all edges between the remaining vertices. We can compute a_M by Lemma 4.24 since the remaining part of A

in G_M is an independent set. It is also easily verified that

$$\mathrm{PerfMatch}(G) = \sum_{M \in \mathcal{A}} a_M \cdot \prod_{e \in M} w(e),$$

so we can compute PerfMatch as a linear combination of 2^{k^2} values, each of which can be computed by Lemma 4.24. □

It should be mentioned that this last step could also be shown using combined signatures, but we decided to give a self-contained proof.

Part II.

Counting small subgraphs

Introduction to Part II

Recall that, in Part I, we were given a graph G together with a structural parameter, namely, its Hadwiger number, and we wished to count occurrences of a large and fixed type of subgraph in G, namely perfect matchings. In the present part, we consider an orthogonal setting: We wish to count subgraph patterns H from an *arbitrary* fixed class $\mathcal{H}$, and rather than parameterizing by structural properties of G, we consider the size of the pattern graph H as parameter. Note that all reasonable parameters of H are bounded by a function of $|V(H)|$, and by restricting H to be chosen from $\mathcal{H}$, we can additionally impose fixed restrictions on the structure of H, such as a constant bound on its treewidth. For simplicity, let us always assume in the following that $\mathcal{H}$ is recursively enumerable. This gives rise to the following problems.

Problem 4.1 ($\#\mathsf{Sub}(\mathcal{H})$ for fixed graph class $\mathcal{H}$)**.** Given graphs H and G as input, with $H \in \mathcal{H}$, compute the number of H-copies in G, parameterized by $|V(H)|$. We write $\#\mathsf{Sub}(H \to G)$ for this number.

For instance, the problem $\#\mathsf{Sub}(\mathcal{K})$ with the class $\mathcal{K}$ of complete graphs is simply the problem $\#\mathsf{Clique}$ used in our definition of $\#\mathsf{W}[1]$, and it is $\#\mathsf{W}[1]$-complete by definition. Deciding the existence of a k-clique is however $\mathsf{W}[1]$-complete as well, and therefore the $\#\mathsf{W}[1]$-hardness of $\#\mathsf{Clique}$ does not come as a big surprise. More interestingly, it was shown in [FG04] that the problems $\#\mathsf{Sub}(\mathcal{H}_{paths})$ and $\#\mathsf{Sub}(\mathcal{H}_{cycles})$ of counting paths and cycles are $\#\mathsf{W}[1]$-complete as well. Since the decision versions of these problems are fixed-parameter tractable, e.g., by application of the color coding technique [AYZ95], the counting problems cannot be $\#\mathsf{W}[1]$-complete under parsimonious reductions unless $\mathsf{FPT} = \mathsf{W}[1]$, and this makes these results particularly interesting.

Starting from these initial hardness results for paths and cycles, we wondered which restrictions on the graph class $\mathcal{H}$ make the problem $\#\mathsf{Sub}(\mathcal{H})$ easy. This is also explicitly asked in [CTW08]. Note, for instance, that a constant bound on the *treewidth* of $\mathcal{H}$ is not sufficient to make $\#\mathsf{Sub}(\mathcal{H})$ tractable, since even counting paths (which are trees) of size k is $\#\mathsf{W}[1]$-complete. The *decision* problem however does become fixed-parameter tractable under such a fixed treewidth bound [AYZ95], and we hence obtain another example for the commonplace remark that counting is harder than deciding.

Indeed, counting is *so* hard that the few known graph classes $\mathcal{H}$ with fixed-parameter tractable $\#\mathsf{Sub}(\mathcal{H})$ are heavily constrained. For instance, we can count the copies of the k-star S_k in a graph G in linear time: The star S_k is an independent k-set together with

one additional "center" vertex adjacent to this set. To count copies of S_k in G, simply guess a center $v \in V(G)$, and then observe that there are $\binom{\deg_G(v)}{k}$ possibilities to extend v to a copy of S_k in G. The stars S_k are not only trees, but they even admit a vertex-cover of size 1, and as it turns out, the vertex-cover number is central to the tractability of $\#\mathsf{Sub}(\mathcal{H})$. In particular, the simple algorithm that allowed us to count stars can also be generalized to pattern graphs with bounded vertex-covers, noted also in [WW13, KLL13]. We also give a simple self-contained algorithm for this case in Section 6.1.

Theorem 6.1. *Let H be a k-vertex graph with a vertex-cover of size τ, and let G be a graph on n vertices, for $n \in \mathbb{N}$. Then we can determine the number of H-copies in G in time $k^{2^{\mathcal{O}(\tau)}} n^{\tau+\mathcal{O}(1)}$. Note that, if $\tau = \mathcal{O}(1)$, then this running time is polynomial.*

If a constant bound on the vertex-cover number of $\mathcal{H}$ makes $\#\mathsf{Sub}(\mathcal{H})$ tractable, then it is natural to ask whether the problem is conversely $\#\mathsf{W}[1]$-hard if such a bound is *not* given for $\mathcal{H}$. For a minimal example of such a problem, it was conjectured in [FG04] that the problem $\#\mathsf{Match}$ is $\#\mathsf{W}[1]$-complete, where $\#\mathsf{Match} = \#\mathsf{Sub}(\mathcal{H}_{match})$ denotes the problem of counting k-matchings. This conjecture was solved by the author [Cur13], following up on joint work with Markus Bläser for a weighted variant of the problem [BC12]. We will give a novel proof of this result in Section 5.2, and we include the old proof in Appendix A for completeness. The problem of counting k-matchings will later be revisited in full depth.

In view of the hardness of $\#\mathsf{Match}$, and following the intuition that $\#\mathsf{Sub}(\mathcal{H})$ should admit a fixed-parameter reduction to $\#\mathsf{Sub}(\mathcal{H}')$ if the graphs in $\mathcal{H}$ appear as subgraphs in $\mathcal{H}'$, it is natural to conjecture that $\#\mathsf{Sub}(\mathcal{H}')$ is $\#\mathsf{W}[1]$-complete whenever $\mathcal{H}'$ contains all matchings as subgraphs. Note that this is equivalent to requiring the vertex-cover number of $\mathcal{H}'$ to be unbounded. The main outcome of this part is a proof of this conjecture.

Theorem 6.11. *Let $\mathcal{H}$ be a recursively enumerable graph class. If $\mathsf{FPT} \neq \#\mathsf{W}[1]$, then the following are equivalent:*

- *$\#\mathsf{Sub}(\mathcal{H})$ is polynomial-time solvable.*
- *$\#\mathsf{Sub}(\mathcal{H})$ is fixed-parameter tractable when parameterized by $|V(H)|$.*
- *$\mathcal{H}$ has bounded vertex-cover number.*

This dichotomy theorem exhaustively classifies the complexity of $\#\mathsf{Sub}(\mathcal{H})$, and it shows in particular that there exist no classes $\mathcal{H}$ such that $\#\mathsf{Sub}(\mathcal{H})$ is fixed-parameter tractable, but not polynomial-time solvable. Thus, assuming $\mathsf{FPT} \neq \#\mathsf{W}[1]$, we can precisely say when $\#\mathsf{Sub}(\mathcal{H})$ admits a polynomial-time algorithm. Similar results of this type are known in the literature:

Homomorphisms: If the treewidth of the *cores* of $\mathcal{H}$ is bounded, then we can decide in polynomial time whether $H \in \mathcal{H}$ admits a homomorphism into G. Otherwise, the problem is #W[1]-complete [Gro07]. For the problem of *counting* homomorphisms, a bound on the treewidth of $\mathcal{H}$ itself is required for polynomial-time solvability, otherwise the problem is #W[1]-hard [DJ04].

Induced subgraphs: If $\mathcal{H}$ is finite, then we can trivially find and count induced copies of $H \in \mathcal{H}$ in a graph G in polynomial time. On the other hand, if $\mathcal{H}$ is infinite, then it was shown that the problems of counting induced H-copies and deciding their existence are #W[1]-complete and W[1]-complete, respectively [CTW08].

Colorful subgraphs: A classification of tractable database queries from [GSS01] can be rephrased in terms of finding a vertex-colorful graph $H \in \mathcal{H}$ in a vertex-colored graph G. Again, a bound on the treewidth of $\mathcal{H}$ yields a polynomial-time algorithm, while the problem becomes #W[1]-complete if $\mathcal{H}$ has unbounded treewidth.

For all problems above, the assumption FPT $\neq$ #W[1] allows to identify precisely the polynomial-time solvable cases. It might seem too strong to assume a statement in parameterized complexity in order to identify polynomial-time solvability, but it is known that weaker assumptions, such as P $\neq$ NP or FP $\neq$ #P, do not always suffice for such dichotomies: As shown in [CTW08], if FP $\neq$ #P holds, then there exist graph classes $\mathcal{H}$ such that counting *induced* subgraphs from $\mathcal{H}$ is #P-intermediate. That is, the problem is in #P, but not in FP, and it is not #P-complete.

Let us return to counting subgraphs, and in particular, to Theorem 6.11. To prove this theorem, we distinguish three types of graph classes. Firstly, if $\mathcal{H}$ has bounded vertex-cover number, then we can apply the algorithm from Theorem 6.1.

If $\mathcal{H}$ has unbounded treewidth, then we prove in Section 5.1 that a vertex-colorful variant of subgraph counting is #W[1]-hard, in a way that is similar to [GSS01, DJ04, Gro07, Mee14]. By inclusion-exclusion, we then obtain #W[1]-hardness for the uncolored case #Sub($\mathcal{H}$) as well. Building upon this reduction, we will also obtain #W[1]-hardness for counting k-matchings in Section 5.2, and we will explain this more in a few pages.

If $\mathcal{H}$ has bounded treewidth, but unbounded vertex-cover number, then we reduce counting k-matchings to #Sub($\mathcal{H}$) in Section 6.2. This is the most involved part of the reduction, and it heavily depends on certain *k-matching gadgets* whose existence needs to be proven in an involved graph-theoretical argument.

Vertex-colorful subgraphs

To prove #W[1]-completeness for #Sub($\mathcal{H}$) if $\mathcal{H}$ has unbounded treewidth, we consider the problem #PartitionedSub($\mathcal{H}$), a vertex-colored version of #Sub($\mathcal{H}$). In this variant, we are given vertex-colored graphs H and G on the same color set, where H has precisely one vertex of each color, and where the uncolored graph underlying H is contained in $\mathcal{H}$. Our task is then to count subgraphs F of G such that H admits an isomorphism to F which maps each vertex in H to a vertex in F of the same color. The parameter is again $|V(H)|$.

This problem can be solved in polynomial time by dynamic programming if the treewidth of $\mathcal{H}$ is bounded by $\mathcal{O}(1)$, shown in [AR02]. On the other hand, we can show by relatively standard techniques that #PartitionedSub($\mathcal{H}$) is #W[1]-hard whenever $\mathcal{H}$ has unbounded treewidth: First, we show #W[1]-hardness of #PartitionedSub on the class of square grids. Then we prove that the complexity of this problem is minor-monotone, i.e., we show that #PartitionedSub($\mathcal{H}$) can be reduced to #PartitionedSub($\mathcal{H}'$) if, for every graph $H \in \mathcal{H}$, there is some graph $H' \in \mathcal{H}'$ such that $H \preceq H'$. By the Excluded Grid Theorem, every graph class $\mathcal{H}$ of unbounded treewidth includes all square grids as minors, and we consequently obtain #W[1]-hardness of #PartitionedSub($\mathcal{H}$). This is by now a well-trodden line of reasoning, see [GSS01, DJ04, Gro07, Mee14]. Finally, we observe that #PartitionedSub($\mathcal{H}$) can be reduced to #Sub($\mathcal{H}$) by the inclusion-exclusion principle, and we obtain:

Theorem 5.6. *The problems* #PartitionedSub($\mathcal{H}$) *and* #Sub($\mathcal{H}$) *are* #W[1]*-complete whenever* $\mathcal{H}$ *is recursively enumerable and has unbounded treewidth.*

Edge-colorful matchings

As noted before, the result for #PartitionedSub($\mathcal{H}$) does not come as a surprise in view of the previously cited results in the literature. However, if we color *edges* instead of vertices, we obtain a somewhat different situation. For instance, the algorithm for bounded-treewidth patterns from [AR02] fails when considering *edge*-colored graphs instead of their vertex-colored counterparts.

In fact, we can even show that the problem of counting edge-colorful *matchings* is #W[1]-complete: Here, we are given a $[k]$-edge-colored graph G, for $k \in \mathbb{N}$, and we wish to count the matchings in G that have precisely one edge from each color. This stands in contrast with the *vertex*-colorful variant of counting matchings, which is fixed-parameter tractable, since matchings are forests. The edge-colorful version can be reduced to the uncolored variant by the inclusion-exclusion principle, as seen in Lemma 1.33.

Our proof of this result, shown in Section 5.2, proceeds by reduction from #PartitionedSub on the class of 3-regular graphs, for which an almost-tight lower bound of $f(k) \cdot n^{o(k/\log k)}$ under #ETH is known [Mar10]. Our reduction preserves this lower bound, and additionally, we can also prove #W[1]-hardness when G may be assumed to be bipartite, which is later required for the reduction to #Sub($\mathcal{H}$) in Section 6.2. In fact, the main reason for us to revisit the problem #Match was because the original proof did not support a restriction to bipartite graphs. We obtain the following results.

Theorem 5.22. *The problem* #EdgeColMatch *and its uncolored variant* #Match *are* #W[1]*-complete and admit no algorithm with running time* $f(k) \cdot n^{o(k/\log k)}$ *unless* #ETH *fails. Here, we may even assume* G *to be bipartite.*

Using the notion of combined signatures from Section 2.3, the proof of Theorem 5.22 is quite transparent. In particular, it allows to circumvent all of the machinery used in the original #W[1]-hardness proof for #Match by the author [Cur13], which is included in Appendix A for the sake of completeness.

Using fpt-reductions from #Match that incur linear blowup, we also provide new #W[1]-hardness proofs for the problems of counting (directed or undirected) paths and cycles. These are somewhat simpler than the original proofs in [FG04], and they allow to transfer the lower bound of $f(k)n^{o(k/\log k)}$ for #Match to the four target problems, where no such bounds were known before.

Theorem 5.29. *The problems* #DirCycle*,* #UndirCycle*,* #UndirPath*,* #DirPath *of counting directed/undirected paths/cycles of length k are all* #W[1]*-hard and admit no algorithms with running time $f(k)n^{o(k/\log k)}$ unless* #ETH *fails.*

We could also aim at a dichotomy for edge-colorful subgraph counting, but we decided to focus on edge-colorful *matchings*, as these allow us to prove hardness for uncolored k-matchings, which is ultimately required for the uncolored case of subgraph counting.

From matchings to bounded-treewidth graphs

Let us return to the problem #Sub($\mathcal{H}$) introduced before. We know by Theorem 5.6 that #Sub($\mathcal{H}$) is #W[1]-hard when $\mathcal{H}$ has unbounded treewidth, and we know by Theorem 6.1 that it is polynomial-time solvable when $\mathcal{H}$ has bounded vertex-cover number. In Section 6.2, we handle the remaining classes $\mathcal{H}$, i.e., we show #W[1]-hardness of #Sub($\mathcal{H}$) if $\mathcal{H}$ has bounded treewidth and unbounded vertex-cover number. To this end, we reduce from the problem #BipMatch of counting bipartite k-matchings, which we have shown to be #W[1]-hard in Theorem 5.22.

We can show with a Ramsey argument that, if $\mathcal{H}$ has bounded treewidth and unbounded vertex-cover number, then $\mathcal{H}$ contains induced matchings of all sizes, i.e., for all $k \in \mathbb{N}$, there is some graph $H \in \mathcal{H}$ which contains an induced k-matching. In Section 6.2, we use this to reduce #BipMatch to #Sub($\mathcal{H}$): Given a graph $H \in \mathcal{H}$ that contains an induced k-matching M, let us consider the following attempt at counting copies of M in a graph G when given an oracle for counting H-copies in arbitrary graphs.

1. Partition $V(H) = V(M) \cup C$, where C is the "remainder" of H after deleting M.
2. Construct a graph G' by adding a vertex-disjoint copy of $H[C]$ to G on a vertex set C' that is connected to all vertices of G.
3. By a simple inclusion-exclusion argument, we can use the oracle to count the H-copies F in G' that use all vertices and edges of the $H[C]$-copy in G'. These are precisely those copies F where $F[C']$ is isomorphic to $H[C]$.

One might be tempted to believe now that the remainder of F, i.e., the induced subgraph $F[G]$ contained in the copy of G in G', is in fact isomorphic to M. If this were true, then we could indeed count k-matchings by reduction to $\#\mathsf{Sub}(\mathcal{H})$, since each matching M in G would correspond to a fixed number $\alpha \in \mathbb{N}$ of H-copies F in G' with $F[C'] \simeq H[C]$.

However, this argument fails: Given H and C as above, we generally cannot rule out the existence of a set $D \subseteq V(H)$ such that $H[D] \simeq H[C]$, but $H - D$ is not a k-matching. In other words, even if we count H-copies F in G' with $F[C'] \simeq H[C]$, then it is not guaranteed that the subgraph $F[G]$ is a k-matching, and we cannot hope to count k-matchings of G. Instead, we count occurrences of *some* unknown set of graphs on $2k$ vertices in G, and we can only guarantee that this set includes the k-matchings.

To overcome this, we introduce a general machinery of *k-matching gadgets*; these are pairs (H, C) where H is a graph and $C \subseteq V(H)$ is such that the problem mentioned above does not occur. That is, whenever we partition $V(H)$ into sets D and B such that $H[D] \simeq H[C]$ and B satisfies certain technical conditions, then B is in fact a k-matching. We then show, given oracle access to counting H-copies, how to count H-copies F in G' such that the induced subgraph $F[G]$ satisfies the technical conditions, and this allows us to count k-matchings as in the idealized setting described above. This step requires a combination of interpolation and the inclusion-exclusion principle.

It remains to prove the existence of k-matching gadgets in classes $\mathcal{H}$ of bounded treewidth and unbounded vertex-cover number. This is done by a detailed graph-theoretic study by Dániel Marx, which is included in Appendix B, and for which the author of this thesis does not claim any contribution.

Notes

This part is based on joint work with Dániel Marx, partly carried out when the author was visiting him at MTA SZTAKI Budapest in 2013. This content has also appeared in [CM14], except for Section 5.2, which differs substantially from the published version.

In Section 6.2, an involved graph-theoretic proof is necessary for the reduction from $\#\mathsf{BipMatch}$ to $\#\mathsf{Sub}(\mathcal{H})$ on classes $\mathcal{H}$ with unbounded vertex-cover number and bounded treewidth. This part was contributed by Dániel Marx, and it is contained in Appendix B.

Recently, problems of counting induced subgraphs that satisfy certain fixed properties have been been studied in [JM15b, Mee14, JM15a]. These papers identify some classes of properties that make the problem $\#\mathsf{W}[1]$-hard. As an example, it was shown in [JM14] that counting k-vertex induced subgraphs with an odd number of edges is $\#\mathsf{W}[1]$-hard. The findings obtained in this line of research are orthogonal to ours, apart from a result in [Mee14] that can be considered as an independent $\#\mathsf{W}[1]$-hardness proof for Theorem 5.6.

5. Colored subgraphs

We consider the colored subgraph counting problems mentioned in the introduction to Part II. In Section 5.1, we consider the subgraph counting problems $\#\mathsf{PartitionedSub}(\mathcal{H})$ for fixed classes $\mathcal{H}$. Based on our findings in this section, we will then study the problem of counting edge-colorful matchings in Section 5.2 and use this to derive $\#\mathsf{W}[1]$-hardness for counting uncolored k-cycles and k-paths.

5.1. Vertex-colorful subgraphs

In this subsection, we study *vertex*-colorful counting problems. Rather than considering arbitrary color sets, we restrict ourselves to the color set $[k]$ for $k \in \mathbb{N}$. For simplicity, we identify vertices of colorful graphs with their colors.[1]

For the problem of counting vertex-colorful subgraphs, we can define two distinct versions, which we will soon observe to be equivalent. In the following, let $\mathcal{H}$ be an arbitrary fixed class of uncolored graphs.

Problem 5.1 ($\#\mathsf{ColorfulSub}(\mathcal{H})$)**.** Let $H \in \mathcal{H}$ be a graph on k vertices and let G be a $[k]$-vertex-colored graph. We wish to count *colorful* H-copies in G. These are H-copies in G that have exactly one vertex of each color $i \in [k]$.

For any graph class $\mathcal{H}$, we can use Lemma 1.34 on page 44 to show

$$\#\mathsf{ColorfulSub}(\mathcal{H}) \leq^{lin}_{fpt} \#\mathsf{Sub}(\mathcal{H}). \tag{5.1}$$

In the following, we define a more restricted version of vertex-colorful subgraph counting. This version will be used in the remainder of this chapter.

Problem 5.2 ($\#\mathsf{PartitionedSub}(\mathcal{H})$)**.** Let H and G be $[k]$-vertex-colored graphs such that H is colorful and the uncolored graph underlying H is contained in $\mathcal{H}$. We wish to count *color-preserving* H-copies F in G. These are subgraphs F of G that admit an isomorphism $f : V(H) \to V(F)$ such that, for all $v \in V(H)$, the vertex v has the same color as $f(v)$. We denote their number by $\#\mathsf{PartitionedSub}(H \to G)$.

[1] This allows us to contract linguistic accidents such as "the edge between the vertex of color i and the vertex of color j" to a mere "the edge ij".

When counting color-preserving copies of a vertex-colored graph H in G, it is clear that only those edges of G are relevant whose endpoints are colored such that H also has an edge between these colors. This gives rise to the following observation.

Remark 5.3. Let H and G be $[k]$-vertex-colored and let F be a subgraph of G that is color-preserving isomorphic to H. If $uv \in E(F)$ is an edge with endpoints of colors i and j, for some $i, j \in [k]$, then the edge ij is present in H. We may therefore assume that $E_{i,j}(G) = \emptyset$ if $ij \notin E(H)$. If this is not the case, we may delete edges in $E_{i,j}(G)$ with $ij \notin E(H)$ to ensure this.

If G is preprocessed as in Remark 5.3, and if H_u is the uncolored graph underlying H, then the colorful H_u-copies in G correspond to the color-preserving H-copies in G.

Lemma 5.4. *Let H and G be $[k]$-vertex-colored graphs, and let H_u denote the uncolored graph underlying H. Let G' be obtained from G by deleting all edges in $E_{i,j}(G)$ with $ij \notin E(H)$. Then it holds that*

$$\#\mathsf{PartitionedSub}(H \to G) = \#\mathsf{ColorfulSub}(H_u \to G').$$

Proof. It is clear that every color-preserving H-copy F in G is also a colorful H_u-copy in G' when considering F as an uncolored graph.

On the other hand, if F is a colorful H_u-copy in G', then $|E_{i,j}(F)| \leq 1$ holds for all $i, j \in [k]$. By construction of G', there are at most $|E(H)|$ pairs $\{i, j\} \subseteq [k]$ such that $E_{i,j}(G) \neq \emptyset$, namely those with $ij \in E(H)$. Therefore, $|E(F)| = |E(H)|$ is possible only if $|E_{i,j}(F)| = 1$ for all $ij \in E(H)$. But then F is in fact a color-preserving H-copy in G. □

Then the reduction (5.1) and Lemma 5.4 together imply the following lemma.

Lemma 5.5. *We have* $\#\mathsf{PartitionedSub}(\mathcal{H}) \leq^{lin}_{fpt} \#\mathsf{ColorfulSub}(\mathcal{H}) \leq^{lin}_{fpt} \#\mathsf{Sub}(\mathcal{H})$.

5.1.1. Dichotomy along treewidth

In this subsection, we show that $\#\mathsf{PartitionedSub}(\mathcal{H})$ is $\#\mathsf{W}[1]$-hard whenever $\mathcal{H}$ has unbounded treewidth. As already stated in the introduction, the proof uses well-known techniques, and it proceeds along the following two steps:

1. In Lemma 5.7, we prove that $\#\mathsf{PartitionedSub}(\mathcal{H}_{grid})$ is $\#\mathsf{W}[1]$-complete, where $\mathcal{H}_{grid}$ denotes the class of square grids.

2. In Lemma 5.8, we show that the problem $\#\mathsf{PartitionedSub}$ is minor-monotone. That is, whenever $\mathcal{H} \preceq \mathcal{H}'$ holds for classes $\mathcal{H}$ and $\mathcal{H}'$, we obtain

   $$\#\mathsf{PartitionedSub}(\mathcal{H}) \leq^{pars}_{fpt} \#\mathsf{PartitionedSub}(\mathcal{H}').$$

   Recall from Definition I that we write $\mathcal{H} \preceq \mathcal{H}'$ if, for every $H \in \mathcal{H}$, there exists some $H' \in \mathcal{H}'$ with $H \preceq H'$.

Then the Excluded Grid Theorem (Theorem 1.49 in our numbering) implies Theorem 5.6 from Lemmas 5.7 and 5.8, since it asserts that every graph class $\mathcal{H}$ of unbounded treewidth satisfies $\mathcal{H}_{grid} \preceq \mathcal{H}$. By Lemma 5.5, the same hardness result follows for the uncolored version $\#\mathsf{Sub}(\mathcal{H})$ via inclusion-exclusion.

Theorem 5.6 (restated from page 122)**.** *The problems* $\#\mathsf{PartitionedSub}(\mathcal{H})$ *and* $\#\mathsf{Sub}(\mathcal{H})$ *are* $\#\mathsf{W}[1]$*-complete whenever* $\mathcal{H}$ *is recursively enumerable and has unbounded treewidth.*

To prove this theorem, we need to prove Lemmas 5.7 and 5.8.

Lemma 5.7. *The problem* $\#\mathsf{PartitionedSub}(\mathcal{H}_{grid})$ *is* $\#\mathsf{W}[1]$*-complete under* $\leq_{fpt}^{pars}$*, where* $\mathcal{H}_{grid}$ *denotes the class of square grids.*

Proof. We reduce from the problem $\#\mathsf{GridTiling}$, whose $\#\mathsf{W}[1]$-completeness under parsimonious reductions was shown in Lemma 1.13 on page 32. Given an instance $\mathcal{T} : [k]^2 \to 2^{[n]^2}$ to this problem, for some numbers $k, n \in \mathbb{N}$, let H denote the $k \times k$ square grid on vertex set $[k]^2$. We construct a $[k]^2$-colored graph $G = G(\mathcal{T})$ such that

$$\#\mathsf{GridTiling}(\mathcal{T}) = \#\mathsf{PartitionedSub}(H \to G).$$

To this end, we proceed along the following steps. Please recall the notions of horizontally and vertically adjacent indices among $[k]^2$, as introduced in Section 1.1.

- For each $\kappa \in [k]^2$, and each $(r, s) \in \mathcal{T}(\kappa)$, create a vertex (κ, r, s) of color κ
- For all horizontally adjacent indices $\kappa, \kappa' \in [k]^2$, and for all numbers $r, s, s' \in [n]$, add an edge between (κ, r, s) and (κ', r, s') if both vertices exist in G. Call this edge *horizontal*.
- For all vertically adjacent indices $\kappa, \kappa' \in [k]^2$, and for all numbers $r, r', s \in [n]$, add an edge between (κ, r, s) and (κ', r', s) if both vertices exist in G. Call this edge *vertical*.

In the following, we verify that the consistent grid tilings of $\mathcal{T}$ stand in bijection with the color-preserving H-copies in G. Given such a color-preserving H-copy F, let v_κ denote the unique vertex of color κ in F. We claim that the following assignment $a_F : [k]^2 \to [n]^2$ is a consistent grid tiling of $\mathcal{T}$:

$$a_F : \ \kappa \mapsto (r, s) \quad \text{if } v_\kappa = (\kappa, r, s).$$

Note that a_F is a well-defined function. We verify in the following that a_F satisfies the three conditions specified in the definition of $\#\mathsf{GridTiling}$, see Problem 1.12 on page 32. The first condition is easily seen:

(C) The vertex (κ, r, s) is present in G only if $(r, s) \in \mathcal{T}(\kappa)$.

Furthermore, an edge between vertices of colors κ and κ' is present in G only if these indices are either vertically or horizontally adjacent. Using this, we can verify the remaining two conditions.

(H) If κ, κ' are horizontally adjacent, then an edge between vertices (κ, r, s) and (κ', r', s') is present in G (and thus in F) only if $r = r'$.

(V) If κ, κ' are vertically adjacent, then an edge between vertices (κ, r, s) and (κ', r', s') is present in G (and thus in F) only if $s = s'$.

Conversely, every consistent grid tiling a can be transformed to a color-preserving H-copy F_a in G by reversing this mapping, i.e., by including vertex $(\kappa, a(\kappa))$ for all $\kappa \in [k]^2$. □

It remains to prove that the complexity of the problem $\#\mathsf{PartitionedSub}(\mathcal{H})$ is indeed monotone under the relation $\preceq$ on graph classes. For later use, we note that under certain circumstances, a linear-blowup reduction can be achieved as well.

Lemma 5.8. *Let $\mathcal{H}$ and $\mathcal{H}'$ be recursively enumerable graph classes with $\mathcal{H} \preceq \mathcal{H}'$. Then*

$$\#\mathsf{PartitionedSub}(\mathcal{H}) \leq_{fpt}^{pars} \#\mathsf{PartitionedSub}(\mathcal{H}'). \tag{5.2}$$

If additionally, for every graph $H \in \mathcal{H}$ on k vertices, there is a graph $H' \in \mathcal{H}'$ with $H \preceq H'$ on $\mathcal{O}(k)$ vertices, then

$$\#\mathsf{PartitionedSub}(\mathcal{H}) \leq_{fpt}^{lin} \#\mathsf{PartitionedSub}(\mathcal{H}'). \tag{5.3}$$

Proof. Let $V(H) = [k]$ and let G be $[k]$-vertex-colored. Let $H' \in \mathcal{H}'$ with $H \preceq H'$ and let $V(H') = [k']$ for some $k' \in \mathbb{N}$. Given H, we can find H' in time $f(H)$, for a computable function f, by enumerating the graphs $H' \in \mathcal{H}'$ by non-decreasing number of vertices, and testing for each H' via brute force whether $H \preceq H'$ holds. Note that $k' = \mathcal{O}(k)$ if the additional condition of the lemma holds.

In the following, we construct a $[k']$-colored graph G' from the graphs G, H and H', and we claim that

$$\#\mathsf{PartitionedSub}(H \to G) = \#\mathsf{PartitionedSub}(H' \to G'). \tag{5.4}$$

This clearly implies (5.2). Note that this reduction increases the parameter from k to k', so it also implies (5.3) if the additional condition of the lemma holds.

Since $H \preceq H'$ holds, the set $V(H') = [k']$ admits a partition into branch sets $B_0, B_1, \ldots, B_k$ such that the following holds: For all $i \in [k] \setminus \{0\}$, the graph $H'[B_i]$ is connected, and deleting B_0 and contracting each B_i for $i \in [k]$ to a single vertex (which we denote by i) yields some supergraph of H on the vertex set $[k]$. Recall that $V_i(G)$ denotes the set of vertices in G with color i. Then we define G' as follows, shown also in Figure 5.1.

1. For all $i \in [k] \setminus \{0\}$ and $v \in V_i(G)$: Replace v by a copy of $H'[B_i]$, which we denote by L_v. Note that the vertices of $H'[B_i]$ are some subset of $[k']$.
2. For all $ij \in E(H)$ and all $uv \in E_{i,j}(G)$: Insert all edges between L_u and L_v.
3. For all $ij \notin E(H)$, all $u \in V_i(G)$ and $v \in V_j(G)$: Insert all edges between L_u and L_v.
4. Add a copy of $H'[B_0]$ to G'. Connect it to all other vertices of G'.

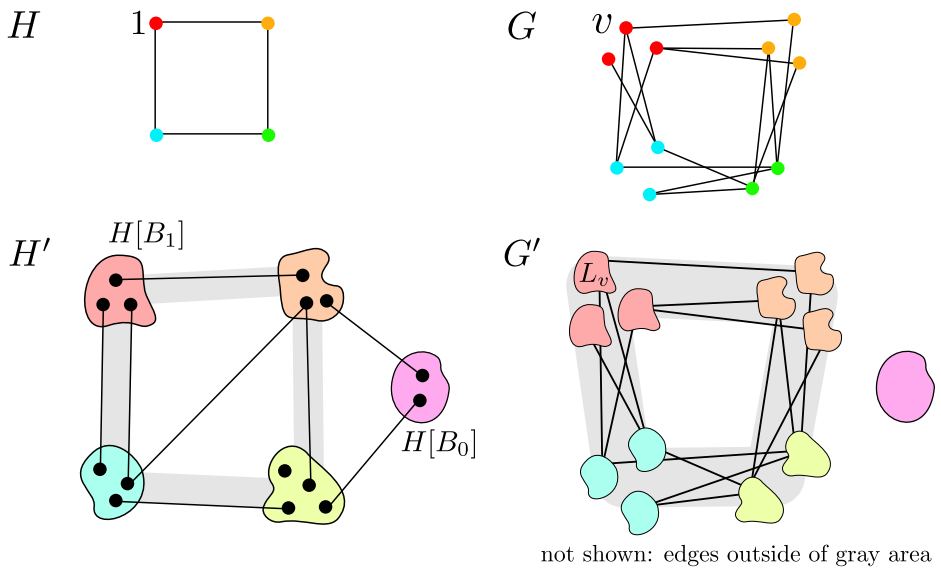


Figure 5.1.: The graphs $H \preceq H'$ are shown on the left. From the [4]-vertex-colored graph G, we construct the graph G'. To avoid clutter, the edges added in Step 3 of constructing G' are not shown, i.e., additionally to shown edges, G' has all possible edges not contained in the gray area.

In the following, we prove (5.4): Every color-preserving H-copy F in G can be extended to a unique color-preserving H'-copy F' in G' by following the very same graph transformation described in the steps above.

Conversely, every such H'-copy F' in G' corresponds to exactly one H-copy F in G: Since F' is color-isomorphic to H', we have that for every $i \in [k] \setminus \{0\}$, the B_i-colored vertices of F' induce a graph $F'_i \simeq H[B_i]$. Since $H'[B_i]$ is connected for all $i \in [k] \setminus \{0\}$, but the subgraphs L_u and L_v of G' are vertex-disjoint for different $u, v \in V_i(G)$, there is some unique vertex $v(i) \in V_i(G)$ with $F'_i = L_{v(i)}$. Applying this to all $i \in [k] \setminus \{0\}$ yields vertices $v(1), \ldots, v(k) \in V(G)$ such that $v(i) \in V_i(G)$ and, by construction of G', an edge between $v(i)$ and $v(j)$ is present in G if $ij \in E(H)$. In other words, the vertices $v(1), \ldots, v(k)$ are a color-preserving H-copy in G.

□

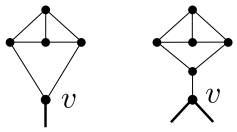
5.1.2. Cubic pattern graphs

In Section 5.2, we show #W[1]-hardness of #Match by reduction from #PartitionedSub on 3-regular patterns H. To this end, we establish in this subsection that #PartitionedSub($\mathcal{C}_\triangle$) is #W[1]-hard, where $\mathcal{C}_\triangle$ denotes the class of all 3-regular graphs. We will also derive a lower bound under #ETH for this problem by using a result in [Mar10].

To begin, we first show that any graph H on k vertices and ℓ edges is always contained as a minor in some 3-regular graph H' on $\mathcal{O}(k + \ell)$ vertices and edges.

Lemma 5.9. *If H is a graph on k vertices and ℓ edges, then there exists a 3-regular graph H' on $\mathcal{O}(k + \ell)$ vertices and edges with $H \preceq H'$.*

Proof. If H contains isolated vertices, add a dummy vertex that is adjacent to all isolated vertices so as to increase the minimum degree of H to 1. Then, for every vertex $v \in V(H)$ with $1 \leq \deg(v) < 3$, attach one of the following gadgets to v in order to increase $\deg(v)$ to 3, and note that this introduces no new vertices of degree other than 3.



For every $v \in V(H)$ with $\deg(v) > 3$, replace v by a *vertex cycle* of length $\deg(v)$ and attach the i-th edge in $I(v)$ to the i-th cycle vertex, for all $i \in [\deg(v)]$. The concrete ordering of $I(v)$ is irrelevant here.

The graph H' constructed this way is clearly 3-regular and has $\mathcal{O}(k + \ell)$ edges. Furthermore, it contains H as minor: To see this, delete the vertices introduced by gadgets, contract all vertex cycles to single vertices, and then delete the dummy vertex and its incident edges, if present. This procedure yields H from H'. □

Note that Lemma 5.9 implies $\mathcal{H} \preceq \mathcal{C}_\triangle$ for every graph class $\mathcal{H}$, and it follows by Lemma 5.8 that

$$\#\mathsf{PartitionedSub}(\mathcal{H}) \leq_{fpt}^{pars} \#\mathsf{PartitionedSub}(\mathcal{C}_\triangle). \tag{5.5}$$

In Lemma 1.11 on page 31, we have shown #W[1]-completeness of the problem #ColClique, which can be rephrased as $\#\mathsf{PartitionedSub}(\mathcal{K})$ for the class $\mathcal{K}$ of complete graphs. In fact, we have also shown #W[1]-completeness of the problem $\#\mathsf{PartitionedSub}(\mathcal{H}_{grid})$ in Lemma 5.7. Together with the reduction (5.5), we obtain the desired hardness result.

Lemma 5.10. *The problem* $\#\mathsf{PartitionedSub}(\mathcal{C}_\triangle)$ *is* #W[1]*-hard.*

By Example 1.21, the problem #ColClique cannot be solved in time $f(k)n^{o(k)}$ for any computable function f, unless #ETH fails. Here, n denotes the size of G and k denotes the size of the cliques to be counted. We would like to transfer this lower bound to $\#\mathsf{PartitionedSub}(\mathcal{C}_\triangle)$, and then ultimately to #Match. However, if H is a k-clique, then the graph H' constructed in Lemma 5.9 has $\mathcal{O}(k^2)$ vertices, so we cannot use the reduction shown in Lemma 5.8 to obtain the same tight lower bound for $\#\mathsf{PartitionedSub}(\mathcal{C}_\triangle)$.

To overcome this, we use a source problem different from #ColClique to prove tighter lower bounds for $\#\mathsf{PartitionedSub}(\mathcal{C}_\triangle)$, namely #PartitionedSub on the class of graphs with maximum degree D, where $D \in \mathbb{N}$ is a fixed universal constant. A lower bound for this problem was shown in [Mar10].

Theorem 5.11 ([Mar10])**.** *Assuming* #ETH*, there is a universal constant* $D \in \mathbb{N}$ *such that the following holds: Let* $\mathcal{C}_D$ *denote the class of graphs with maximum degree* D*. Then* $\#\mathsf{PartitionedSub}(\mathcal{C}_D)$ *cannot be solved in time* $f(k)n^{o(k/\log k)}$*, where* $k = |V(H)|$ *and* f *is any computable function.*[2]

[2] In [Mar10], the lower bound is shown for the decision version of $\#\mathsf{PartitionedSub}(\mathcal{C}_D)$, and assuming the decision version ETH of the exponential-time hypothesis. The techniques used in that paper however transfer easily to the counting version.

For any graph $H \in \mathcal{C}_D$ on k vertices, the construction in Lemma 5.9 yields a graph H' on $\mathcal{O}(Dk)$ vertices and edges, where D is a fixed constant. Hence, Lemma 5.8 yields

$$\#\mathsf{PartitionedSub}(\mathcal{C}_D) \leq^{lin}_{fpt} \#\mathsf{PartitionedSub}(\mathcal{C}_\triangle). \tag{5.6}$$

Then the reduction (5.6) and Theorem 5.11 readily imply the following lower bound.

Lemma 5.12. *Assuming* #ETH*, the problem* $\#\mathsf{PartitionedSub}(\mathcal{C}_\triangle)$ *admits no* $f(k)n^{o(k/\log k)}$ *time algorithm, where* $k = |V(H)|$ *and* f *is any computable function.*

5.2. Edge-colorful subgraphs

In this subsection, we consider *edge*-colorful subgraph problems, and we focus on the problem #EdgeColMatch of counting edge-colorful k-matchings.

Problem 5.13 (#EdgeColMatch)**.** Given a $[k]$-edge-colored graph G, for $k \in \mathbb{N}$, determine the number of edge-colorful matchings in G. The parameter is k.

For our proofs, we first require an edge-colored version of signature graphs and appropriately modified definitions of Holants, gates and signatures.

5.2.1. Edge-colorful Holant problems

In the following definition, we adapt Definition 2.2 on page 54 and Definition 2.9 on page 58 to the edge-colored setting. In this section, all appearing signature graphs will be *unweighted*. In fact, we take great care *not* to introduce edge-weights in our proofs, as they are often quite hard to eliminate in the setting of counting small subgraphs, and were a major obstacle in the step from [BC12] to [Cur13].

Definition 5.14. Let Ω be an edge-colored signature graph with color classes $E_1, \dots, E_\ell \subseteq E(\Omega)$ for $\ell \in \mathbb{N}$. An assignment $x \in \{0,1\}^{E(\Omega)}$ is *colorful* if $|x \cap E_i| = 1$ holds for all $i \in [\ell]$. We define $\mathrm{Holant}_{\mathrm{col}}(\Omega)$ as the restriction of $\mathrm{Holant}(\Omega)$ to colorful assignments:

$$\mathrm{Holant}_{\mathrm{col}}(\Omega) := \sum_{\substack{x \in \{0,1\}^{E(\Omega)} \\ x \text{ colorful}}} \mathrm{val}_\Omega(x).$$

An *edge-colored gate* is an edge-colored graph Γ with a set D of dangling edges. We call all colors appearing in $E(\Gamma) \setminus D$ *internal*. We then define $\mathrm{Sig}_{\mathrm{col}}(\Gamma) : \{0,1\}^D \to \mathbb{Q}$ by

$$\mathrm{Sig}_{\mathrm{col}}(\Gamma, x) := \sum_{\substack{xy \in \{0,1\}^{E(\Gamma)}: \\ y \in \{0,1\}^{E(\Gamma)\setminus D} \\ xy \text{ colorful}}} \mathrm{val}_\Gamma(xy).$$

We call Γ an *edge-colored matchgate* if it uses only the signature $\mathtt{HW}_{\leq 1}$. Please note that this signature is not $\mathtt{HW}_{=1}$, which is used for (uncolored) matchgates.

By definition, we have $\mathrm{Sig}_{\mathrm{col}}(\Gamma, x) = 0$ if some color occurs twice among the active edges of x. Let us also observe that the edge-colorful Holants of edge-colored graphs with $\mathtt{HW}_{\leq 1}$ at each vertex are equivalent to the problem #EdgeColMatch.

Fact 5.15. *Given an edge-colored graph G, let Ω denote the edge-colored signature graph obtained from G by placing the signature $\mathtt{HW}_{\leq 1}$ at each vertex $v \in V(G)$. Then the edge-colorful satisfying assignments to Ω are precisely the edge-colorful matchings of G.*

Given an edge-colored signature graph Ω and a vertex set $S \subseteq V(\Omega)$, we can contract S to a single vertex $w{\downarrow_S}$ with signature $\mathrm{Sig}_{\mathrm{col}}(\Omega_S)$ as in Definition 2.10. It can be verified that this operation preserves the value of $\mathrm{Holant}_{\mathrm{col}}(\Omega)$, similar to Lemma 2.11, as long as the internal colors of Ω_S appear nowhere else in Ω.

Lemma 5.16. *Let Ω be an edge-colored signature graph and let $S \subseteq V(\Omega)$ be such that no internal color in Ω_S appears in $\Omega - S$. Define $\Omega{\downarrow_S}$ by contracting S to a single vertex $w{\downarrow_S}$ with signature $\mathrm{Sig}_{\mathrm{col}}(\Omega_S)$. Then we have $\mathrm{Holant}_{\mathrm{col}}(\Omega) = \mathrm{Holant}_{\mathrm{col}}(\Omega{\downarrow_S})$.*

Proof. Adapt the calculations from Lemma 2.11. □

As in Section 2.1.3, we can reverse the contraction process to realize signatures in edge-colored signature graphs Ω by edge-colored matchgates Γ, provided that the internal colors in Γ are disjoint from the colors in Ω.

Remark 5.17. We note that *combined* signatures in edge-colored signature graphs Ω can be used to write $\mathrm{Holant}_{\mathrm{col}}(\Omega)$ as the linear combination of $\mathrm{Holant}_{\mathrm{col}}(\Omega_\theta)$ for derived edge-colored signature graphs Ω_θ, parallel to the Combined Signature Lemma on page 71. To this goal, note that this lemma also holds in the edge-colored setting: Simply invoke Remark 2.36 with the set X of edge-colorful assignments to Ω.

5.2.2. Hardness of counting k-matchings

We prove hardness of #EdgeColMatch by reduction from #PartitionedSub($\mathcal{C}_\triangle$): Given vertex-colored graphs G and H on colors $[k]$, where H is colorful and 3-regular, we wish to determine #PartitionedSub($H \to G$). In Lemmas 5.10 and 5.12, we have already observed #W[1]-completeness of this problem, as well as a $f(k)n^{o(k/\log k)}$ lower bound under #ETH.

We consider G and H to be fixed in the following. For simplicity, we assume $V(H) = [k]$, we write $E(H) = \{e_1, \ldots, e_\ell\}$ with $\ell = |E(H)|$, and we write $V(G) = \{v_1, \ldots, v_n\}$ with $n = |V(G)|$. By Remark 5.3, we may consider G to be colored with $[k]$ such that $E_{i,j}(G) = \emptyset$ if $ij \notin E(H)$.

In the first step of the reduction, we observe that #PartitionedSub can be expressed as a colorful Holant problem on the edge-colored signature graph $\Omega = \Omega_{H \to G}$ that is constructed from H and G as follows: For each vertex $i \in V(H)$, create a vertex w_i in Ω. For every edge e between vertices u, v (of colors i, j) in G, add an edge h between w_i and w_j to Ω, and "remember" the endpoints of the original edge e by a function π that maps $h \mapsto e$. We formalize this in the following definition.

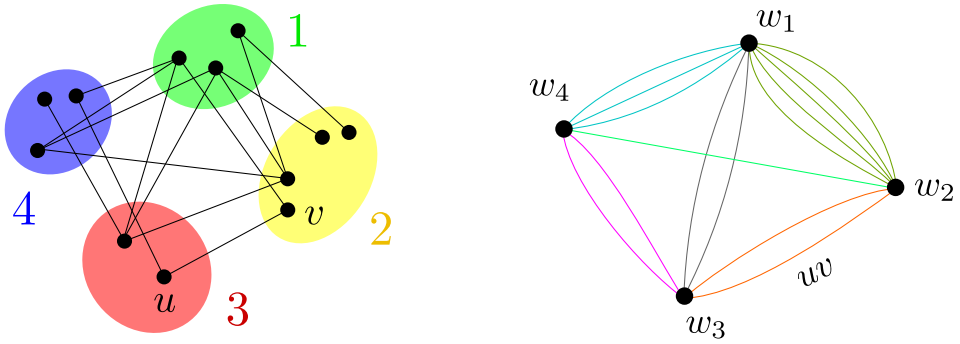


Figure 5.2.: A vertex-colored graph G is shown on the left, and the signature graph $\Omega_{H\to G}$ derived from G is shown on the right. For the edge $uv \in E(G)$, the corresponding edge h in $\Omega_{H\to G}$ is shown with its annotation $\pi(h) = uv$.

Definition 5.18. Let H and G be fixed as above. We define an $[\ell]$-edge-colored signature graph $\Omega = \Omega_{H\to G}$ on vertices $w_1, \dots, w_k$, together with a function $\pi : E(\Omega) \to E(G)$, and for each $i \in [k]$, a type function $\theta_i : \{0,1\}^{I(w_i)} \to \mathbb{N}$.

1. For $t \in [\ell]$, let $e_t = ij$ with $i, j \in [k]$ be the t-th edge in $E(H)$. For each edge $e \in E_{i,j}(G)$, add an edge $h = w_i w_j$ of color t and declare $\pi(h) = e$. Note that this step will usually produce parallel edges in Ω.

2. For assignments $x \in \{0,1\}^{E(\Omega)}$, extend the definition of π to

$$\pi(x) := \{\pi(e) \mid e \in x\}. \tag{5.7}$$

   and observe that $\pi(x) \subseteq E(G)$. For $i \in [k]$ and $x \in \{0,1\}^{I(w_i)}$, let $\theta_i(x)$ denote the number of distinct i-colored vertices in G that are incident with the edge set $\pi(x)$.

3. Attach the following Boolean vertex function $f_i : \{0,1\}^{I(w_i)} \to \{0,1\}$ to w_i, with

$$f_i : \quad x \mapsto \begin{cases} 1 & \theta_i(x) = 1, \\ 0 & \text{else.} \end{cases}$$

Note that $\Omega = \Omega_{H\to G}$ has $|E(G)|$ edges and a set of $|E_{i,j}(G)|$ parallel edges between w_i and w_j, which we call an *edge bundle*. The graph Ω is essentially obtained from G by contracting each set V_i for $i \in [k]$ to a single vertex w_i. For every $i \in [k]$, this vertex w_i is incident with 3 edge bundles, since H is 3-regular and we preprocessed G according to Remark 5.3. Furthermore, we observe the following fact:

Fact 5.19. *For each colorful assignment $x \in \{0,1\}^{E(\Omega)}$, we have $\theta_i(x) \in [3]$ for all $i \in [k]$. Furthermore, the satisfying colorful assignments $x \in \{0,1\}^{E(\Omega)}$ stand in bijection with the color-preserving H-copies F_x in G.*

Proof. For each colorful $x \in \{0,1\}^{E(\Omega)}$, the edge set $\pi(x)$ from (5.7) induces an edge-induced subgraph $F_x = G[\pi(x)]$. Since x is colorful, this subgraph F_x contains, for all

$ij \in E(H)$, precisely one edge in $E_{i,j}(G)$. Since each vertex in Ω has three incident edge-bundles, this graph F_x contains at most 3 distinct vertices of color i, for all $i \in [k]$, which implies $\theta_i(x) \in [3]$ and proves the first claim.

For the second claim, observe that F_x is a color-preserving H-copy in G iff $\theta_i(x) = 1$ holds for all $i \in [k]$. But this in turn is true if and only if $f_i(x) = 1$ at all $i \in [k]$. Conversely, every color-preserving H-copy F in G can be written as $F = F_x$ for a unique satisfying edge-colorful assignment x. □

In the following, we reduce $\text{Holant}_{\text{col}}(\Omega)$ to 3^k instances of #EdgeColMatch. More precisely, we show that every signature f_i appearing in Ω can be expressed as a linear combination from three constituents $(f_{i,\kappa})_{\kappa\in[3]}$, where $f_{i,\kappa}$ is the signature of an edge-colored matchgate $\Gamma_{i,\kappa}$ on 3 internal colors. Then we apply the Combined Signature Lemma for edge-colorful signature graphs.

For $i \in [k]$ and $\kappa \in [3]$, the matchgate $\Gamma_{i,\kappa}$ is obtained as the disjoint union of $t + \kappa - 1$ edge-colorful triangles, to which the dangling edges $I(w_i)$ attach. Here, $t \geq n$ is an arbitrary number.[3] The dangling edges are distributed among the first n triangles, while the remaining triangles function as "dummies" that will enable the desired linear combination. Note that, for fixed $i \in [k]$, the matchgates $\Gamma_{i,\kappa}$ for $\kappa \in [3]$ differ only in their numbers of dummy triangles.

Lemma 5.20. *Let $t \geq n$ be arbitrary. For $i \in [k]$ and $\kappa \in [3]$, let $\Gamma_{i,\kappa}$ denote the following edge-colored matchgate on internal colors $[3]$, with dangling edges $I(w_i)$:*

- *For all $u \in [t + \kappa - 1]$, define the following subgraph of $\Gamma_{i,\kappa}$, which we call the u-th triangle: Create vertices $a_{u,s}$ for $s \in [3]$ with signature $\mathtt{HW}_{\leq 1}$. For all $s \in [3]$, add an s-colored edge between $a_{u,s}$ and $a_{u,s+1}$, writing $3 + 1$ instead of 1 in the indices.*

- *Denote the three edge bundles in $I(w_i)$ by $B_{i,1}$, $B_{i,2}$ and $B_{i,3}$. For $s \in [3]$ and any edge $h \in B_{i,s}$ from the s-th bundle: If v_u for $u \in [n]$ is the (unique) i-colored endpoint of $\pi(h)$ in G, connect h as dangling edge to the triangle vertex $a_{u,s}$.*

Let $f_{i,\kappa} = \text{Sig}_{\text{col}}(\Gamma_{i,\kappa})$. Then there are fixed polynomials $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{Z}[t]$ (independent of the instance graphs G and H) such that

$$f_i = \frac{1}{12(t-2)^2(t-1)} \sum_{\kappa=1}^{3} \alpha_\kappa(t) \cdot f_{i,\kappa}. \tag{5.8}$$

The proof of this lemma is by mere calculation and is postponed to the end of this subsection. Using this lemma, the framework of combined signatures and edge-colorful Holant problems allows to derive the desired reduction.

Lemma 5.21. *We have #PartitionedSub($\mathcal{C}_\triangle$) $\leq^{lin}_{fpt}$ #EdgeColMatch.*

[3] To prove #W[1]-hardness of k-matchings, it suffices to choose $t = n$. We will later see other applications where we have to choose t more carefully.

Proof. Given graphs H and G as described at the beginning of this subsection, let $\Omega = \Omega(H \to G)$ denote the signature graph from Definition 5.18. By Fact 5.19, we have

$$\#\mathsf{PartitionedSub}(H \to G) = \mathrm{Holant}_{\mathrm{col}}(\Omega).$$

Let $t = |V(G)| + 3$, write $q(t) = 12^k(t-2)^{2k}(t-1)^k$ and observe that $q(t) \neq 0$. Let $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{Z}[t]$ be the polynomials from Lemma 5.20. By the Combined Signature Lemma and Remark 5.17 for its edge-colorful variant, we obtain

$$\mathrm{Holant}_{\mathrm{col}}(\Omega) = \frac{1}{q(t)} \cdot \sum_{\theta \in [3]^k} \left(\prod_{i=1}^{k} \alpha_{\theta(i)}(t) \right) \mathrm{Holant}_{\mathrm{col}}(\Omega_\theta), \tag{5.9}$$

where Ω_θ for $\theta \in [3]^k$ is obtained by replacing $f_i \leftarrow f_{i,\theta(i)}$ for all $i \in [k]$. By Lemma 5.20, we can realize $f_{i,\kappa}$ by the edge-colored matchgate $\Gamma_{i,\kappa}$, for all $i \in [k]$ and $\kappa \in [3]$.

Given $\theta \in [3]^k$, we construct an edge-colored signature graph Φ_θ from Ω_θ by inserting, for each $i \in [k]$, a copy of $\Gamma_{i,\theta(i)}$ on fresh internal colors at the vertex w_i in Ω_θ. This yields

$$\mathrm{Holant}_{\mathrm{col}}(\Omega_\theta) = \mathrm{Holant}_{\mathrm{col}}(\Phi_\theta),$$

and by Fact 5.15, we can compute $\mathrm{Holant}(\Phi_\theta)$ with an oracle call to $\#\mathsf{EdgeColMatch}$. Hence, we can compute $\#\mathsf{PartitionedSub}(H \to G)$ by (5.9) and 3^k oracle calls to $\#\mathsf{EdgeColMatch}$.

Every graph Φ_θ for $\theta \in [3]^k$ contains the edges of Ω, which have $|E(H)| = 3k/2$ distinct colors, and k matchgates, each on 3 internal colors. Therefore Ω features $\ell + 3k = 4.5k$ distinct edge-colors, which implies that our reduction indeed incurs linear blowup. □

This implies our main theorem for this subsection. Note that the restriction to bipartite graphs is missing in the following statement; we prove it as Theorem 5.24 in the following subsection by reduction from the not necessarily bipartite case.

Theorem 5.22 (restated from page 122)**.** *The problem* $\#\mathsf{EdgeColMatch}$ *and its uncolored variant* $\#\mathsf{Match}$ *are* $\#\mathsf{W}[1]$*-complete and admit no algorithm with running time* $f(k) \cdot n^{o(k/\log k)}$ *unless* $\#\mathsf{ETH}$ *fails.*

Proof. Recall the hardness results for $\#\mathsf{PartitionedSub}(\mathcal{C}_\triangle)$ from Lemmas 5.10 and 5.12. Together with Lemma 5.21, these imply the statement for $\#\mathsf{EdgeColMatch}$. By inclusion-exclusion via Lemma 1.34, we obtain the statement for the uncolored variant $\#\mathsf{Match}$. □

To complete the proof, it remains to perform the calculations needed for Lemma 5.20.

Proof of Lemma 5.20. Recall the types θ_i for $i \in [k]$ from Definition 5.18, and the falling factorial notation $(n)_k$ from Section 1.1. For fixed $\kappa \in [3]$, write $t' = t + \kappa - 1$. We first prove that, for $x \in \{0,1\}^{I(w_i)}$, we have

$$\mathrm{Sig}_{\mathrm{col}}(\Gamma_{i,\kappa}, x) = \begin{cases} (t'-1)_3 & \theta_i(x) = 1 \\ (t'-2)_3 + (t'-2)_2 & \theta_i(x) = 2 \\ (t'-3)_3 + 3(t'-3)_2 + 3(t'-3) + 1 & \theta_i(x) = 3 \end{cases} \tag{5.10}$$

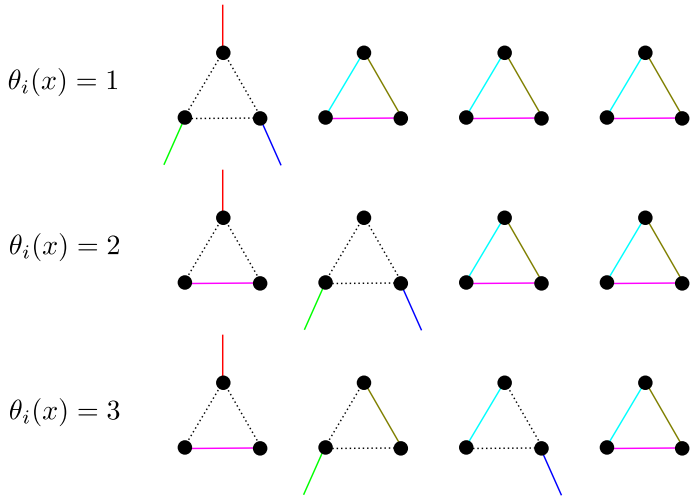


Figure 5.3.: The possible extensions of a colorful assignment to the dangling edges

By Fact 5.19, we have $\theta_i(x) \in [3]$ for all edge-colorful assignments to $I(w_i)$, so the case distinction in (5.10) is indeed sufficient to describe $\mathrm{Sig}_{\mathrm{col}}(\Gamma_{i,\kappa})$ entirely.

For $u \in [t']$, call the u-th triangle in $\Gamma_{i,\kappa}$ *hit by* x if an active edge of x is incident with it; otherwise call it *intact*. By definition of $\Gamma_{i,\kappa}$, the u-th triangle is hit if and only if $\pi(x)$ contains an edge with endpoint v_u.

To prove (5.10), we consider the edge-colorful satisfying assignments $xy \in \{0,1\}^{E(\Gamma_{i,\kappa})}$ that extend x, depending on the type $\theta_i(x)$: The active edges in each such assignment are an edge-colorful matching, and this matching may include some edges from triangles that are hit by x, while the remaining edges are from intact triangles. See Figure 5.3 for an overview of hit and intact triangles in each state of $\theta_i(x)$.

$\theta_i(x) = 1$: There are $t' - 1$ intact triangles, and no edges left in the unique triangle hit by x, as shown in Figure 5.3. We can find $(t'-1)_3$ edge-colorful matchings in the disjoint union of the $t' - 1$ intact triangles.

$\theta_i(x) = 2$: There are $t' - 2$ intact triangles. One of the two hit triangles has one edge left, the other has none left, as shown in Figure 5.3. Each matching may contain

- 3 edges from the intact triangles, yielding $(t'-2)_3$ matchings, or
- 2 such edges, and the one edge from the hit triangles, yielding $(t'-2)_2$ matchings.

$\theta_i(x) = 3$: There are $t' - 3$ intact triangles. Three triangles are hit, and each has one edge left, as shown in Figure 5.3. Each matching may contain

- 3 edges from the intact triangles, yielding $(t'-3)_3$ matchings, or
- 2 such edges, and 1 edge from the hit triangles, yielding $3(t'-3)_2$ matchings, or
- 1 such edge, and 2 edges from the hit triangles, yielding $3(t'-3)$ matchings, or
- 0 such edges, and 3 edges from the hit triangls, yielding 1 matching.

Summing over the disjoint numbers of extending matchings, depending on $\theta_i(x)$, we obtain (5.10). Now let $A \in (\mathbb{Z}[t])^{3\times 3}$ be the well-defined matrix obtained by declaring, for $r, \kappa \in [3]$,

$$A(r,\kappa) = (\mathrm{Sig}_{\mathrm{col}}(\Gamma_{i,\kappa}, x) \text{ on inputs } x \text{ with } \theta_i(x) = r).$$

That is, the signatures of $\Gamma_{i,1}, \Gamma_{i,2}, \Gamma_{i,3}$ (reduced to distinct entries) appear in A as columns of 3 entries each. It can be verified by hand that

$$A = \begin{pmatrix} t^3 - 6t^2 + 11t - 6 & t^3 - 3t^2 + 2t & t^3 - t \\ t^3 - 8t^2 + 21t - 18 & t^3 - 5t^2 + 8t - 4 & t^3 - 2t^2 + t \\ t^3 - 9t^2 + 29t - 32 & t^3 - 6t^2 + 14t - 11 & t^3 - 3t^2 + 5t - 2 \end{pmatrix}.$$

We can then furthermore verify that

$$\det(A) = 12 \cdot (t-2)^2 \cdot (t-1).$$

By solving $A \cdot (\beta_1, \beta_2, \beta_3)^T = (1,0,0)^T$ for indeterminates $\beta_1, \beta_2, \beta_3$ via Cramer's rule, we obtain solutions

$$\beta_i = \frac{\alpha_i}{\det(A)} \quad \text{for } i \in [3],$$

with polynomials $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{Z}[t]$ satisfying (5.8). This completes the proof. □

The proof of Theorem 5.22 is complete. To conclude this subsection, we observe that our reduction is quite indifferent to its target #EdgeColMatch: The computations in Lemma 5.20 could also be carried out for edge-colored gates other than matchgates, so a similar reduction could yield #W[1]-hardness of other edge-colorful Holant problems (and by Lemma 1.34, of their uncolored versions as well). We leave open a general hardness result for edge-colorful Holant problems, possibly even a dichotomy result.

Modular counting of k-matchings

The proof of Theorem 5.22 can also be adapted to prove hardness for modular counting of k-matchings, as we will observe in the following theorem. Recall the definitions of the classes $\oplus$W[1] and $\mathsf{Mod_t}$W[1] for $t \in \mathbb{N}$ from Section 1.2.2, and recall that W[1] is contained in $\oplus$W[1] under randomized fpt-reductions.

Theorem 5.23. *The following problem is $\oplus$W[1]-hard: Given $k \in \mathbb{N}$ and a graph (on $3k$ edge colors) count its (edge-colorful) $3k$-matchings modulo 2^{2k+1}.*

Proof. By Lemma 5.10, we know that #PartitionedSub($\mathcal{C}_\triangle$) is #W[1]-hard under parsimonious reductions, so the $\oplus$W[1]-hardness of its parity problem follows. In the following, let H and G be as in the proof of Theorem 5.22: The graph H has k vertices, and G is vertex-colored with $[k]$. We determine the parity of #PartitionedSub($H \to G$) by oracle calls to #EdgeColMatch modulo $m := 2^{3k+1}$ on graphs with $4.5k$ edge colors; this implies the statement.

In the remainder of the proof, we perform all arithmetic operations over $\mathbb{Z}_m$. Recall that (5.9) in the proof of Lemma 5.20 states that

$$q(t) \cdot \#\mathsf{PartitionedSub}(H \to G) = r,$$

with the quantities

$$\begin{aligned} q(t) &= 12^k (t-2)^{2k} (t-1)^k, \\ r &= \sum_{\theta \in [3]^k} \left(\prod_{i=1}^{k} \alpha_{\theta(i)}(t) \right) \mathrm{Holant}_{\mathrm{col}}(\Omega_\theta). \end{aligned}$$

We can clearly compute $r \bmod m$ using 3^k oracle queries to $\#\mathsf{EdgeColMatch} \bmod m$ and arithmetic in $\mathbb{Z}_m$. To proceed, we ensure that $q(t)$ is divisible by 2^{3k}, but not by 2^{3k+1}.

Recall that the number of triangles $t \in \mathbb{N}$ in the definition of $\Gamma_{i,\kappa}$ can be chosen arbitrarily in Lemma 5.20, provided that $t \geq |V(G)|$. We choose t minimal such that t is odd and $t-1$ is not divisible by 4, which can clearly be achieved with $t \leq |V(G)| + 4$. Then $(t-1)^k$ is divisible by 2^k, but not by 2^{k+1}, so there exists an odd number $s \in \mathbb{N}$ with

$$q(t) = 4^k \underbrace{3^k (t-2)^{2k}}_{\text{odd}} (t-1)^k = 2^{3k} s.$$

But then the value r we computed modulo m, namely

$$r = 2^{3k} s \cdot \#\mathsf{PartitionedSub}(H \to G).$$

is divisble by $m = 2^{3k+1}$ if and only if $\#\mathsf{PartitionedSub}(H \to G)$ is even. Hence, we can solve the $\oplus\mathsf{W}[1]$-complete problem $\oplus\mathsf{PartitionedSub}(\mathcal{C}_\triangle)$ by 3^k oracle calls to $\#\mathsf{EdgeColMatch}$ modulo m and modular arithmetic in $\mathbb{Z}_m$.

The result for uncolored matchings follows by reduction from $\#\mathsf{EdgeColMatch}$ via Lemma 1.34: The inclusion-exclusion formula is clearly also true in $\mathbb{Z}_m$. □

Theorem 5.23 can also be shown for other moduli. For instance, counting edge-colorful or uncolored matchings modulo primes $p > 3$ is complete for $\mathsf{Mod_pW}[1]$ by the same argument: One only needs to observe that, after suitable choice of t, the factor $q(t)$ is not divisible by p and hence has an inverse in $\mathbb{Z}_p$. Note that $\oplus\mathsf{Match}$ admits a simple polynomial-time algorithm. We leave the case $p = 3$ open: Counting k-matchings modulo 3 might turn out to be $\mathsf{Mod_3W}[1]$-complete by using a different construction of matchgates $\Gamma_{i,\kappa}$.

5.2.3. Bipartite k-matchings

In the following, we extend Theorem 5.23 to the bipartite case, as this will be required in Chapter 6. The proof again involves a simple application of combined matchgates.

Theorem 5.24 (listed as Theorem 5.22 on page 122)**.** *The problem* $\#\mathsf{EdgeColMatch}$ *and its uncolored variant* $\#\mathsf{Match}$ *are* $\#\mathsf{W}[1]$*-complete and admit no algorithm with running time* $f(k) \cdot n^{o(k/\log k)}$ *unless* $\#\mathsf{ETH}$ *fails. Here, we may even assume* G *to be bipartite.*

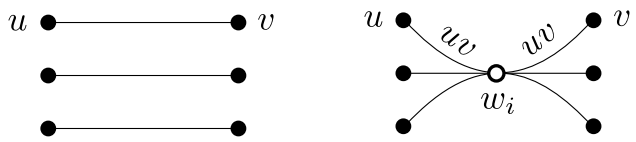


Figure 5.4.: Edges of color i are shown on the left, and their corresponding edges in the graph Ω_{bip} are shown on the right.

Let G be a (not necessarily bipartite) simple edge-colored graph with colors $[k]$, whose edge-colorful matchings we wish to count. We transform G to an edge-colored bipartite signature graph $\Omega_{bip} = \Omega_{bip}(G)$ such that $\mathrm{Holant}_{\mathrm{col}}(\Omega_{bip})$ counts the edge-colorful matchings in G. The graph Ω_{bip} is constructed in a way that resembles Definition 5.18.

Lemma 5.25. *Given an edge-colored graph G on colors $[k]$, let $\Omega_{bip} = \Omega_{bip}(G)$ denote the signature graph on colors $[k] \times [2]$ obtained as follows from G: Assign $\mathtt{HW}_{\leq 1}$ to all vertices, then perform the following steps for each $i \in [k]$.*

1. *Add a vertex w_i to Ω_{bip}.*
2. *For each edge $e \in E(G)$ of color i, with $e = uv$, delete e and insert an edge uw_i of color $(i, 1)$, and an edge w_iv of color $(i, 2)$. Annotate the two added edges with $\pi(uw_i) = \pi(w_iv) = e$. See also Figure 5.4.*
3. *Note that every colorful assignment $x \in \{0,1\}^{I(w_i)}$ has precisely two active edges, call them $e_1(x)$ and $e_2(x)$. Assign to w_i the signature f_i which maps $x \in \{0,1\}^{I(w_i)}$ to*

$$f_i(x) = \begin{cases} 1 & \pi(e_1(x)) = \pi(e_2(x)), \\ 0 & \textit{else.} \end{cases}$$

Then Ω_{bip} admits a bipartition with $\{w_i \mid i \in [k]\}$ on one side and the original vertices from G on the other side. Furthermore, it holds that

$$\mathrm{Holant}_{\mathrm{col}}(\Omega_{bip}) = \#\mathsf{EdgeColMatch}(G).$$

Proof. It is clear that Ω_{bip} is bipartite, since every edge of G appears in Ω_{bip} as a subdivided edge with a subdivision vertex w_i for some $i \in [k]$.

The edge-colorful satisfying assignments $x \in \{0,1\}^{E(\Omega_{bip})}$ stand in bijection with the edge-colorful matchings of G: In any such x, the vertex w_i for $i \in [k]$ is incident with two active edges e_i and e_i' that have the same annotation $h_i = \pi(e_i) = \pi(e_i')$, for some $h_i \in E(G)$. We can hence contract e_i and e_i' to one edge h_i. The resulting edge set is an edge-colorful matching in G due to the signature $\mathtt{HW}_{\leq 1}$ at non-subdivision vertices. Likewise, every edge-colorful matching in G can be extended to a unique satisfying assignment $x \in \{0,1\}^{E(\Omega_{bip})}$. □

As in Section 5.2.2, we now realize the signatures in $\Omega_{bip} = \Omega_{bip}(G)$ as combined signatures whose constituents admit edge-colored matchgates, and which result in bipartite

signature graphs when inserted into Ω_{bip}. Let $m = |E(G)|$ and consider the edges in G to be ordered in some arbitrary fixed way.

Lemma 5.26. *For $i \in [k]$, let $\Gamma_{i,1}$ denote the matchgate on dangling edges $I(w_i)$ that consists of $2m$ vertices and is defined as follows:*

1. *Create independent sets $a_1, \dots, a_m$ and $b_1, \dots, b_m$, which we call "external" vertices.*
2. *For all $j \in [m]$ and all edges $e, e' \in E(\Omega_{bip})$ of colors $(i,1)$ and $(i,2)$ with $\pi(e) = \pi(e')$: If $\pi(e)$ is the j-th edge in the ordering of $E(G)$, for $j \in \mathbb{N}$, then attach e as dangling edge to a_j and e' as dangling edge to b_j.*

Let $\Gamma_{i,2}$ be defined likewise, with the following addition: For all $j \in [m]$, add an extra vertex c_j, an edge $a_j c_j$ of color $(i,3)$ and an edge $c_j b_j$ of color $(i,4)$. Then it holds that

$$f_i = (m^2 - 3m + 3) \cdot \mathrm{Sig}_{\mathrm{col}}(\Gamma_{i,1}) - \mathrm{Sig}_{\mathrm{col}}(\Gamma_{i,2}). \tag{5.11}$$

Proof. Concerning $\Gamma_{i,1}$, we observe that, for all edge-colorful $x \in \{0,1\}^{I(w_i)}$, we have

$$\mathrm{Sig}_{\mathrm{col}}(\Gamma_{i,1}, x) = 1.$$

This is because x trivially is the only satisfying assignment that extends x, since there are no edges other than $I(w_i)$ in $\Gamma_{i,1}$. Concerning $\Gamma_{i,2}$, let $x \in \{0,1\}^{I(w_i)}$ be a colorful assignment with active edges e_1, e_2. We show

$$\mathrm{Sig}_{\mathrm{col}}(\Gamma_{i,2}, x) = \begin{cases} m^2 - 3m + 2 & \pi(e_1) = \pi(e_2), \\ m^2 - 3m + 3 & \text{else}, \end{cases} \tag{5.12}$$

which implies (5.11). We prove the two cases in (5.12) separately:

" $=$ " : There are $m-1$ paths in $\Gamma_{i,2}$ not hit by x, each on the same two colors. This gives $(m-1)_2 = m^2 - 3m + 2$ matchings.

" $\neq$ " : There are $m-2$ paths not hit by a. Each matching may contain

- 2 edges from the intact paths, yielding $(m-2)_2$ matchings, or
- 1 such edge, yielding $2(m-2)$ matchings, or
- 0 such edges, yielding 1 matching.

By summing over the disjoint possible choices, we obtain (5.12). □

This allows to conclude the hardness result for #BipMatch.

Proof of Theorem 5.24. Recall that Ω_{bip} is bipartite, with the subdivision vertices $\{w_i\}_{i \in [k]}$ on one side, say the left side. For $i \in [k]$, we can insert $\Gamma_{i,\kappa}$ for $\kappa \in [2]$ at the vertex w_i in Ω_{bip}. Only the external vertices of the matchgate $\Gamma_{i,\kappa}$ connect to vertices outside of $\Gamma_{i,\kappa}$, and by construction of $\Gamma_{i,\kappa}$, we may put its external vertices on the left side and its remaining vertices on the right side of the bipartition.

Hence, invoking the Combined Signature Lemma as in the proof of Theorem 5.21, we can write $\text{Holant}_{\text{col}}(\Omega_{bip})$ as a linear combination of 2^k numbers of edge-colorful matchings in bipartite graphs. This proves Theorem 5.24. □

Remark 5.27. Since no divisions were performed in the linear combination (5.11), we may also assume in Theorem 5.23 that the input graph is bipartite.

5.2.4. Paths and cycles

With a simple linear-blowup reduction, we can transfer the lower bound of Theorem 5.24 for #BipMatch to counting directed k-cycles.

Theorem 5.28. *The problem* #DirCycle *of counting directed k-cycles in a directed graph G is* #W[1]*-hard and admits no $f(k)n^{o(k/\log k)}$ time algorithm, unless* #ETH *fails.*

Proof. We count k-matchings in a graph G with bipartition $V(G) = L \dot\cup R$ by counting directed $2k$-cycles in the directed graph G' obtained from G by directing each edge from L to R and adding each directed edge from R to L. No vertices were added in the construction of G', and this implies the lower bound under #ETH.

Since every directed $2k$-cycle C of G' alternates between L and R, it contains k edges from L to R. Since C is simple, it visits no vertex twice, and hence the edges from L to R induce a k-matching of G. Conversely, every k-matching M of G can be extended to a $2k$-cycle in G' by firstly orienting the edges of M from L to R and then adding edges from R to L. The added edges fix a permutation of M up to cyclic equivalence, hence each matching M corresponds to $(k-1)!$ cycles of length $2k$ in G'.

Thus, if M_k denotes the number of k-matchings in G and C_{2k} denotes the number of $2k$-cycles in G', then $C_{2k} = (k-1)! \cdot M_k$, which proves the claim. □

We show how to use this to prove hardness for counting undirected cycles and paths.

Theorem 5.29 (restated from page 123)**.** *The four problems* #DirCycle*,* #UndirCycle*,* #UndirPath*,* #DirPath *of counting directed/undirected paths/cycles of length k are all* #W[1]*-hard and admit no algorithms with running time $f(k)n^{o(k/\log k)}$ unless* #ETH *fails.*

Proof. We reduce each problem to its right neighbor in the order they appear in the statement. The last two reductions are shown in [FG04] and can be observed to preserve the parameter. We thus only show the reduction from #DirCycle to #UndirCycle.[4]

Let D be a directed graph whose number of directed k-cycles we wish to determine. We transform D to an undirected graph G on edge-colors $\{0, 1, \ldots, k\}$ that we will later remove by an application of the inclusion-exclusion principle.

1. Replace each vertex $v \in V(D)$ by vertices v^{in} and v^{out}, and replace each edge $uv \in E(D)$ by the undirected edge $u^{out}v^{in}$ in G. Consider these edges to be colored with 0. See Figure 5.5 for an example.

[4]Such a reduction was also shown in [FG04], but it incurs a polynomial parameter blowup.

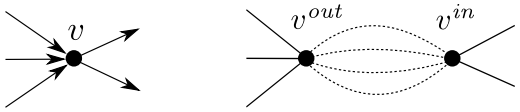


Figure 5.5.: The transformation from D to G in the proof of Theorem 5.29. Solid edges are assigned color 0, while dashed edges have colors $1, \dots, k$.

2. For each vertex $v \in V(D)$, add k parallel edges, called *internal edges at* v, between vertices v^{in} and v^{out}. Assign color $i \in [k]$ to i-th parallel edge. We will later show how to obtain simple graphs that feature no parallel edges.

Let $\mathcal{C}$ denote the set of directed k-cycles in D, and let $\mathcal{B}$ denote the set of undirected $2k$-cycles in G that choose at least one edge of each color $[k]$.

Claim 5.30. It holds that $|\mathcal{B}| = k! \cdot |\mathcal{C}|$.

Proof. For each cycle $C \in \mathcal{C}$, we define a subset $\mathcal{B}_C \subseteq \mathcal{B}$, where $B \in \mathcal{B}_C$ if and only if B contains all edges $u^{out}v^{in}$ for $uv \in E(C)$ and, at each vertex $w \in V(C)$, some internal edge $w^{out}w^{in}$. It is clear that $\mathcal{B}_C \cap \mathcal{B}_{C'} = \emptyset$ if $C \neq C'$.

For all $C \in \mathcal{C}$, we have $|\mathcal{B}_C| = k!$ since there are $k!$ ways of choosing internal edges for each $v \in V(C)$: Exactly k internal edges are present in B, and each $i \in [k]$ is the color of exactly one internal edge. Therefore, we have $|\mathcal{B}| \geq k! \cdot |\mathcal{C}|$.

We now show $\mathcal{B} \subseteq \bigcup_{C \in \mathcal{C}} \mathcal{B}_C$, which implies $|\mathcal{B}| \leq k! \cdot |\mathcal{C}|$ and proves the claim. Since each color $i \in [k]$ is present in $B \in \mathcal{B}$, the cycle B passes through $s \geq k$ internal edges. But since internal edges at distinct vertices are vertex-disjoint and no simple cycle of length > 2 can visit two internal edges of the same vertex, at least s additional 0-colored edges are required for B to be a cycle. Since B has length $2k$, this is possible only when $s = k$. Let $v_1, \dots, v_k \in V(D)$ be the vertices whose internal edges B visits, then B contains both v_i^{in} and v_i^{out} for all $i \in [k]$. Thus, if we orient edges from out-vertices to in-vertices and contract each internal edge, then we obtain a directed k-cycle of D, i.e., some element of $\mathcal{C}$, and it holds that $B \in \mathcal{B}_C$. □

This allows to compute $|\mathcal{C}|$ from $|\mathcal{B}|$. To determine $|\mathcal{B}|$, we count the $2k$-cycles in certain uncolored subgraphs of G by using the inclusion-exclusion principle: For each $S \subseteq [k]$, let D_S denote the subgraph of D that contains only edges of color S. Every $2k$-cycle C in D is contained in $\mathcal{B}$ if and only if C is contained in $D_{[k]}$, but in none of D_S for $S \subsetneq [k]$. Together with oracle calls for counting undirected uncolored $2k$-cycles, this allows to apply inclusion-exclusion (Lemma 1.33). In each oracle call, the parameter is $2k$.

In the above reduction, the oracle for uncolored $2k$-cycles is called on graphs with parallel edges, but we can easily reduce these to graphs without parallel edges: Let F' denote the graph obtained from F by subdividing each edge. Then F' contains no parallel edges, and for $t > 2$, the $2t$-cycles of F' stand in bijection with the t-cycles of G. □

6. Uncolored subgraphs

In this chapter, we consider the *uncolored* subgraph counting problem $\#\mathsf{Sub}(\mathcal{H})$ for fixed classes $\mathcal{H}$. In Section 6.1, we present a simple algorithm for solving $\#\mathsf{Sub}(H \to G)$ in polynomial time if H has a vertex-cover of constant size. This gives the polynomial-time solvable cases in the dichotomy of Theorem 6.11. Note that we have already seen $\#\mathsf{W}[1]$-hardness for classes $\mathcal{H}$ of unbounded treewidth in Section 5.1. For classes $\mathcal{H}$ of bounded treewidth and unbounded vertex-cover number, we prove $\#\mathsf{W}[1]$-hardness in Section 6.2, and thus finish the proof of Theorem 6.11.

6.1. Bounded vertex-cover number

We first present a simple algorithm for determining $\#\mathsf{Sub}(H \to G)$ in time polynomial in $|V(H)|$ and $|V(G)|$ when H admits a vertex-cover of size $\mathcal{O}(1)$. As already stated in the introduction, more efficient algorithms are known in the literature, and we include the following theorem only for its simple proof, and for sake of completeness.

In fact, we give an algorithm for the more general problem of counting *embeddings* from H to G. These are injective functions $f : V(H) \to V(G)$ such that $uv \in E(H)$ implies $f(u)f(v) \in E(G)$, and we write $\#\mathsf{Emb}(H \to G)$ for the number of such embeddings. It is clear that

$$\#\mathsf{Sub}(H \to G) = \frac{\#\mathsf{Emb}(H \to G)}{\#\mathsf{Emb}(H \to H)}, \tag{6.1}$$

because every H-copy F in G can be uniformly extended to an embedding of H in G by applying an automorphism of H to F.

The algorithm we present in the following is based on a similar idea as an algorithm for PerfMatch that we have considered in Section 4.3: After having guessed an image S for the vertex-cover of H in G, we classify the vertices of G into *types* according to the subset of S adjacent to them. Then we show how to exploit that any vertex of G can only have one of at most $2^{|S|}$ types.

Theorem 6.1 (restated from page 120)**.** *Let H be a k-vertex graph with a vertex-cover of size τ, and let G be a graph on n vertices, for $n \in \mathbb{N}$. Then we can determine the number of H-copies in G in time $k^{2^{\mathcal{O}(\tau)}} n^{\tau+\mathcal{O}(1)}$. Note that, if $\tau = \mathcal{O}(1)$, then this running time is polynomial.*

Proof. Let $C = \{c_1, \ldots, c_\tau\}$ be a vertex-cover of H. For every $X \subseteq C$, let R^H_X be the set of vertices in $V(H) \setminus C$ with $N_H(v) = X$. Note that $\sum_{X \subseteq C} |R^H_X| = k - \tau$.

For each tuple $\mathbf{s} \in V(G)^\tau$, with $\mathbf{s} = (s_1, \ldots, s_\tau)$, let

$$\mathcal{A}_{\mathbf{s}} = \{f \in \mathsf{Emb}(H \to G) \mid \forall i \in [\tau] : f(c_i) = s_i\}.$$

That is, the set $\mathcal{A}_{\mathbf{s}}$ contains all subgraph embeddings that map C as prescribed by $\mathbf{s}$. Note that, if a vertex $v \in V(G)$ appears more than once in $\mathbf{s}$, then clearly $\mathcal{A}_{\mathbf{s}} = \emptyset$. Since the sets $\mathcal{A}_{\mathbf{s}}$ clearly partition $\#\mathsf{Emb}(H \to G)$, it suffices to compute $\#\mathcal{A}_{\mathbf{s}}$ for each of the n^τ tuples $\mathbf{s} \in V(G)^\tau$, because we have

$$\#\mathsf{Emb}(H \to G) = \sum_{\mathbf{s} \in V(G)^\tau} \#\mathcal{A}_{\mathbf{s}}. \tag{6.2}$$

We show how to compute $\#\mathcal{A}_{\mathbf{s}}$ in time $k^{2^{\mathcal{O}(\tau)}} n^{\mathcal{O}(1)}$ for arbitrary *fixed* $\mathbf{s}$, which implies the claimed total runtime. Since $V(H) \setminus C$ is an independent set, we can safely delete all edges in G that are not incident with any vertex in $\mathbf{s}$. The resulting graph G' has the vertex cover $S = \{s_1, \ldots, s_\tau\}$. For every $Y \subseteq S$, let

$$R_Y^G = \{v \in V(G') \setminus S \mid N_{G'}(v) = Y\}.$$

Construct a bipartite directed graph I on $2^\tau + 2^\tau$ vertices, with a left vertex ℓ_Y for each $Y \subseteq S$ and a right vertex r_X for each $X \subseteq C$. Identify S with C by declaring $c_i \simeq s_i$ for $i \in [\tau]$. For X, Y with $X \subseteq Y$, add the directed edge (ℓ_Y, r_X) to I. Intuitively, this edge signifies that any vertex of R_Y^G may be the image of a vertex in R_X^H in an embedding.

We consider I as a flow network: For $Y \subseteq S$, consider $|R_Y^G|$ as the supply of ℓ_Y, and for $X \subseteq C$, consider $|R_X^H|$ as the demand of r_X. Let $\mathcal{F}$ denote the set of all feasible integral flows $F : E(I) \to \mathbb{N}$ in I that exactly satisfy the demands. As the total demand is $k - \tau \leq k$ and I has $t = 2^{\mathcal{O}(\tau)}$ edges, we have $|\mathcal{F}| \leq k^{2^{\mathcal{O}(\tau)}}$ and can thus enumerate $\mathcal{F}$ by brute force.

Let $F \in \mathcal{F}$ be such a flow. If F has value m on the edge (ℓ_Y, r_X), then this corresponds to mapping m vertices of R_X^H to R_Y^G, and the number of such mappings is given by the falling factorial expression $(|R_Y^G|)_m$. For $Y \subseteq S$, let $d_Y(F)$ denote the total outgoing flow at ℓ_Y. Then it can be verified that

$$\#\mathcal{A}_{\mathbf{s}} = \sum_{F \in \mathcal{F}} \prod_{Y \subseteq S} (|R_Y^G|)_{d_Y(F)}.$$

Hence $\#\mathcal{A}_{\mathbf{s}}$ can be computed in time $k^{2^{\mathcal{O}(\tau)}} n^{\mathcal{O}(1)}$ for fixed tuples $\mathbf{s} \in V(G)^\tau$, so it follows by (6.2) that $\#\mathsf{Emb}(H \to G)$ can be computed in the claimed time bound. The statement for $\#\mathsf{Sub}(H \to G)$ then follows by (6.1). □

6.2. Hardness for unbounded vertex-cover number

In this section, we prove $\#\mathsf{W}[1]$-hardness of $\#\mathsf{Sub}(\mathcal{H})$ for classes of unbounded vertex-cover number. To this end, we use a machinery of so-called *k-matching gadgets*, which are graphs $H \in \mathcal{H}$ together with a partition of $V(H)$ into an induced matching M and some remainder C. These gadgets satisfy certain technical properties which will be used in Lemma 6.5, the main technical part of this section. It states that, if $\mathcal{H}$ is a class of graphs that contains

k-matching gadgets for all $k \in \mathbb{N}$, then

$$\#\mathsf{BipMatch} \leq_{fpt}^{T} \#\mathsf{Sub}(\mathcal{H}).$$

In the remainder of this section, we define k-matching gadgets formally and then prove Lemma 6.5. The existence of k-matching gadgets in graph classes $\mathcal{H}$ is shown in Appendix B by Dániel Marx.

Definition 6.2. Let H be a graph and let $C \subseteq V(H)$. Then we denote by $\partial_H(C)$ the set of vertices in C that have a neighbor in $V(H) \setminus C$. If f is an isomorphism from $H[C]$ to $H[C']$ for some $C, C' \subseteq V(H)$ such that $f(\partial_H(C)) = \partial_H(C')$, then we say that f is *boundary preserving*.

The following definition formulates the properties of gadgets we need in Lemma 6.5.

Definition 6.3. Let H be a graph, let M be an induced k-matching in H, and let $C := V(H) \setminus V(M)$. We say that (H, M) is a *k-matching gadget* if the following holds: Whenever an isomorphism f from $H[C]$ to $H[C']$ for some $C' \subseteq V(H)$ satisfies the conditions

(C1) $H \setminus C'$ has no isolated vertex,

(C2) $H \setminus C'$ is bipartite, and

(C3) f is boundary preserving,

then $H \setminus C'$ is a k-matching, i.e., $H \setminus C'$ is isomorphic to the graph on $2k$ vertices that contains k vertex-disjoint edges.

Recall that, by Theorem 6.1, the problem $\#\mathsf{Sub}(\mathcal{H})$ admits a polynomial-time algorithm if $\mathcal{H}$ has bounded vertex-cover number. If $\mathcal{H}$ has unbounded treewidth, then $\#\mathsf{Sub}(\mathcal{H})$ is $\#\mathsf{W}[1]$-complete by Theorem 5.6, since even $\#\mathsf{PartitionedSub}(\mathcal{H})$ is $\#\mathsf{W}[1]$-complete. Using a rather extensive graph-theoretical analysis, it was shown by Dániel Marx that every graph class $\mathcal{H}$ that is not covered by Theorem 5.6 or Theorem 6.1 admits k-matching gadgets. We include this proof, which also appeared in [CM14], in Appendix B.

Theorem 6.4 (Appendix B)**.** *Let $\mathcal{H}$ be a graph class of unbounded vertex-cover number and bounded treewidth. Then, for all $k \in \mathbb{N}$, there exists a graph $H \in \mathcal{H}$ and a subset $M \subseteq V(H)$ such that (H, M) is a k-matching gadget.*

The hardness result for $\#\mathsf{Sub}(\mathcal{H})$ when $\mathcal{H}$ has unbounded vertex-cover number and bounded treewidth follows by reducing $\#\mathsf{BipMatch}$ to $\#\mathsf{Sub}(\mathcal{H})$ for classes $\mathcal{H}$ that contain k-matching gadgets of all sizes. This reduction is shown in the following lemma.

Lemma 6.5. *Let G be a graph and let (H, M) be a k-matching gadget. Then we can compute the number of k-matchings in G from $2^{\mathcal{O}(|V(H)|^2)}$ oracle queries of the form $\#\mathsf{Sub}(H \to G')$, where G' is a graph constructed from G.*

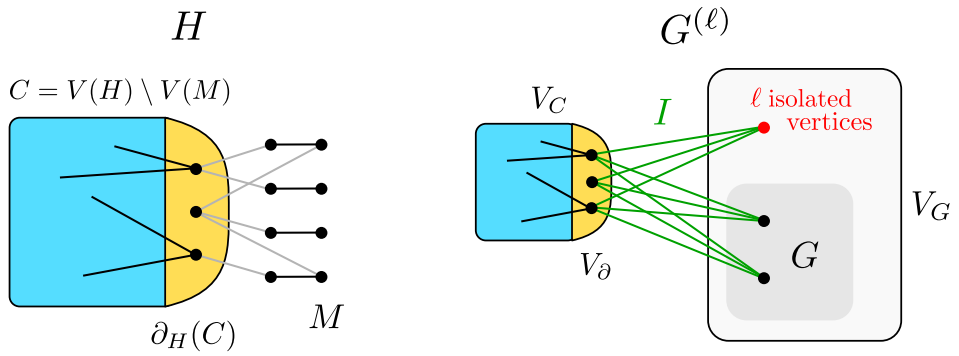


Figure 6.1.: A k-matching gadget (H, C). The graph $G^{(\ell)}$ constructed from (H, C) and G.

Proof. For $\ell \in \mathbb{N}$, let $G^{(\ell)}$ be defined as the following supergraph of G, shown also in Figure 6.1. For vertex sets X, we abbreviate $F[X \cap V(F)]$ to $F[X]$.

- Starting from the empty graph, place a copy of G into $G^{(\ell)}$ and add ℓ isolated vertices, shown red in Figure 6.1. Let V_G denote the set of vertices that contains the isolated vertices *and* the vertices of the G-copy, and let E_G denote the edges present in $G^{(\ell)}$.
- Add a copy of $H[C]$ on a vertex set V_C disjoint from V_G and an edge-set E_C disjoint from E_G. Denote the copy of $\partial_H(C)$ by $V_\partial \subseteq V_C$, shown orange.
- Add *all* edges between V_∂ and V_G, call this edge set I, shown green.

Using the principle of inclusion and exclusion, we will count the H-copies in $G^{(\ell)}$ that contain all of V_C and E_C, and where every vertex of V_∂ has an edge to V_G. If such a copy F of H in $G^{(\ell)}$ additionally satisfies that $F[V_G]$ contains no isolated vertices, then the properties of k-matching gadgets from Definition 6.3 imply that $F[V_G]$ actually is a k-matching. Therefore, the number of such copies can be put into relationship with the number of k-matchings in G.

To calculate the number of H-copies F such that $F[V_G]$ has no isolated vertices, we count the number of H-copies for different values of ℓ (which correspond to different numbers of introduced isolated vertices) and use an interpolation argument to determine the contribution of those copies where $F[V_G]$ has no isolated vertices.

We first aim at determining the size of

$$T^{(\ell)} = \{F \in \mathrm{Sub}(H \to G^{(\ell)}) \mid F[V_C] \simeq H[C] \ \wedge \ \forall v \in V_\partial : \deg_{I \cap F}(v) > 0\},$$

that is, the set of H-copies that fully use E_C, and where every vertex of V_∂ has a neighbor outside V_C in the copy, by an edge from I. Note that $F[V_G]$ however still may feature isolated vertices.

The size of $T^{(\ell)}$ will be computed via inclusion-exclusion in the following claim. Building upon this, we will later use interpolation to count those $F \in T(\ell)$ where $F[V_G]$ additionally contains no isolated vertices. (Note that ℓ is irrelevant if this additional property holds.)

Claim 6.6. We can compute $\#T(\ell)$ with $2^{\mathcal{O}(|V(H)|^2)}$ oracle calls to $\#\mathsf{Sub}(H \to G')$, where G' is a subgraph of $G^{(\ell)}$ in each call.

Proof. For $e \in E_C$, $v \in V_C$ and $w \in V_\partial$, let us define

$$\begin{aligned} \mathcal{A}_e &\equiv \{F \in \mathrm{Sub}(H \to G^{(\ell)}) \mid e \notin E(F)\}, \\ \mathcal{B}_v &\equiv \{F \in \mathrm{Sub}(H \to G^{(\ell)}) \mid v \notin V(F)\}, \\ \mathcal{D}_w &\equiv \{F \in \mathrm{Sub}(H \to G^{(\ell)}) \mid \deg_{I \cap F}(w) = 0\}. \end{aligned}$$

Then we observe that

$$T^{(\ell)} = \mathrm{Sub}(H \to G^{(\ell)}) \setminus \bigcup_{\substack{e \in E_C,\, v \in V_C \\ w \in V_\partial}} \mathcal{A}_e \cup \mathcal{B}_v \cup \mathcal{D}_w.$$

For $A \subseteq E_C$, $B \subseteq V_C$ and $D \subseteq V_\partial$, define a graph $G^{(\ell)}_{A,B,D}$ from $G^{(\ell)}$: Delete all edges in A, all vertices in B, and for all $w \in D$, delete all edges in $I(w) \cap F$. It is clear that then

$$\mathrm{Sub}(H \to G^{(\ell)}_{A,B,D}) \simeq \bigcap_{\substack{e \in A,\, v \in B \\ w \in D}} \mathcal{A}_e \cap \mathcal{B}_v \cap \mathcal{D}_w,$$

where the empty intersection is the set $\#\mathsf{Sub}(H \to G^{(\ell)})$ by convention.

The number $\#\mathsf{Sub}(H \to G^{(\ell)}_{A,B,D})$ can be computed by an oracle call to $\#\mathsf{Sub}$ with the pattern graph H, and hence we can compute $\#T^{(\ell)}$ by using the inclusion-exclusion principle (Lemma 1.33). □

Consider the graphs that can be written as $H - C'$, where $C' \subseteq V(H)$ is such that $H[C] \simeq H[C']$ via an isomorphism f satisfying (C2) and (C3) of Definition 6.3. Let $\mathcal{R}$ denote the set of such graphs, modulo isomorphism. Note that every $R \in \mathcal{R}$ is a graph on $2k$ vertices. Slightly abusing notation, we will henceforth write $A \in \mathcal{R}$ if there is some $A' \in \mathcal{R}$ with $A \simeq A'$.

Claim 6.7. If $F \in T(\ell)$, then $F[V_G] \in \mathcal{R}$.

Proof. Let $F \in T(\ell)$ and let f be an isomorphism from F to H that maps $F[V_C] \simeq H[C]$ to $H[C']$ for some $C' \subseteq V(H)$. We claim that f satisfies (C2) and (C3), which implies $F[V_G] \in \mathcal{R}$.

Condition (C2) holds since $F[V_G]$ is contained in (a copy of) the bipartite graph G. Concerning condition (C3), because $F \in T(\ell)$ holds, each vertex in V_∂ is adjacent to at least one vertex in $F[V_G]$, and thus $\partial_F(V_C) \supseteq V_\partial$. By construction of $G^{(\ell)}$, only vertices in V_∂ can be adjacent to $F[V_G]$, thus $\partial_F(V_C) \subseteq V_\partial$ and $\partial_F(V_C) = V_\partial$. The isomorphism f must therefore map V_∂ to $\partial_H(C')$. □

For $R \in \mathcal{R}$, let $\iota(R)$ denote the set of isolated vertices of R. If $R \in \mathcal{R}$ satisfies $\iota(R) = \emptyset$, then $R \simeq M$, where M is the induced matching of H: This holds because then the isomorphism that establishes $R \in \mathcal{R}$ satisfies (C1)-(C3) of Definition 6.3. However, since we

cannot generally assume that $\iota(R) = \emptyset$ holds, it may occur that $R \not\simeq M$. In the remainder of this proof, we will therefore "interpolate out" isolated vertices. To this end, we show that $\#T(\ell)$ may be interpreted as a weighted sum over $\mathcal{R}$, where the weight of each $R \in \mathcal{R}$ depends on its number of isolated vertices. In fact, we will see that $\#T(\ell)$ can be considered as a polynomial in ℓ that we can use for univariate interpolation.

For $R \in \mathcal{R}$, let $\alpha_R \in \mathbb{N}$ be defined as follows: Given the vertex-disjoint union of $H[C]$ and R, let α_R denote the number of edge-subsets that can be added between $\delta_H(C)$ and $V(R)$ such that the resulting graph is isomorphic to H. Note that we can compute $\alpha_R \in \mathbb{N}$ by brute force in time $2^{\mathcal{O}(|V(H)|^2)}$, and observe also that

$$\alpha_M > 0, \tag{6.3}$$

since H admits a partition into $H[C]$ and M by assumption.

Claim 6.8. With the abbreviation $\text{pure}(R) := R - \iota(R)$, we have

$$\#T(\ell) = \sum_{R \in \mathcal{R}} \#\mathsf{Sub}(\text{pure}(R) \to G) \cdot \alpha_R \cdot \binom{n + \ell - 2k + |\iota(R)|}{|\iota(R)|}. \tag{6.4}$$

Proof. By Claim 6.7, every $F \in T(\ell)$ satisfies $F[V_G] \in \mathcal{R}$. This induces a partition of $T(\ell)$ according to the isomorphism type of $F[V_G]$ for $F \in T(\ell)$: For $R \in \mathcal{R}$, let $\mathcal{A}_R$ denote the set of $F \in T(\ell)$ with $F[V_G] \simeq R$. Since the sets $\{\mathcal{A}_R\}_{R \in \mathcal{R}}$ partition $T(\ell)$, we have

$$\#T(\ell) = \sum_{R \in \mathcal{R}} \#\mathcal{A}_R. \tag{6.5}$$

We show (6.4) by calculating $\#\mathcal{A}_R$ for $R \in \mathcal{R}$ and observing that $\#\mathcal{A}_R$ is equal to the summand of R in (6.4). To this end, observe that every $F \in \mathcal{A}_R$ can be written as a union of a uniquely determined graph $F' \in \#\mathsf{Sub}(\text{pure}(R) \to G)$ and the following extensions:

E1: The induced graph $G^{(\ell)}[V_C] \simeq H[C]$, which must be contained in F, since $F \in T(\ell)$,

E2: A set of $|\iota(R)|$ isolated vertices in V_G, which may be chosen from vertices of the G-copy as well as from the ℓ added isolated vertices, and

E3: A set of edges from I.

For fixed $F' \in \#\mathsf{Sub}(\text{pure}(R) \to G)$, we determine the number of such extensions. In E1, there is only one choice. In E2, we extend by adding isolated vertices in V_G. As V_∂ connects to *all* of V_G, these $|\iota(R)|$ isolated vertices can be chosen arbitrarily among the $n + \ell - |V(\text{pure}(R))|$ vertices of V_G not already contained in $F'[V_G]$. Since $\text{pure}(R)$ has exactly $2k - |\iota(R)|$ vertices, the number of such extensions is given by the binomial coefficient

$$\binom{n + \ell - 2k + |\iota(R)|}{|\iota(R)|}.$$

So far, this procedure has fixed $F[V_G]$, and we reach step E3, where we extend the vertex-disjoint parts $F[V_G] \simeq R$ and $F[V_C] \simeq H[C]$ to a graph F by including edges

between V_∂ and V_G. The number of possible extensions in this step is α_R by definition. Thus, each $F' \in \#\mathsf{Sub}(\mathrm{pure}(R) \to G)$ can be extended to some $F \in \mathcal{A}_R$ by

$$\alpha_R \cdot \binom{n + \ell - 2k + |\iota(R)|}{|\iota(R)|}$$

possible extensions, and each F is the extension of a unique F', which shows (6.4). □

By (6.4), the value $\#T(\ell)$ can be interpreted as a polynomial $p \in \mathbb{Z}[x]$ of maximum degree $2k$ in the indeterminate $x := n + \ell - 2k$. Note that n and k are fixed by the input, whereas ℓ can be varied. Considering the binomial coefficient

$$\binom{n + \ell - 2k + |\iota(R)|}{|\iota(R)|} = \binom{x + |\iota(R)|}{|\iota(R)|}$$

as a polynomial in x, we can observe that $R \in \mathcal{R}$ with differing $|\iota(R)|$ yield polynomials of different degrees. In particular, this binomial coefficient (as a polynomial in x) has degree 0 (and is then equal to 1) if and only if $|\iota(R)| = 0$, i.e., when R contains no isolated vertices.

We observe that p can be interpolated by evaluating

$$\#T(0), \ldots, \#\mathsf{T}(2k),$$

where each evaluation can be performed with $2^{\mathcal{O}(|V(H)|^2)}$ oracle calls by Claim 6.6. This yields the representation of p in the standard monomial basis $\{x^i \mid i \in \mathbb{N}\}$. We then represent p as a linear combination of the basis elements $\mathcal{B} := \{\binom{x+i}{i} \mid i \in \mathbb{N}\}$. For every $i \in \mathbb{N}$, there is precisely one polynomial in $\mathcal{B}$ of degree i, and thus $\mathcal{B}$ is linearly independent. Thus, the coefficients of p over the basis $\mathcal{B}$ are uniquely determined and we can extract the constant coefficient

$$c_0 = \alpha_M \cdot \#\mathsf{Sub}(M \to G).$$

Recall that $\alpha_M > 0$ by (6.3). We have shown how to compute k-matchings in G. □

Remark 6.9. The final interpolation step could equivalently be achieved by solving a linear system of equations whose system matrix consists of the binomial coefficients in (6.4). In fact, this system appears implicitly in the argument above.

Lemma 6.5 readily implies the following reduction.

Lemma 6.10. *If $\mathcal{H}$ is a recursively enumerable graph class that contains a k-matching gadget for every $k \in \mathbb{N}$, then $\#\mathsf{BipMatch} \leq^T_{fpt} \#\mathsf{Sub}(\mathcal{H})$.*

Proof. We reduce $\#\mathsf{BipMatch} \leq_{fpt} \#\mathsf{Sub}(\mathcal{H})$: Given a bipartite graph G and $k \in \mathbb{N}$, we wish to compute the number of k-matchings in G.

For graphs H and subsets $M \subseteq V(H)$, we can test whether (H, M) is a k-matching gadget by checking all isomorphisms relevant to Definition 6.3 by brute force. The runtime required for this step depends only on $|V(H)|$. To determine the number of k-matchings in

G, enumerate the graphs $H \in \mathcal{H}$ and subsets $M \subseteq H$ until we find a k-matching gadget (H, M), which is guaranteed to exist by assumption. Then invoke Lemma 6.5 with (H, M).

Let $g(k) = |V(H)|$, where H is the k-matching gadget found by the enumeration procedure above. Then the function g is computable, which implies together with Lemma 6.5 that the reduction to $\#\mathsf{Sub}(\mathcal{H})$ is indeed a parameterized Turing reduction. □

This also concludes the proof of Theorem 6.11.

Theorem 6.11 (restated from page 120)**.** *Let $\mathcal{H}$ be a recursively enumerable graph class. If $\mathsf{FPT} \neq \#\mathsf{W}[1]$, then the following are equivalent:*

- $\#\mathsf{Sub}(\mathcal{H})$ *is polynomial-time solvable.*
- $\#\mathsf{Sub}(\mathcal{H})$ *is fixed-parameter tractable when parameterized by $|V(H)|$.*
- $\mathcal{H}$ *has bounded vertex-cover number.*

Proof. For convenience, we recall the proof:

- If $\mathcal{H}$ has bounded vertex-cover number, then we can apply Theorem 6.1 to solve $\#\mathsf{Sub}(\mathcal{H})$ in polynomial time.
- If $\mathcal{H}$ has unbounded treewidth, then we use Theorem 5.6 to obtain $\#\mathsf{W}[1]$-hardness of $\#\mathsf{Sub}(\mathcal{H})$.
- If $\mathcal{H}$ has bounded treewidth and unbounded vertex-cover number, we use Theorem 6.4 and Lemma 6.10 to obtain $\#\mathsf{BipMatch} \leq^T_{fpt} \#\mathsf{Sub}(\mathcal{H})$. By Theorem 5.24, the problem $\#\mathsf{BipMatch}$ is $\#\mathsf{W}[1]$-complete, so we obtain the same for $\#\mathsf{Sub}(\mathcal{H})$.

This proves the theorem. □

Part III.

Quantitative lower bounds for counting

Introduction to Part III

In the final part of this thesis, we focus on conditional lower bounds for counting problems under the exponential-time hypothesis #ETH from Section 1.2.3. Furthermore, we also undertake a short promenade in the structural complexity of the problem PerfMatch.

We have already shown conditional lower bounds in Section 5.2.2: Unless #ETH fails, we cannot count k-matchings, k-cycles or k-paths of an n-vertex graph in time $n^{o(k/\log k)}$. In this part, we focus on lower bounds for parameters that explicitly depend on the input size, such as the number of vertices or edges.

Counting unweighted perfect matchings revisited

In Section 2.2.3, we have seen a #P-hardness proof for counting perfect matchings in unweighted graphs by reduction from #3-SAT. This proof proceeds along the following lines, which are common to the standard proofs in the literature [Val79b, Pap94]. Recall that PerfMatch^X asks to evaluate PerfMatch on graphs with edge-weights from X.

$$\#\text{3-SAT} \leq_p \text{PerfMatch}^{-1,0,1} \leq_p^T \text{PerfMatch}^{0,1}.$$

More specifically, given a 3-CNF formula φ on n variables and m clauses, we first constructed, in Lemma 2.29 and Corollary 2.31, a graph $G = G(\varphi)$ on edge-weights -1 and 1, and a number $T \in \mathbb{N}$ such that

$$\#\text{SAT}(\varphi) = 2^{-T} \cdot \text{PerfMatch}(G). \tag{6.6}$$

This graph G has $\mathcal{O}(n+m)$ vertices and edges. In Corollary 2.32, the computation of $\text{PerfMatch}(G)$ was then reduced to counting perfect matchings in unweighted graphs. Note that hardness for unweighted graphs is required for subsequent reductions from counting perfect matchings to other problems, as we could otherwise only reduce to versions of these problems that are weighted as well. For implementing this *weight removal step* for PerfMatch, the literature offers us two choices, of which we chose the first:

Weight removal by interpolation: As in Lemma 1.37, we can use univariate interpolation and $\mathcal{O}(n)$ oracle calls to the problem $\text{PerfMatch}^{0,1}$ on unweighted graphs. With Lemma 1.37, each oracle query has $\mathcal{O}(nm)$ vertices and edges, but these quantities can be reduced to $\mathcal{O}(m \log n)$ by simple matchgates. This way, a $2^{\Omega(m/\log n)}$ lower bound under #ETH was obtained for computing $\text{PerfMatch}^{0,1}$ on graphs with m edges and n vertices [DHM$^+$14]. We will revisit weight removal by interpolation later in this introduction.

Weight removal by modular arithmetic: We can replace the edge-weight -1 by $r-1$, with $r := 2^m + 1$, to obtain a new graph G'. Since $r-1$ is positive on all relevant instances G, this weight can be simulated by a matchgate as in Lemma 1.29, and it can be seen as above that a matchgate on $\mathcal{O}(m)$ vertices and edges suffices, yielding a total number of $\mathcal{O}(nm)$ vertices and $\mathcal{O}(m^2)$ edges. Then we compute $\text{PerfMatch}(G')$ modulo r, which gives the correct value of $\text{PerfMatch}(G)$, since $\text{PerfMatch}(G) < r$. This is the original route that was taken in [Val79b].

Note that none of these approaches yield tight lower bounds for $\text{PerfMatch}^{0,1}$, a problem that can be solved trivially in time $2^{\mathcal{O}(m)}$ on graphs with m edges. In fact, the problem can also be solved in time $2^{O(n)}$ on graphs with n vertices, as noted in Section 1.3.2.

In Chapter 7, we present a third and novel way of performing the weight removal step, which substantially differs from both approaches mentioned before:

Weight removal by difference: Compute two unweighted graphs G_1 and G_2 from G such that $\text{PerfMatch}(G)$ is the difference of $\text{PerfMatch}(G_1)$ and $\text{PerfMatch}(G_2)$.

By the following lemma, shown in Section 7.1, we can indeed construct such unweighted graphs G_1 and G_2 in polynomial time, and we can also show that a linear blowup on the number of vertices and edges suffices to obtain G_1 and G_2 from G. This gives us a new weight removal step for proving #P-completeness of $\text{PerfMatch}^{0,1}$.

Lemma 7.1. *Let G be a graph on n vertices and m edges with edge-weights $\{-1, 1\}$. Then we can construct two unweighted graphs G_1 and G_2 with $\mathcal{O}(n+m)$ vertices and edges such that*

$$\text{PerfMatch}(G) = \text{PerfMatch}(G_1) - \text{PerfMatch}(G_2).$$

Furthermore, this construction can be carried out in time $\mathcal{O}(n+m)$.

Since the edge-weighted graph G obtained from φ in (6.6) has only $\mathcal{O}(n+m)$ vertices and edges, and its edge-weights are only -1 and 1, this implies a lower bound for $\text{PerfMatch}^{0,1}$ under #ETH.

Theorem 7.4. *Unless #ETH fails, the problem $\text{PerfMatch}^{0,1}$ admits no algorithm with running time $2^{o(m)}$ on simple graphs with m edges.*

Deciding whether PerfMatch agrees on two graphs

Apart from tight quantitative lower bounds, Lemma 7.1 also allows us to derive implications for the structural complexity of PerfMatch: We show that deciding whether two graphs agree in their numbers of perfect matchings is complete for the complexity class $\mathsf{C}_=\mathsf{P}$.

To define $\mathsf{C}_{=}\mathsf{P}$, let us first define the following decision problem $\mathsf{A}_{=}$ associated with each counting problem $\mathsf{A} \in \#\mathsf{P}$: The inputs to $\mathsf{A}_{=}$ are pairs (x, y), and we are asked to determine whether $\mathsf{A}(x) = \mathsf{A}(y)$ holds. The class $\mathsf{C}_{=}\mathsf{P}$ is then defined as

$$\mathsf{C}_{=}\mathsf{P} := \{\mathsf{A}_{=} \mid \mathsf{A} \in \#\mathsf{P}\}.$$

For instance, it is clear that

$$\mathsf{SAT}_{=} := \{(\varphi, \varphi') \mid \#\mathsf{SAT}(\varphi) = \#\mathsf{SAT}(\varphi')\}$$

is $\mathsf{C}_{=}\mathsf{P}$-complete under the classical notion of polynomial-time many-one reductions. In fact, $\mathsf{C}_{=}\mathsf{P}$-hardness holds for every problem $\mathsf{A}_{=}$ whose counting version $\#\mathsf{A}$ is $\#\mathsf{P}$-hard under parsimonious reductions.

The relationship between $\mathsf{C}_{=}\mathsf{P}$ and other complexity classes has been studied in the literature of structural complexity theory, and several results are surveyed in [HO02]. For instance, we have $\mathsf{coNP} \subseteq \mathsf{C}_{=}\mathsf{P}$, and using the Isolation Lemma [VV86], we see that NP is contained in $\mathsf{C}_{=}\mathsf{P}$ under randomized reductions. Let us also observe that $\mathsf{NP}^{\#\mathsf{P}} \subseteq \mathsf{NP}^{\mathsf{C}_{=}\mathsf{P}}$: Whenever we issue an oracle call to $\#\mathsf{P}$, we may guess the output number, and then check whether we guessed correctly by using the $\mathsf{C}_{=}\mathsf{P}$ oracle.

To the best of the author's knowledge, the literature on complexity theory does not contain any *natural* $\mathsf{C}_{=}\mathsf{P}$-complete problem $\mathsf{A}_{=}$ whose counting version A is *not* $\#\mathsf{P}$-complete under parsimonious reductions. Here, we stressed *natural*, because we can easily construct artifical $\mathsf{C}_{=}\mathsf{P}$-complete problems $\mathsf{A}_{=}$ whose counting version $\#\mathsf{A}$ admits no parsimonious reduction from $\#\mathsf{SAT}$. To see this, consider the counting problem $\#\mathsf{SAT}'$ that asks to compute the number of satisfying assignments to a CNF-formula φ, incremented by 1. If $\#\mathsf{SAT}'$ had a parsimonious reduction from $\#\mathsf{SAT}$, then every CNF-formula would be satisfiable. On the other hand, the reduction from $\#\mathsf{SAT}_{=}$ to $\#\mathsf{SAT}'_{=}$ is trivial.

Given this background, the following completeness result for $\mathrm{PerfMatch}^{0,1}_{=}$, which we prove in Section 7.3, might be of interest in the context of structural complexity theory. As mentioned in Section 1.2.1, the problem $\mathrm{PerfMatch}^{0,1}$ of counting unweighted perfect matchings cannot be $\#\mathsf{P}$-complete under parsimonious reductions, unless $\mathsf{P} = \mathsf{NP}$.

Theorem 7.5. *The problem* $\mathrm{PerfMatch}^{0,1}_{=}$ *is* $\mathsf{C}_{=}\mathsf{P}$*-complete under polynomial-time many-one reductions. Recall that this problem asks whether two unweighted graphs* G_1 *and* G_2 *have the same number of perfect matchings.*

The complexity of a similar problem was posed as an open question in [Che10]: Given two directed acyclic graphs, decide whether their numbers of topological orderings agree. It is known that counting topological orderings is $\#\mathsf{P}$-complete under $\leq_p^T$, as shown in [BW91], but the decision version is trivial for acyclic graphs. Our result for $\mathrm{PerfMatch}^{0,1}_{=}$ might be useful to prove $\mathsf{C}_{=}\mathsf{P}$-completeness for this problem.

The limits of interpolation

In the following, we reconsider the technique of polynomial interpolation for lower bounds under #ETH. As a running example, let us recall how we reduce PerfMatch$^{-1,0,1}$ to PerfMatch^{0,1} by means of interpolation in Lemmas 1.36 and 1.37. Let G be a given graph on n vertices and m edges with edge-weights $\{-1, 1\}$.

Interpolation: In Lemma 1.36, we define a polynomial p of degree $\frac{n}{2}$ such that $p(-1) = \text{PerfMatch}(G)$. Using univariate interpolation on $p(0), \ldots, p(\frac{n}{2})$, we recover all coefficients of $p(G)$ and can thus evaluate $p(-1)$. The evaluation $p(i)$ for $i \in [\frac{n}{2}]$ is obtained as PerfMatch(G') for some graph with the edge-weights $\{1, i\}$.

Removing weights: We reduce the evaluation of PerfMatch(G') for graphs with edge-weights $\{1, i\}$ to PerfMatch(G'') for unweighted graphs G'' with $\mathcal{O}(n + im)$ vertices and edges. Here, we use the edge-weight simulation for PerfMatch described in Lemma 1.29: Each edge of weight i is replaced by a matchgate on $2i$ vertices, obtained by subdividing i parallel edges twice.

By Corollary 2.31, the problem PerfMatch$^{-1,0,1}$ admits no $2^{o(m)}$ time algorithm on simple graphs with m edges. The above reduction from PerfMatch$^{-1,0,1}$ to PerfMatch^{0,1} also yields a lower bound for the target problem, which can however only rule out a $2^{o(\sqrt{n})}$ time algorithm: One of the unweighted graphs simulates $\Omega(m)$ edges of weight $\frac{n}{2}$, and thus has $\Omega(nm) = \Omega(n^2)$ vertices and edges.

Using slightly more complicated matchgates, as in [DHM$^+$14], an edge-weight $i \in \mathbb{N}$ can be simulated by a matchgate with $\mathcal{O}(\log i)$ rather than $\mathcal{O}(i)$ vertices, implying a $2^{\Omega(m/\log n)}$ lower bound for PerfMatch^{0,1}. This bound is however still not tight, and in particular, we have "lost tightness" in the reduction from PerfMatch$^{-1,0,1}$ to PerfMatch^{0,1}.

In fact, we observe that a combination of interpolation and weight removal as above can *never* yield tight lower bounds for PerfMatch^{0,1}: When carried out on n-vertex graphs G, we need to simulate $\frac{n}{2}+1$ distinct edge-weights, because the polynomial p to be interpolated has degree $\frac{n}{2}$. This in turn requires $\frac{n}{2}+1$ *distinct* matchgates to be placed at vertices of G, since isomorphic matchgates clearly have the same signature. Then, since there are only finitely many simple graphs on $\mathcal{O}(1)$ vertices, the size of such gadgets must necessarily grow as some unbounded function $\alpha(n)$, and hence we can only obtain lower bounds of the form $2^{\Omega(m/\alpha(n))}$ for $\alpha \in \omega(1)$ under #ETH.

Additionally, the above type of reduction runs in polynomial time, which is required in the setting of #P-hardness, but not essential in exponential-time complexity: We are free to use a subexponential reduction family, as defined in Section 1.2.3. Intuitively speaking, the reduction we described above was *too* efficient on the running time.

The limitations we observed in the preceding paragraphs are immanent to every known lower bound under #ETH, such as $2^{\Omega(m/\log^c n)}$ time bounds for the Tutte polynomial and the permanent [DHM$^+$14], and $2^{\Omega(n/\log^c n)}$ bounds for the independent set polynomial [Hof10]. All of these results proceed essentially along the lines described above, and they construct "economical" gadgets for weight simulations that yield only poly-logarithmic (yet

super-constant) blowup. By arguments as for PerfMatch, these reductions cannot yield tight lower bounds under #ETH.

The block interpolation framework

We circumvent these barriers in Chapter 8 by introducing the *block interpolation framework*, a general method that allows us to apply the full power of subexponential reductions to counting problems. To this end, we show how to apply multivariate polynomial interpolation in a useful way:[1] In the setting we establish in our framework, we will not be given an unknown *univariate* polynomial p of degree n, which we have to interpolate from $n+1$ oracle calls, but rather a *multivariate* polynomial $\mathbf{p}$ with total degree n and maximum degree $c = \mathcal{O}(1)$ in each indeterminate. We can interpolate this polynomial from $2^{o(n)}$ evaluations $\mathbf{p}(\xi)$ at tuples ξ whose individual entries are contained in a fixed set of size $c+1$. This will enable us to compute $\mathbf{p}(\xi)$ by attaching copies of only $c+1$ distinct gadgets to G. The catch here is that different edges may obtain different gadgets, which was not feasible in the univariate setting.

Our technique is phrased as a general framework that allows us to convert a large body of #P-hardness proofs into tight lower bounds under #ETH. The growth of the gadgets used to simulate weights in the original proofs is *irrelevant* in our framework, as only a constant number of gadgets will be used. In particular, we do not need to worry about economical gadget constructions, and can thus obtain proofs that are significantly simpler than the previously known proofs. And more importantly, our results are tight.

To showcase our framework, we prove that #ETH implies $2^{\Omega(m)}$ lower bounds for the following problems on unweighted simple graphs G with m edges. All of these problems admit trivial algorithms with running time $2^{\mathcal{O}(m)}$ on such graphs.

Theorem 8.12. *Unless* #ETH *fails, the problem* $\text{perm}^{0,1}$ *does not admit an algorithm with running time* $2^{o(m)}$ *on simple graphs with* m *edges. Recall that* perm *is the restriction of* PerfMatch *to bipartite graphs.*

It was shown in [DHM+14] that, unless #ETH fails, the problem $\text{perm}^{-1,0,1}$ admits no algorithm with running time $2^{o(m)}$. It was also shown that such an algorithm for $\text{perm}^{0,1}$ would falsify rETH, the randomized version of ETH. Under #ETH, only a lower bound of $2^{\Omega(m/\log n)}$ was obtained.

Note that Theorem 8.12 also improves upon Theorem 7.4, which proved a tight lower bound only for $\text{PerfMatch}^{0,1}$, that is, for graphs that are not necessarily bipartite. However, the block interpolation technique cannot be applied to show $\mathsf{C}_{=}\mathsf{P}$-completeness.

[1] In the context of the Tutte polynomial, Sokal [Sok05] coined the term of a "multivariate ideology" for arguments that benefit strongly from a multivariate point of view. The block interpolation framework definitely is a profiteer of this ideology.

Theorem 8.15. *Unless #ETH fails, the problem of evaluating the matching polynomial $\mu(G;\xi)$ for any fixed $\xi \in \mathbb{Q}$ does not admit an algorithm with running time $2^{o(m)}$ on simple graphs G with m edges, even when the maximum degree $\Delta(G)$ may be assumed to be a constant. The same applies to the independent set polynomial $I(G;\xi)$ at fixed $\xi \in \mathbb{Q} \setminus \{0\}$.*

Note that by Definition 1.22 on page 37, the problems above are formally denoted by $\mathsf{Eval}_\xi(\mu)$ and $\mathsf{Eval}_\xi(I)$ for fixed $\xi \in \mathbb{Q}$. We have seen in Theorem 1.24 that the evaluation of $\mu(G;\xi)$ is #P-complete at all $\xi \in \mathbb{Q}$, but to the best of our knowledge, no lower bounds under #ETH are known in the literature. In [Hof10], a lower bound of $2^{\Omega(n/\log^3 n)}$ for the independent set polynomial was shown at general $\xi \in \mathbb{Q} \setminus \{0\}$, and $2^{\Omega(n)}$ was obtained at $\xi = 1$, but neither of these bounds yield hardness results for sparse graphs.

Theorem 8.21. *Unless #ETH fails, the Tutte polynomial $T(x,y)$ for fixed $(x,y) \in \mathbb{Q}^2$ cannot be evaluated in time $2^{o(m)}$ on simple graphs with m edges, provided that*

- $y \notin \{0,1\}$,
- $(x,y) \notin \{(1,1),(-1,-1),(0,-1),(-1,0)\}$, *and*
- $(x-1)(y-1) \neq 1$.

In [DHM+14], only lower bounds of the type $2^{\Omega(n/\log^c n)}$ were shown on sparse (simple) graphs, and quite involved gadget constructions were required for these bounds. We can circumvent these constructions almost entirely, and instead use only elementary gadgets.

If $(x,y) \in \mathbb{Q}^2$ violates either of the last two conditions of the corollary, then $T(x,y)$ is known to admit a polynomial-time algorithm. If (x,y) satisfies both conditions, then it is #P-hard, shown in [JVW90]. The #P-hard points with $y \in \{0,1\}$ are however not covered by the lower bounds of Theorem 8.21. At the line $y = 0$, a lower bound of $2^{\Omega(n)}$ can be achieved on n-vertex graphs that are not necessarily sparse, but the line $y = 1$ is left open, as in [DHM+14].

Notes

All content of this part was obtained independently. After submission of this thesis, the content of Chapter 8 was published in [Cur15].

7. Unweighted perfect matchings

7.1. Weight reduction by difference

In this section, we prove Lemma 7.1, and thus show how to reduce PerfMatch on graphs with edge-weights $\{-1, 1\}$ to the difference of PerfMatch on two unweighted graphs.

Lemma 7.1 (restated from page 154)**.** *Let G be a graph on n vertices and m edges with edge-weights $\{-1, 1\}$. Then we can construct two unweighted graphs G_1 and G_2 with $\mathcal{O}(n + m)$ vertices and edges such that*

$$\text{PerfMatch}(G) = \text{PerfMatch}(G_1) - \text{PerfMatch}(G_2).$$

Furthermore, this construction can be carried out in time $\mathcal{O}(n + m)$.

Our proof proceeds by establishing the reduction chain

$$\text{PerfMatch}^{-1,0,1} \leq_p \text{MatchSum}^{-1,0,1} \leq_p^T \text{PerfMatch}^{0,1}.$$

Recall the definitions of the graph polynomials PerfMatch and MatchSum from Section 1.3, and the convention that a superscript $X \subseteq \mathbb{Q}$ on each of these problems denotes the restriction of the problem to graphs with weights from the set X.

Note that the computational problem $\text{PerfMatch}^{0,1}$ is more general than $\text{perm}^{0,1}$, as the input graphs to $\text{PerfMatch}^{0,1}$ are not required to be bipartite. The approach we describe in the following fails to yield hardness for bipartite graphs. If we are only interested in tight lower bounds under #ETH, we can instead use the approach from Chapter 8.

Proof of Lemma 7.1. Let G be a graph with edge-weights from $\{-1, 1\}$. We assume that $|V(G)|$ is even, as otherwise $\text{PerfMatch}(G) = 0$. To prove Lemma 7.1, we first write

$$\text{PerfMatch}(G) = \text{Holant}(\Omega), \tag{7.1}$$

where Ω is the signature graph constructed by assigning $\texttt{HW}_{=1}$ to all vertices of G, subdividing all edges of weight -1, removing all edge-weights, and assigning the signature $\texttt{EDGE}_{-1}$ to the newly introduced subdivision vertices. We have already seen this way of removing weights from signature graphs in Lemma 2.8. For convenience, we recall that

$$\texttt{EDGE}_{-1} : \quad x \mapsto \begin{cases} -1 & \text{if } x = 11, \\ 0 & \text{if } x \in \{01, 10\}, \\ 1 & \text{if } x = 00. \end{cases}$$

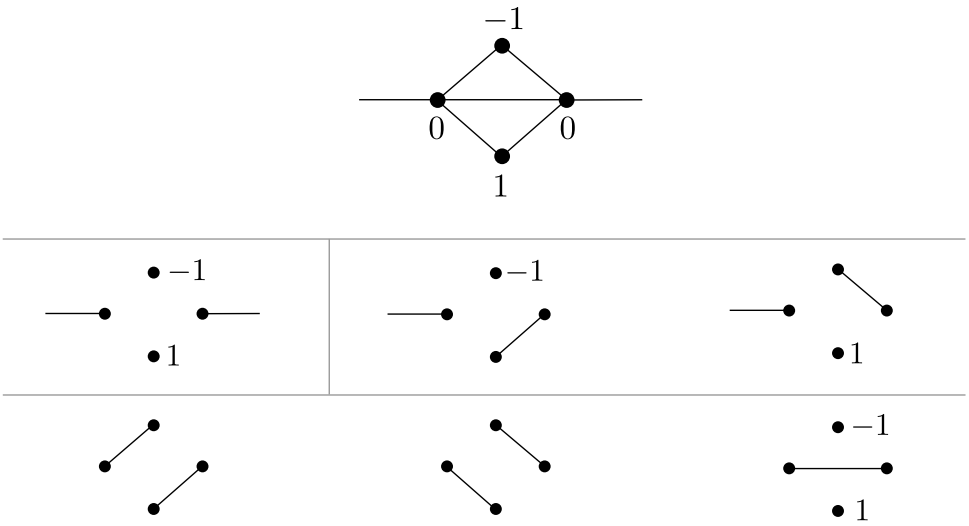


Figure 7.1.: The gate Γ is shown at the top, where a number $c \in \mathbb{Z}$ at a vertex indicates that the signature $\mathtt{VTX}_c$ is assigned to it. The satisfying assignments of Γ are shown below; note that these are (not necessarily perfect) matchings. Symmetric cases in which only the right dangling edge is active are not shown.

Then we realize each occurrence of the signature $\mathtt{EDGE}_{-1}$ by the gate Γ from Figure 7.1, which features no edge-weights, and only the signature $\mathtt{VTX}_w$ for $w \in \{-1, 0, 1\}$. Recall that this is the signature used in Example 2.6 on page 56 to express MatchSum as a Holant problem. For convenience, we recall that

$$\mathtt{VTX}_w : \quad x \mapsto \begin{cases} w & \text{if } \mathrm{hw}(x) = 0, \\ 1 & \text{if } \mathrm{hw}(x) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Claim 7.2. We have $\mathrm{Sig}(\Gamma) = \mathtt{EDGE}_{-1}$, where Γ is the gate from Figure 7.1.

Proof. Given an assignment $x \in \{0,1\}^2$ to the dangling edges of Γ, we list the satisfying assignments $xy \in \{0,1\}^{E(\Gamma)}$ that extend x in the bottom part of Figure 7.1. Note that all such assignments are (not necesssarily perfect) matchings.

$x = 11$: Only the empty matching can be chosen. It has weight -1, thus $\mathrm{Sig}(\Gamma, 11) = -1$.

$x = 10$: Two matchings can be chosen, which have opposite weights, thus $\mathrm{Sig}(\Gamma, 10) = 0$. By symmetry, the same is true for $\mathrm{Sig}(\Gamma, 01)$.

$x = 00$: Three matchings can be chosen, of which two have weight 1 and one has weight -1, thus $\mathrm{Sig}(\Gamma, 00) = 1$.

This proves the claim. □

By realizing every occurrence of $\texttt{EDGE}_{-1}$ in Ω with Γ, we obtain a signature graph Ω' whose signatures are all of the type $\texttt{VTX}_w$ for $w \in \{-1, 0, 1\}$, and which satisfies

$$\text{Holant}(\Omega) = \text{Holant}(\Omega'). \tag{7.2}$$

Note here that $\texttt{HW}_{=1}$ is equivalent to $\texttt{VTX}_0$. As in Example 2.6, we consider

$$\text{Holant}(\Omega') = \text{MatchSum}(G'), \tag{7.3}$$

where G' is a vertex-weighted graph that is obtained from Ω' as follows: Keep all vertices and edges of Ω' intact, and if $v \in V(\Omega')$ features the signature $\texttt{VTX}_w$, for $w \in \{-1, 0, 1\}$, then assign the vertex weight w to v in G'.

Note that $|V(G')|$ is even, since evenly many additional vertices were added to G to construct G'. Let V_x for $x \in \{-1, 1\}$ denote the set of vertices of weight x in G', and note that $V_{-1} \cap V_1 = \emptyset$ holds trivially. From G', we construct the signature graph $\Phi = \Phi(G')$ from Lemma 2.39 on page 73: Add a vertex w_1 with signature $\texttt{PROD}_{\mathbf{1}}$ to G' that is adjacent to V_1, and a vertex w_{-1} with $\texttt{PROD}_{-\mathbf{1}}$ adjacent to V_{-1}. Here, $\mathbf{1}$ denotes the all-ones vector of appropriate length.[1] Note that

$$\begin{aligned} \texttt{PROD}_{\mathbf{1}} &: \quad x \mapsto 1 \\ \texttt{PROD}_{-\mathbf{1}} &: \quad x \mapsto (-1)^{\text{hw}(x)}. \end{aligned}$$

All other vertices are assigned $\texttt{HW}_{=1}$ by the construction of Φ in Lemma 2.39. Then Lemma 2.39 implies that

$$\text{MatchSum}(G') = \text{Holant}(\Phi). \tag{7.4}$$

Observe that, for signatures EVEN and ODD of appropriate arities, we have

$$\begin{aligned} \texttt{PROD}_{\mathbf{1}} &= \texttt{EVEN} + \texttt{ODD}, \\ \texttt{PROD}_{-\mathbf{1}} &= \texttt{EVEN} - \texttt{ODD}. \end{aligned}$$

Using the Combined Signature Lemma (Lemma 2.35), we hence obtain four signature graphs Φ_{ab} for $a, b \in \{0, 1\}$ such that

$$\text{Holant}(\Phi) = \sum_{a,b \in \{0,1\}} (-1)^b \cdot \text{Holant}(\Phi_{ab}),$$

where Φ_{ab} is obtained from Φ by replacing

- $\texttt{PROD}_{\mathbf{1}}$ with EVEN if $a = 0$, or with ODD if $a = 1$, and
- $\texttt{PROD}_{-\mathbf{1}}$ with EVEN if $b = 0$, or with ODD if $b = 1$.

[1] Lemma 2.39 also adds a vertex w_0 adjacent to vertices of zero weight, with a signature $\texttt{PROD}_{\mathbf{0}}$, where $\mathbf{0}$ is the all-zeros vector. Since no satisfying assignment has an active edge incident with w_0, we can simply delete w_0 and its incident edges.

By Remark 2.40, only Φ_{00} and Φ_{11} can have satisfying assignments, so we obtain

$$\mathrm{Holant}(\Phi) = \mathrm{Holant}(\Phi_{00}) - \mathrm{Holant}(\Phi_{11}).$$

We realize Φ_{00} and Φ_{11} by the matchgates from Lemma 2.24, which are unweighted. This yields two unweighted graphs G_{00} and G_{11} such that

$$\mathrm{Holant}(\Phi) = \mathrm{PerfMatch}(G_{00}) - \mathrm{PerfMatch}(G_{11}), \tag{7.5}$$

which concludes the proof of Lemma 7.1 by following (7.1)-(7.5) □

7.2. Tight lower bounds under #ETH

From Lemma 7.1, we obtain #P-completeness of $\mathrm{PerfMatch}^{0,1}$ and a tight lower bound under #ETH for sparse graphs. To this end, we first reduce #SAT to $\mathrm{PerfMatch}^{-1,0,1}$ by means of Corollary 2.31, and then use Lemma 7.1 to remove the edge-weight -1.

Lemma 7.3. *Let φ be a 3-CNF formula with n variables and m clauses. Then we can compute a number $T \in \mathbb{N}$ and construct two unweighted graphs G_1 and G_2 on $\mathcal{O}(n+m)$ vertices and edges, all in time $\mathcal{O}(n+m)$, such that*

$$\#\mathsf{SAT}(\varphi) = 2^{-T} \cdot \left(\mathrm{PerfMatch}(G_1) - \mathrm{PerfMatch}(G_2)\right). \tag{7.6}$$

Proof. Given φ, use Corollary 2.31 to construct a graph G on $\mathcal{O}(n+m)$ vertices and edges with edge-weights from $\{-1, 1\}$, together with a number $T \in \mathbb{N}$, such that

$$\#\mathsf{SAT}(\varphi) = 2^{-T} \cdot \mathrm{PerfMatch}(G).$$

Use Lemma 7.1 to obtain the unweighted graphs G_1, G_2. □

In particular, Lemma 7.3 gives a polynomial-time Turing reduction from 3-#SAT to the problem $\mathrm{PerfMatch}^{0,1}$, which yields an alternative proof of Corollary 2.32.

By the exponential-time hypothesis #ETH, there is no $2^{o(n)}$ algorithm for counting satisfying assignments to 3-CNF formulas φ with n variables and $m = \mathcal{O}(n)$ clauses. Then Lemma 7.3 implies the following theorem.

Theorem 7.4 (restated from page 154)**.** *Unless #ETH fails, the problem $\mathrm{PerfMatch}^{0,1}$ admits no algorithm with running time $2^{o(m)}$ on simple graphs with m edges.*

7.3. Completeness for $\mathsf{C}_{=}\mathsf{P}$

We can also use Lemma 7.1 to show $\mathsf{C}_{=}\mathsf{P}$-completeness of the problem $\mathrm{PerfMatch}^{0,1}_{=}$, and hence prove Theorem 7.5: Given as input two unweighted graphs G_1 and G_2 on n vertices and $\mathcal{O}(n)$ edges, this problem asks to decide whether their numbers of perfect matchings agree. We will also show that this problem does not admit a $2^{o(n)}$ algorithm under ETH, the decision version of #ETH.

In the following, let us say that two formulas φ_1 and φ_2 are *equipollent* if their numbers of satisfying assignments agree.[2] Likewise, call unweighted graphs G_1 and G_2 equipollent if their numbers of perfect matchings agree.

Theorem 7.5 (restated from page 155)**.** *The problem* $\text{PerfMatch}_{=}^{0,1}$ *is* $\mathsf{C}_{=}\mathsf{P}$*-complete. Recall that this problem asks whether two unweighted graphs* G_1 *and* G_2 *are equipollent.*

Proof. The problem $\text{PerfMatch}_{=}^{0,1}$ is clearly contained in $\mathsf{C}_{=}\mathsf{P}$. For the hardness part, we reduce from the problem $\#\mathsf{SAT}_{=}$ defined on page 155, so we are given 3-CNF formulas φ and φ' on n variables, and we wish to determine whether they are equipollent. To this end, we construct unweighted graphs G and G' that are equipollent if and only if φ and φ' are.

Assume that φ and φ' are defined on the same set of variables $x_1, \dots, x_n$ and feature the same number m of clauses. This can be achieved by renaming variables, and by adding dummy variables and clauses.[3] Let $C_1, \dots, C_m$ and $C'_1, \dots, C'_m$ denote the clauses in φ and φ', respectively.

We introduce a *selector* variable x^* and define a formula ψ on the variable set $\mathcal{X} = \{x^*, x_1, \dots, x_n\}$, which has clauses $D_1, \dots, D_m$ and $D'_1, \dots, D'_m$, where

$$\begin{aligned} D_i &:= (x^* \vee C_i) && \text{for } i \in [m], \\ D'_i &:= (\neg x^* \vee C'_i) && \text{for } i \in [m]. \end{aligned}$$

If $a(x^*) = 0$ holds in an assignment $a \in \{0,1\}^{\mathcal{X}}$, then all clauses $D'_1, \dots, D'_m$ are satisfied by $\neg x^*$, but in order for a to satisfy ψ, the clauses $D_1, \dots, D_m$ have to be satisfied by $x_1, \dots, x_n$. In other words,

- if a satisfies ψ and $a(x^*) = 0$, then the restriction of a to $x_1, \dots, x_n$ satisfies φ, and
- if a satisfies ψ and $a(x^*) = 1$, then the restriction of a to $x_1, \dots, x_n$ satisfies φ'.

Hence, we can define the quantity

$$S := \sum_{a \in \{0,1\}^{\mathcal{X}}} (-1)^{a(x^*)} \cdot [\psi \text{ satisfied by } a],$$

and we observe that

$$S = \#\mathsf{SAT}(\varphi) - \#\mathsf{SAT}(\varphi'). \tag{7.7}$$

It is clear that $S = 0$ holds if and only if φ and φ' are equipollent.

As in Example 2.7 on page 57, we express $S = \text{Holant}(\Omega)$ for a signature graph $\Omega = \Omega(\psi)$, with one modification: At the vertex v^* corresponding to the variable x^*, we do not use

[2] The author found the adjective "equipollent" by checking a list of adjectives that are equivalent to "equivalent", and choosing the least commonly used.

[3] If, say, φ has less variables than φ', then we can add dummy variables to φ', together with clauses that ensure that every dummy variable has the same assignment as x_1. We can also duplicate clauses.

the signature $\texttt{EQ}$, but rather a modified signature $\texttt{EQ}_- : \{0,1\}^{I(v^*)} \to \{-1,1\}$

$$\texttt{EQ}_- : \quad y \mapsto \begin{cases} -1 & \text{if } y = 1 \dots 1, \\ 1 & \text{if } y = 0 \dots 0, \\ 0 & \text{otherwise.} \end{cases}$$

By Lemma 2.25 on page 66, this signature admits a matchgate with edge-weights $\{-1, 1/2, 1\}$ and $\mathcal{O}(n)$ vertices and edges. We realize Ω via Theorem 2.28, simulate the edge-weight $1/2$ via Remark 1.30, and obtain an edge-weighted graph H with weights $\{-1, 1\}$ and a number $T \in \mathbb{N}$ such that

$$S = \text{Holant}(\Omega) = 2^{-T} \cdot \text{PerfMatch}(H). \tag{7.8}$$

Using Lemma 7.1, we then obtain unweighted graphs G and G' such that

$$\text{PerfMatch}(H) = \text{PerfMatch}(G) - \text{PerfMatch}(G'). \tag{7.9}$$

Finally, combining (7.7)-(7.9), we see that G and G' are equipollent iff φ and φ' are. This concludes the hardness proof. □

It is even easier to show lower bounds for $\text{PerfMatch}_=^{0,1}$ under ETH than to prove its $\mathsf{C}_=\mathsf{P}$-completeness: Firstly, we may reduce from an arbitrary problem that admits a lower bound, such as SAT rather than $\mathsf{SAT}_=$. Secondly, we may use the more permissive notion of Turing reductions.

Theorem 7.6. *Assuming* ETH*, the problem* $\text{PerfMatch}_=^{0,1}$ *admits no* $2^{o(m)}$ *time algorithm on simple graphs with* m *edges.*

Proof. Assuming ETH, we cannot decide in time $2^{o(m)}$ whether a given 3-CNF formula φ on m clauses is satisfiable. With Lemma 7.3, we can construct unweighted graphs G_1 and G_2 on $\mathcal{O}(m)$ vertices and edges that are equipollent iff φ is *unsatisfiable*.

Hence, invoking an algorithm with running time $2^{o(m)}$ for $\text{PerfMatch}_=^{0,1}$ on G_1 and G_2, and flipping its answer, implies a $2^{o(m)}$ algorithm for determining whether φ is satisfiable, thus contradicting ETH. □

8. The block-interpolation framework

In this chapter, we show how to obtain tight lower bounds under #ETH by means of multivariate polynomial interpolation. Please recall the notion of a graph polynomial p, in particular Definition 1.22 on page 37, where the problems $\mathsf{Coeff}(p)$, $\mathsf{Eval}(p)$ and $\mathsf{Eval}_S(p)$ were defined. Furthermore, recall the reduction notion $\leq_{serf}$ from Definition 1.15 on page 34.

We consider all three problems above to be parameterized by the number of edges $m = |E(G)|$ of the input graph G. Then, for a general class of univariate graph polynomials p, we show that, for fixed $\xi \in \mathbb{Q}$,

$$\mathsf{Coeff}(p) \leq_{serf} \mathsf{Eval}_\xi(p). \tag{8.1}$$

This is useful due to the following reasons:

1. Many counting problems can be expressed as evaluation problems $\mathsf{Eval}_\xi(p)$ for adequate graph polynomials p and points ξ. For instance, the Tutte polynomial from Section 1.3.3 provides an abundance of such examples.

2. Many classical #P-hardness proofs for $\mathsf{Eval}_\xi(p)$ first establish #P-hardness for $\mathsf{Coeff}(p)$, and then reduce this to the evaluation problem by some form of interpolation. In many cases, the classical #P-hardness proof for $\mathsf{Coeff}(p)$ already yields a tight lower bound under #ETH, and in such cases, our technique allows to transfer this lower bound to $\mathsf{Eval}_\xi(p)$.

8.1. Setting up the framework

In Section 8.1.1, we first describe the "format" required from a univariate graph polynomial p for our framework to apply. Then we show in Section 8.1.2 how to reduce

$$\mathsf{Coeff}(p) \leq_{serf} \mathsf{Eval}_S(\mathbf{p}), \tag{8.2}$$

where $\mathbf{p}$ is a certain "multivariate version" of p, and the size of $S \subseteq \mathbb{Q}$ depends only on the running time parameter ϵ in the sub-exponential reduction family, but not on the size of the input graph. In Section 8.1.3, we then show how to reduce

$$\mathsf{Eval}_S(\mathbf{p}) \leq_p^{lin} \mathsf{Eval}_\xi(p) \tag{8.3}$$

by means of gadget families, provided that these families exist. The reductions (8.2) and (8.3) together will imply (8.1).

8.1.1. Admissible graph polynomials

Our framework applies to all graph polynomials that admit a canonical multivariate generalization. More specifically, we call a graph polynomial p *subset-admissible* if it can be described by a pair (χ, ω) consisting of a *sieving function* χ which "sieves out" the structures counted by p, and a *weight selector* ω which assigns a specific kind of weight to each of these structures.

Definition 8.1. Let $\mathcal{G}$ denote the set of all unweighted simple graphs, and let $\mathcal{F} = \mathcal{V} \cup \mathcal{E}$ denote the set of all vertices and edges of graphs in $\mathcal{G}$. Let χ and ω be functions

$$\begin{aligned} \chi &: \mathcal{G} \times 2^{\mathcal{F}} \to \mathbb{Q}, \\ \omega &: \mathcal{G} \times 2^{\mathcal{F}} \to 2^{\mathcal{F}}. \end{aligned}$$

We call χ a *sieve function,* we call ω a *weight selector*, and we say that (χ, ω) *induce* the graph polynomial $p : \mathcal{G} \to \mathbb{Q}[x]$ defined by

$$p_{\chi,\omega}(G; x) = \sum_{A \subseteq V(G) \cup E(G)} \chi(G, A) \cdot x^{|\omega(G,A)|}. \tag{8.4}$$

We call p *subset-admissible* if it is induced by some pair (χ, ω) as above.

The *sieve function* χ obtains its name from its ability to assign zero value to unwanted subsets, and ω is dubbed a *weight selector* because it selects vertices and edges, each of which then contributes a multiplicative weight of x. Note that, if p is subset-admissble, then the functions χ, ω that induce p suffice to specify p completely. Please also note that the expression (8.4) is similar to that seen in the context of MSOL-definable graph polynomials in Theorem 1.44. In particular, any univariate graph polynomial that is defined by an MSOL-formula as in Theorem 1.44 is also subset-admissible.

Example 8.2 (Matching polynomials)**.** The polynomial μ is induced by

$$\begin{aligned} \chi &: (G, A) \mapsto [A \in \mathcal{M}[G]], \\ \omega &: (G, A) \mapsto \operatorname{usat}(G, A). \end{aligned}$$

This holds because

$$p_{\chi,\omega}(G) = \sum_{A \subseteq V(G) \cup E(G)} \chi(G, A) \cdot x^{|\omega(G,A)|} = \sum_{A \in \mathcal{M}[G]} x^{|\operatorname{usat}(G,A)|} = \mu(G).$$

The edge-generating matching polynomial M can be obtained with $\omega(G, A) = A$. □

For our next example, we would like to show that the Tutte polynomial from Section 1.3.3 is subset-admissible, but this fails for syntactic reasons: According to Definition 8.1, only

univariate polynomials can be admissible. To apply our framework to the Tutte polynomial, we use its random-cluster formulation Z from Section 1.3.3 and consider restrictions $Z_q := Z(q, \cdot)$ for fixed $q \in \mathbb{Q}$. For $q = 0$, we consider $Z'(0, \cdot)$ rather than $Z(0, \cdot)$. These are typical steps used in #P-hardness results, see also [DHM+14].

Example 8.3 (Tutte polynomial)**.** Given a graph $G = (V, E)$ and $A \subseteq E$, let $k(G, A)$ denote the number of connected components of (V, A), including isolated vertices. For fixed $q \in \mathbb{Q}\backslash\{0\}$, the polynomial $Z_q := Z(q, \cdot)$ is induced by

$$\begin{aligned} \chi: &\quad (G, A) \mapsto q^{k(G,A)}, \\ \omega: &\quad (G, A) \mapsto A. \end{aligned}$$

Likewise, we can show that $Z_0 := Z'(q, \cdot)$ is subset-admissible. We stress again that Z_q for $q \in \mathbb{Q}$ are univariate polynomials, where $q \in \mathbb{Q}$ is considered fixed.

Every subset-admissible graph polynomial $p_{\chi,\omega}$ admits a canonical *multivariate generalization* $\mathbf{p}_{\chi,\omega}$, which we define in the following, and which will later enable our use of multivariate interpolation.

Definition 8.4. Let $\mathbf{x} = \{x_a \mid a \in \mathcal{F}\}$ be a set of indeterminates, where $\mathcal{F}$ denotes the set of all vertices and edges of graphs. Given functions χ and ω as in Definition 8.1, we define the multivariate generalization $\mathbf{p}_{\chi,\omega}$ of $p_{\chi,\omega}$ as

$$\mathbf{p}_{\chi,\omega}(G; \mathbf{x}) = \sum_{A \subseteq V(G) \cup E(G)} \chi(G, A) \cdot \prod_{a \in \omega(G,A)} x_a. \tag{8.5}$$

Please note that only finitely many indeterminates from $\mathbf{x}$ are present in $\mathbf{p}_{\chi,\omega}(G)$ for any (finite) graph G.

Compare (8.5) to the univariate version defined in (8.4): It is clear that $\mathbf{p}_{\chi,\omega}$ coincides with $p_{\chi,\omega}$ when substituting $x_a \leftarrow x$ for all $a \in \mathcal{F}$. Note also that $\mathbf{p}$ is multilinear by definition. Such multivariate generalizations are known, e.g., for the Tutte polynomial [Sok05], and we have also seen a general expression like (8.5) in the context of MSOL-definable polynomials in Theorem 1.44.

As seen for PerfMatch and MatchSum in Section 1.3, rather than evaluating $\mathbf{p}_{\chi,\omega}(G; \xi)$ for unweighted graphs G and tuples ξ, we consider $\mathbf{p}_{\chi,\omega}(G')$ for graphs G' with weights on edges and vertices. In fact, in all our applications, we can restrict ourselves to either vertex-weighted graphs or to edge-weighted graphs.

Example 8.5. The polynomial MatchSum is the multivariate generalization of the matching polynomial μ, with sieve function and weight selector defined as in Example 8.2.

The multivariate generalizations of $Z(q, \cdot)$ and $Z'(q, \cdot)$ yield the so-called *multivariate Tutte polynomial*

$$\mathbf{Z}(q, \cdot) = \sum_{A \subseteq E(G)} q^{k(G,A)} \prod_{e \in A} x_e$$

and its variant $\mathbf{Z}'(q,\cdot)$. The polynomial $\mathbf{Z}(q,\cdot)$ has already been considered in [Sok05].

Finally, consider the polynomial p induced by $\chi(G,A) = [A \in \mathcal{PM}[G]]$ and $\omega(G,A) = A$. This polynomial has at most one monomial, which has degree $\frac{|V(G)|}{2}$, and whose coefficient counts the perfect matchings in G. The multivariate generalization of p is the polynomial $\mathbf{p} = \text{PerfMatch}$. □

If p is a univariate subset-admissible polynomial and $\mathbf{p}$ is its multivariate generalization, then the following simple relation holds between the coefficients of p and $\mathbf{p}$:

Fact 8.6. *For any monomial θ, let $c(\theta)$ denote the coefficient of θ in $\mathbf{p}$. For $k \in \mathbb{N}$, let W_k denote the set of monomials in $\mathbf{p}$ with total degree k. Then, for all $k \in \mathbb{N}$, the coefficient of x^k in p is equal to $\sum_{\theta \in W_k} c(\theta)$.*

Proof. When substituting $x_a \leftarrow x$ for all $a \in \mathcal{F}$, we obtain p from $\mathbf{p}$, and the monomials transformed to x^k are precisely the members of W_k. This proves the claim. □

8.1.2. Using multivariate interpolation

Let $p = p_{\chi,\omega}$ be subset-admissible. For ease of presentation, we assume for now that $\omega : \mathcal{G} \times 2^{\mathcal{F}} \to 2^{\mathcal{E}}$, that is, ω maps only into edge-subsets rather than subsets of edges and vertices. The general case is shown identically and merely requires more notation.

We reduce $\mathsf{Coeff}(p) \leq_{serf} \mathsf{Eval}(\mathbf{p})$ by means of interpolation, where $\mathbf{p} = \mathbf{p}_{\chi,\omega}$ denotes the multivariate generalization of p. Recall that, in the univariate case, to obtain $p(G)$ for an m-edge graph G, we require evaluations $p(G;\xi)$ at $m+1$ distinct points $\xi \in \mathbb{Q}$. For the multivariate generalization $\mathbf{p}(G)$, we can use grid interpolation as shown in Lemma 1.38: Since $\mathbf{p}(G)$ is multilinear, this requires the evaluations of $\mathbf{p}(G;\xi)$ on a grid with two distinct values per coordinate, say $\Xi = [2]^m$. By Fact 8.6, the coefficients of p can be obtained from those of $\mathbf{p}$, so we could in principle interpolate $\mathbf{p}$ to recover p.

While this detour seems extremely wasteful due to its 2^m (rather than $m+1$) incurred evaluations, it yields the following reward: For each variable x_e in $\mathbf{p}$, grid interpolation only requires us to substitute *two* distinct values/weights into x_e, whereas univariate interpolation on p requires $m+1$ distinct substitutions to its only variable x. This small number of involved weights will be very useful, since each such weight is later simulated by a certain gadget at e, as we have seen, e.g., in Lemma 1.29 for PerfMatch. If only two weights are to be simulated, then we require only two fixed gadgets, whose sizes are trivially bounded by $\mathcal{O}(1)$.

However, to interpolate $\mathbf{p}$, we still need the prohibitively large number of 2^m evaluations. To overcome this, we trade off the number of evaluations with the numbers of distinct values that need to be simulated at each edge, and thus, with the size of the gadgets ultimately required.

Lemma 8.7. *Let p be subset-admissible, with multivariate generalization $\mathbf{p}$, and let $W = (w_0, w_1, \ldots)$ be an infinite recursively enumerable sequence of distinct numbers in $\mathbb{Q}$.*

Consider $\mathsf{Coeff}(p)$ and $\mathsf{Eval}_W(\mathbf{p})$ to be parameterized by the number of edges of the input graph G. Then $\mathsf{Coeff}(p) \leq_{serf} \mathsf{Eval}_W(\mathbf{p})$ holds by a sub-exponential reduction family

	$p \in \mathbb{Q}[x]$	$\mathbf{p} \in \mathbb{Q}[\mathbf{x}]$	$\mathbf{q} \in \mathbb{Q}[\mathbf{y}]$
number of indeterminates	1	m	$t = \lceil m/d \rceil$
max. degree of each indeterminate	m	1	d
max. number of monomials	$m+1$	2^m	$(d+1)^t$

Table 8.1.: The polynomials p, $\mathbf{p}$ and $\mathbf{q}$ appearing in the proof of Theorem 8.7.

that, on input (G, ϵ), only asks queries of the form $\mathbf{p}(G')$ on graphs G' obtained from G by introducing edge-weights from $W_d = \{w_0, \ldots, w_d\}$, where $d = d(\epsilon)$ is computable and depends only on ϵ.

When invoking Lemma 8.7, the list W contains the weights that can be simulated by gadgets. Note that *any* such list W can be used if W is infinite and recursively enumerable. Furthermore, note that $\mathbf{p}$ is evaluated only on edge-weighted versions of G itself; properties such as the number of edges or bipartiteness are hence trivially preserved.

Proof of Lemma 8.7. Let $d \in \mathbb{N}$ be a parameter, to be chosen later depending on ϵ, and let $G = (V, E)$ be an m-edge graph for which we want to determine the coefficients of $p = p(G)$. Let the indeterminates of $\mathbf{p}$ be denoted by

$$\mathbf{x} = \{x_e \mid e \in E\}$$

and note that both p and $\mathbf{p}$ have maximum degree m by definition.

Partition E into $t := \lceil m/d \rceil$ sets $E_1, \ldots, E_t$ of size at most d each (which we call *blocks*), using an arbitrary equitable assignment of edges to blocks. Define new indeterminates

$$\mathbf{y} := \{y_1, \ldots, y_t\}$$

and a new multivariate polynomial $\mathbf{q} \in \mathbb{Q}[\mathbf{y}]$ by substituting $x_e \leftarrow y_i$ for all $i \in [t]$ and $e \in E_i$ in $\mathbf{p}$. We are working with three polynomials, namely p, $\mathbf{p}$ and $\mathbf{q}$, summarized in Table 8.1 for convenience. While the total degree of $\mathbf{q}$ is bounded by m, the degree of each indeterminate y_i in $\mathbf{q}$ is bounded by d, since each block contains at most d edges. Hence, the number of monomials in $\mathbf{q}$ is at most $(d+1)^t = 2^{d'm}$, with $d' = \mathcal{O}(\log(d)/d)$. Note that $d' \to 0$ as $d \to \infty$.

We will obtain the coefficients of $\mathbf{q}$ via interpolation, but first, let us observe that the coefficients of $\mathbf{q}$ allow to determine those of the univariate polynomial p. Write $c_p(k)$ for the coefficient of x^k in p, and write $c_{\mathbf{q}}(\theta)$ for the coefficient of the monomial θ in $\mathbf{q}$. Analogously to Fact 8.6, we have $c_p(k) = \sum_{\theta \in C_k} c_{\mathbf{q}}(\theta)$, where C_k for $k \in \mathbb{N}$ is the set of all monomials with total degree k in $\mathbf{q}$. This allows us to compute the coefficients of p from those of $\mathbf{q}$.

It remains to obtain the coefficients of $\mathbf{q}$. For this, recall the definition of W_d from the statement. We evaluate $\mathbf{q}$ on the grid $\Xi = (W_d)^t$ using the oracle for $\mathsf{Eval}(\mathbf{p})$: For each $\xi \in \Xi$, substitute $y_i \leftarrow \xi_i$ for all $i \in [t]$ to obtain an edge-weighted graph G_ξ that contains only weights from W_d, and for which we can thus compute $\mathbf{p}(G_\xi)$ by an oracle call.

Using $|\Xi| = (d+1)^t = 2^{d'm}$ oracle calls and grid interpolation via Lemma 1.38, we obtain all coefficients of $\mathbf{q}$ with $\mathcal{O}(2^{3d'm})$ arithmetic operations.[1] Since $d' \to 0$ as $d \to \infty$, we can pick d large enough such that $3d' \leq \epsilon$, and thus achieve running time $\mathcal{O}(2^{\epsilon m})$. No vertices or edges are added to G, so the parameter is not increased. □

8.1.3. Weight simulation

With Lemma 8.7, we obtain a sub-exponential reduction family from $\mathsf{Coeff}(p)$ on instances (G, ϵ) to $\mathsf{Eval}(\mathbf{p})$ on versions obtained from G by introducing $f(\epsilon)$ distinct edge-weights. For the full reduction, this latter problem must be reduced to $\mathsf{Eval}_\xi(p)$ for *fixed* $\xi \in \mathbb{Q}$. This may not work for all $\xi \in \mathbb{Q}$: For instance, $\mathsf{Eval}_0(I)$ for the independent-set polynomial I at 0 is trivial. We must hence impose conditions on ξ to enable this reduction.

Definition 8.8. Let p be subset-admissible, let $\xi \in \mathbb{Q}$ and

- let $W = (w_0, w_1, \ldots)$ be a sequence of pairwise distinct values in $\mathbb{Q}$,
- let $\mathcal{H} = (H_0, H_1, \ldots)$ be a sequence of *edge-gadgets*, which are triples (H, u, v) consisting of an unweighted graph H and *attachment vertices* $u, v \in V(H)$, and
- let $F : \mathcal{G} \times \mathbb{Q} \to \mathbb{Q} \setminus \{0\}$ be a polynomial-time computable function, which we call a *factor function*.

If G is edge-weighted with weights from W, let $T(G)$ be the unweighted graph obtained by replacing, for $i \in \mathbb{N}$, each edge $uv \in E(G)$ of weight w_i with a fresh copy of H_i by identifying u, v across G and H_i. We say that $(\mathcal{H}, F)$ *allows to reduce* $\mathsf{Eval}_W(\mathbf{p})$ *to* $\mathsf{Eval}_\xi(p)$ if the following holds: Whenever G is a graph with edge-weights from W, then

$$p(T(G); \xi) = F(G, \xi) \cdot \mathbf{p}(G). \tag{8.6}$$

The same definition applies to vertex-weighted graphs; here we use *vertex-gadgets*, which are pairs (H, v) with an attachment vertex $v \in V(H)$. Vertex-gadgets are inserted at a vertex $v \in V(G)$ by identifying v in H and G.

At the end of this subsection, we discuss some possible extensions of this definition. As a first example for the notions introduced in Definition 8.8, we consider (well-known) edge-gadgets for PerfMatch.

Example 8.9. Let p denote the polynomial from Example 8.5 with $\mathbf{p} = \mathrm{PerfMatch}$. Let $\mathcal{H} = (H_1, H_2, \ldots)$ be such that H_k for $k \in \mathbb{N}$ is the graph obtained by placing k parallel edges between u and v and then subdividing each edge twice, as in Lemma 1.29 on page 42.

[1] By definition of $\mathbf{p}$ and $\mathbf{q}$, each arithmetic operation involves numbers of size at most $2^m \cdot (\max W_d)^m$. This amounts to $m \cdot g(d)$ bits for a computable function g.

Let $\mathbb{N} = (1, 2, 3, \ldots)$ and let F denote the function that maps all inputs to 1. We have seen in Lemma 1.29 that $(\mathcal{H}, F)$ allows to reduce $\mathsf{Eval}_{\mathbb{N}}(\mathbf{p})$ to $\mathsf{Eval}_1(p)$.

By combining Lemma 8.7 and Definition 8.8, we obtain the wanted reduction from $\mathsf{Coeff}(p)$ to $\mathsf{Eval}_\xi(p)$ at fixed points $\xi \in \mathbb{Q}$. This will be the foundation for all lower bounds to be derived in the next section.

Theorem 8.10 (Block interpolation theorem)**.** *Let p be subset-admissible and let $\xi \in \mathbb{Q}$ be fixed. Assuming #ETH, the problem $\mathsf{Eval}_\xi(p)$ admits no $2^{o(m)}$ time algorithm on simple graphs with m edges, provided that the following two conditions hold:*

Coefficient hardness: *Assuming #ETH, the problem $\mathsf{Coeff}(p)$ admits no $2^{o(m)}$ time algorithm on simple graphs with m edges.*

Weight simulation: *There is a recursively enumerable sequence $W = (w_0, w_1, \ldots)$ of pairwise distinct weights, a sequence of gadgets $\mathcal{H} = (H_0, H_1, \ldots)$ and a factor function F such that $(\mathcal{H}, F)$ allows to reduce $\mathsf{Eval}_W(\mathbf{p})$ to $\mathsf{Eval}_\xi(p)$.*

Proof. We show that $\mathsf{Coeff}(p) \leq_{serf} \mathsf{Eval}_\xi(p)$, where we consider both problems to be parameterized by the number of edges of the input graph. Given $\epsilon > 0$ and a graph G with m edges, apply Lemma 8.7 to reduce $\mathsf{Coeff}(p)$ to $2^{\epsilon m}$ instances of $\mathsf{Eval}_S(\mathbf{p})$ on $S = \{w_0, \ldots, w_s\}$, where $s = s(\epsilon)$ is computable and depends only upon ϵ.

Let G' be an instance for $\mathsf{Eval}_S(\mathbf{p})$ obtained in this step, i.e., an edge-weighted graph with weights from S. By Lemma 8.7, we have $|E(G')| = m$. Since $(\mathcal{H}, F)$ allows to reduce $\mathsf{Eval}_W(\mathbf{p})$ to $\mathsf{Eval}_\xi(p)$, we can reduce the evaluation of $\mathbf{p}$ on G' to an instance of $\mathsf{Eval}_\xi(p)$ on the graph $T(G')$ from Definition 8.8. Let

$$b = \max_{i \in \{0,\ldots,s\}} |E(H_i)|$$

be the maximum number of edges used by edge-gadgets in $T(G')$, and observe that, for a fixed family $\mathcal{H}$, the number b depends only on s, which in turn depends only on ϵ. Then

$$|E(T(G'))| \leq bm,$$

with computable $b = b(\epsilon)$. This fulfills the requirements of a sub-exponential Turing reduction family. □

Let us remark some corollaries of the reduction shown above.

Remark 8.11. If the source instance G for $\mathsf{Coeff}(p)$ has maximum degree Δ, then the reduction images $T(G')$ obtained above feature maximum degree $\Delta \cdot f(\epsilon)$ for a computable function f. By suitable choice of $\mathcal{H}$, we can also ensure other properties on $T(G)$:

- If G is bipartite and all edge-gadgets $(H, u, v) \in \mathcal{H}$ can be 2-colored such that u and v receive different colors, then $T(G')$ is bipartite as well.

- If G is planar and all edge-gadgets $(H, u, v) \in \mathcal{H}$ admit planar drawings with u and v on their outer faces, then $T(G')$ is planar as well.

To conclude this subsection, we list several possible generalizations of Definition 8.8 and Theorem 8.10 that we did not include in our formulation, since they did not prove useful. For future research, it might however be a good idea to keep these extensions in mind.

1. We may extend Definition 8.8 to incorporate weight simulations that proceed non-locally, that is, by something less obvious than inserting local gadgets at edges.

2. In Lemma 8.7, we do not need to evaluate $\mathbf{p}$ on a grid W^t for a *fixed* coordinate set W. Instead, we could as well interpolate on a grid $W_1 \times \ldots \times W_t$, provided that each W_i is large enough and that the weights do not depend on the size of G.

3. We may also allow $2^{o(m)}$ time for the computation of the factor $F(G_w, \xi)$. Rather than allowing only a multiplicative factor, we could also allow an arbitrary function to be computed from $p(T(G); \xi)$ and the input.

8.2. Applications

In the following subsections, we apply block interpolation (Theorem 8.10) to obtain tight lower bounds of the type $2^{\Omega(m)}$ for a variety of counting problems, including the unweighted permanent, the matching polynomials and the Tutte polynomial.

8.2.1. Unweighted permanent

We prove Theorem 8.12, which claims a lower bound of $2^{\Omega(m)}$ for $\mathsf{Eval}_{0,1}(\mathrm{perm}) = \mathrm{perm}^{0,1}$ on graphs with m edges under #ETH.

Theorem 8.12 (restated from page 157)**.** *Unless* #ETH *fails, the problem* $\mathrm{perm}^{0,1}$ *does not admit an algorithm with running time* $2^{o(m)}$ *on simple graphs with* m *edges.*

Proof. We invoke Theorem 8.10 to show

$$\mathsf{Eval}_{-1,0,1}(\mathrm{perm}) \leq_{serf} \mathsf{Eval}_{0,1}(\mathrm{perm}).$$

Let G be a graph on m edges, with edge-weights from $\{-1, 1\}$, and let $E_{-1}(G)$ denote the set of edges with weight -1 in G. As in Lemma 1.36 on page 45, where we removed the edge-weight -1 by univariate interpolation, we define a polynomial $p(G)$ with

$$p(G; -1) = \mathrm{perm}(G).$$

To this end, let $p = p_{\chi,\omega}$ be the polynomial induced by

$$\begin{aligned} \chi(G, A) &= [A \in \mathcal{PM}[G]], \\ \omega(G, A) &= A \cap E_{-1}(G). \end{aligned}$$

Since knowledge of the coefficients of $p(G)$ allows to evaluate $p(G;-1)$, we obtain from [DHM$^+$14, Theorem 1.3] that $\mathsf{Coeff}(p)$ admits no $2^{o(m)}$ time algorithm under #ETH. Hence the requirement of coefficient hardness from Theorem 8.10 is fulfilled.[2]

To check the requirement of weight simulation in Theorem 8.10, recall the pair $(\mathcal{H}, F)$ from Example 8.9 that allows to reduce $\mathsf{Eval}_{\mathbb{N}}(\text{perm})$ to $\mathsf{Eval}_1(p)$. This pair in particular allows to reduce $\mathsf{Eval}_{\mathbb{N}}(\mathbf{p})$ to $\mathsf{Eval}_1(p)$. By Remark 8.11, the reduction images $T(G)$ constructed by Theorem 8.10 are bipartite as well. □

Using a reduction to decrease the maximum degree to 3, together with reductions via line graphs, we can collect a series of corollaries for other counting problems. In the following, we say that G is a line graph if G is the line graph of some graph, as defined on page 39.

Corollary 8.13. *Assuming #ETH, the following cannot be solved in time $2^{o(n)}$:*

1. *The problem $\mathsf{Eval}_{0,1}(\text{perm})$ on graphs G with n vertices and $\Delta(G) \leq 3$.*
2. *Counting maximum-cardinality independent sets (or equivalently, minimum-cardinality vertex covers), even in line graphs G with n vertices and $\Delta(G) \leq 4$.*
3. *Counting minimum-weight satisfying assignments to monotone 2-CNF formulas on n variables, even if every variable appears in at most four clauses.*

Proof. For the first statement, we use a reduction from the permanent on general graphs G to graphs G' with $\Delta(G) \leq 3$, shown in [DL92]. If G has n vertices and m edges, then the graph G' constructed by this reduction has $\mathcal{O}(n+m)$ vertices and edges and maximum degree 3. In fact, we could also prove this statement by repeated application of Fact 3.4 on page 88 to "split" vertices of degree greater than 3.

For the second statement, if a graph H has m edges, then its line graph $L(H)$ has m vertices and maximum degree $2(\Delta(H)-1)$. The set $\mathcal{PM}[G]$ corresponds bijectively to the independent sets of size $\frac{|V(G)|}{2}$ in $L(G)$. These are the maximum-cardinality independent sets in $L(G)$.[3] The maximum-cardinality independent sets of any graph stand in bijection with its minimum vertex covers via complementation. We thus obtain the second statement by reduction from $\mathsf{Eval}_{0,1}(\text{perm})$ on graphs of maximum degree 3.

For the third statement, observe that the minimum vertex covers of a graph $H = (V, E)$ correspond bijectively to the minimum-weight satisfying assignments of the following monotone 2-CNF formula φ: Create a variable x_v for each vertex $v \in V$, and a clause $(x_u \vee x_v)$ for each edge $uv \in E$. This is noted also in [Vad01]. □

8.2.2. Matching and independent set polynomials

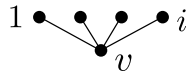
We prove Theorem 8.15, a tight lower bound for $\mathsf{Eval}_\xi(\mu)$ at fixed $\xi \in \mathbb{Q}$, by invoking Theorem 8.10. The perfect matchings of G are counted by the coefficient of x^0 in $\mu(G)$,

[2] Rather than using [DHM$^+$14] for this step, we could as well fine-tune our reduction in Corollary 2.31 to yield bipartite graphs, and then obtain a polynomial-time weakly parsimonious reduction from 3-#SAT to $\mathsf{Eval}_{-1,0,1}(\text{perm})$ that incurs only linear blowup.

[3] That is, unless G has no perfect matching. However, we may assume that all graphs considered *have* perfect matchings, as they would otherwise not be required in a reduction.

so $\mathsf{Coeff}(\mu)$ and $\mathsf{Eval}_0(\mu)$ have the same lower bound as $\mathsf{Eval}_{0,1}(\text{perm})$. This settles the requirement of coefficient hardness in Theorem 8.10. In the following, we show that μ also admits weight simulation.

Lemma 8.14. *Let $\mathcal{H} = (H_i)_{i\in\mathbb{N}}$ be the following family of vertex-gadgets that is constructed as follows: The graph H_i contains one attachment vertex v, adjacent to an independent set of i vertices.*

H_i : 

Consider $\xi \in \mathbb{Q} \setminus \{0\}$ to be fixed. Let $W = (w_t)_{t\in\mathbb{N}}$ with $w_t = 1 + \frac{t}{\xi^2}$ for $t \in \mathbb{N}$. Given a graph G with vertex-weights from W, let a_t for $t \in \mathbb{N}$ denote the number of vertices of weight w_t in G. We define

$$F(G, \xi) = \prod_{t\in\mathbb{N}} \xi^{t\cdot a_t}.$$

Then $(\mathcal{H}, F)$ allows to reduce $\mathsf{Eval}_W(\text{MatchSum})$ to $\mathsf{Eval}_\xi(\mu)$.

Proof. Recall the graph transformation $T(G)$ from Definition 8.8: At every vertex of weight w_t, for $t \in \mathbb{N}$, we attach the gadget H_t. To show that $(\mathcal{H}, F)$ indeed satisfies Definition 8.8, we need to show that

$$\mu(T(G), \xi) = F(G, \xi) \cdot \text{MatchSum}(G). \tag{8.7}$$

To see this, observe that every matching M in G can be extended locally at vertices $v \in V(G)$ by edges of vertex-gadgets to obtain a matching in $T(G)$.[4] Let $v \in V(G)$ be a vertex of weight w_t for $t \in \mathbb{N}$. If $v \notin \text{usat}(G, M)$, then M cannot be extended at the vertex v, and v incurs the factor ξ^t in $\mu(T(G))$. If $v \in \text{usat}(G, M)$, then we can choose not to extend v, or we may choose to extend by one of the t edges of H_t, so we obtain a factor of $\xi^t + t\xi^{t-2} = \xi^t(1 + \frac{t}{\xi^2})$. This establishes (8.7), and hence the lemma. □

From this, we can conclude the lower bound for $\mathsf{Eval}_\xi(\mu)$ via block interpolation.

Theorem 8.15 (restated from page 158)**.** *Unless #ETH fails, the problem of evaluating the matching polynomial $\mu(G; \xi)$ for any fixed $\xi \in \mathbb{Q}$ does not admit an algorithm with running time $2^{o(m)}$ on simple graphs G with m edges, even when the maximum degree $\Delta(G)$ may be assumed to be a constant.*

Proof. By Corollary 8.13, the problem $\mathsf{Coeff}(\mu)$ admits no $2^{o(m)}$ algorithm unless #ETH fails, even on graphs with maximum degree 3. This settles the requirement of coefficient hardness in Theorem 8.10, even on graphs of maximum degree 3.

In Lemma 8.14, we have seen that μ admits weight simulations. By Remark 8.11 and the reduction from $\mathsf{Coeff}(\mu)$ on graphs of maximum degree 3, the queries issued by Theorem 8.10 have maximum degree $\mathcal{O}(1)$, which implies the degree bound in Theorem 8.15. □

[4] Note that this is similar to the rakes from Section 4.2.

Remark 8.16. We can easily verify that the block interpolation framework also allows us to prove the same lower bound for $\mu(G;\xi)$ even when $\xi \in \mathbb{C}$ is a complex number with $\xi = \sqrt{c}$ for $c \in \mathbb{Q}$. Note that we may assume that G has an even number of vertices; in this case, we can compute $\mu(G;\xi)$ over the rational numbers: Every matching in G has an even number of unmatched vertices, and thus only even powers of ξ appear in $\mu(G;\xi)$.

As in Corollary 8.13, we can easily obtain corollaries for the independent set polynomial I, defined in Section 1.3, and for monotone 2-SAT, improving upon [Hof10, DHM$^+$14].

Corollary 8.17. *Assuming* #ETH*, the following cannot be solved in time* $2^{o(n)}$*:*

1. *Evaluating the independent-set polynomial* $I(G;\xi)$ *on line graphs with* n *vertices and maximum degree* $\mathcal{O}(1)$*, for any* $\xi \in \mathbb{Q} \setminus \{0\}$*. This holds especially at* $\xi = 1$*, which amounts to counting independent sets (or equivalently, vertex covers).*

2. *Counting satisfying assignments to monotone 2-CNF formulas on* n *variables, even if every variable appears in at most* $\mathcal{O}(1)$ *clauses.*

Proof. We prove the first statement: If G has m edges and $\Delta(G) = \mathcal{O}(1)$, then $L(G)$ has m vertices and $\Delta(L(G)) = \mathcal{O}(1)$. Recall that $M(G) = I(L(G))$, where M is the edge-generating matching polynomial, and that

$$\mu(G;\xi) = \xi^{|V(G)|} \cdot M(G;\xi^{-2}) = \xi^{|V(G)|} \cdot I(L(G);\xi^{-2}),$$

where the first identity is (1.4) on page 37. Hence, for fixed $\xi \neq 0$, an algorithm for $\mathsf{Eval}_\xi(I)$ on line graphs implies one for $\mathsf{Eval}_{\xi'}(\mu)$ on general graphs with $\xi' = \sqrt{\xi^{-1}}$, but this is ruled out by Theorem 8.15 and Remark 8.16.

For the second statement, recall that independent sets and vertex covers stand in bijection. We then reduce from counting vertex covers as in Corollary 8.13. □

8.2.3. Tutte polynomial

To prove a lower bound for the evaluation of the Tutte polynomial T, we use univariate restrictions of the random-cluster model Z. As discussed in Example 8.3, we write $Z_q := Z(q,\cdot)$ for fixed $q \in \mathbb{Q} \setminus \{0\}$ and $Z_0 := Z'(0,\cdot)$, where Z and Z' are defined as in Section 8.2.3. Then we use block interpolation to prove lower bounds for $\mathsf{Eval}_w(Z_q)$ at fixed $w \in \mathbb{Q}$. As in the previous examples, we require a lower bound for $\mathsf{Coeff}(Z_q)$, which was shown in Propositions 4.1 and 4.3 of [DHM$^+$14].

Lemma 8.18 ([DHM$^+$14])**.** *Assuming* #ETH*, the problem* $\mathsf{Coeff}(Z_q)$ *for* $q \in \mathbb{Q} \setminus \{1\}$ *cannot be solved in time* $2^{o(m)}$ *on graphs with* m *edges.*

In fact, we could also use block interpolation to simplify the proof of Lemma 8.18 from [DHM$^+$14] by performing an interpolation step that needed to be circumvented by the authors with some tricks. However, since Lemma 8.18 was explicitly shown in [DHM$^+$14], we omit the self-contained proof that would still require some arguments which are very

specific to the Tutte polynomial. Note that the case $q = 1$ is left uncovered by this lemma; we consequently cannot prove lower bounds at $q = 1$.

In [DHM+14], the problem $\mathsf{Coeff}(Z_q)$ is then reduced to the unweighted evaluation problem via *Theta graphs* and *wumps*, families of edge-gadgets that incur only $\mathcal{O}(\log^c n)$ blowup. This economical (but *still* not constant) factor however requires a somewhat involved analysis. Using block interpolation, we can instead use mere paths as gadgets, and hence perform *stretching*, a classical weight simulation technique for the Tutte polynomial [JVW90, GJ08]. Please recall the definition of the multivariate Tutte polynomial $\mathbf{Z_q}$ from Example 8.5.

Lemma 8.19. *For $k \in \mathbb{N}$, let P_k denote the path on k edges with distinguished start/end vertices $u, v \in V(P_k)$, and let $\mathcal{P} = (P_1, P_2, \ldots)$. Let $w, q \in \mathbb{Q}$ be fixed with $w \neq 0$ and $q \notin \{1, -w, -2w\}$. Then there is an infinite sequence of pairwise distinct weights W and a factor function F such that $(\mathcal{P}, F)$ allows to reduce $\mathsf{Eval}_W(\mathbf{Z_q})$ to $\mathsf{Eval}_w(Z_q)$.*

Proof. We have to distinguish whether $q = 0$ or $q \neq 0$ holds, and we obtain different weights and factor functions in the different cases.

If $q = 0$, we define $W = (w_k)_{k\in\mathbb{N}}$ with the pairwise distinct weights $w_k = \frac{w}{k}$ for $k \in \mathbb{N}$. Given a graph G with edge-weights from W, let $a_k(G)$ for $k \in \mathbb{N}$ denote the number of edges in G with weight w_k, and define

$$F(G) = \prod_{k\in\mathbb{N}} (kw^{k-1})^{\alpha_k(G)}.$$

Then it is known that $(\mathcal{P}, F)$ allows to reduce $\mathsf{Eval}_W(\mathbf{Z_0})$ to $\mathsf{Eval}_w(Z_0)$, as shown in [DHM+14, Corollary 6.7] and [GJ08].

If $q \neq 0$, then the family of paths realizes different weights and requires a different factor function. Define $W = (w_k)_{k\in\mathbb{N}}$ with

$$w_k = \frac{q}{(1 + \frac{q}{w})^k - 1}$$

and observe that these weights are pairwise distinct provided that $1 + \frac{q}{w} \notin \{-1, 0, 1\}$, which holds by $q \neq 0$ and the prerequisites of the proposition. Given a graph G with edge-weights from W, let $a_k(G)$ for $k \in \mathbb{N}$ denote the number of edges in G with weight w_k and define

$$F(G) = q^{-|E(G)|} \prod_{k\in\mathbb{N}} ((q + w)^k - w^k)^{a_k(G)}.$$

Then it is shown in [DHM+14, Lemma 6.2] and [Sok05, Propositions 2.2 and 2.3] that $(\mathcal{P}, F)$ allows to reduce $\mathbf{Z_q}$ on W to $Z_q(w)$. □

By combining Lemma 8.18 for the requirement of coefficient hardness and Lemma 8.19 for weight simulations, we can invoke Theorem 8.10 and obtain:

Lemma 8.20. *Let $w \neq 0$ and $q \notin \{1, -w, -2w\}$. Assuming #ETH, the problem $\mathsf{Eval}_w(Z_q)$ admits no $2^{o(m)}$ algorithm on graphs with m edges.*

Using the substitution (1.14) on page 44 that maps $Z(\cdot,\cdot)$ to the classical parameterization $T(\cdot,\cdot)$ of the Tutte polynomial, and using (1.16) if $q = 0$, we obtain Theorem 8.21.

Theorem 8.21 (restated from page 158)**.** *Unless* #ETH *fails, the Tutte polynomial* $T(x,y)$ *for fixed* $(x,y) \in \mathbb{Q}^2$ *cannot be evaluated in time* $2^{o(m)}$ *on simple graphs with* m *edges, provided that*

- $y \notin \{0,1\}$, and
- $(x,y) \notin \{(1,1),(-1,-1),(0,-1),(-1,0)\}$, and
- $(x-1)(y-1) \neq 1$.

The points on the lines given by $y \in \{0,1\}$ are not covered by Theorem 8.21, and they actually do not fit into the block interpolation framework: The restriction $T(\cdot,0)$ specializes to the chromatic polynomial, for which it is unclear how to even define a weighted version. The known #P-hardness proof [Lin86], used in [DHM$^+$14], yields a lower bound of $2^{\Omega(n)}$ on graphs with n vertices, but $\Omega(n^2)$ edges. For $T(\cdot,1)$, we cannot even prove a lower bound for the coefficient problem.

Epilogue

Open problems

In each part of this thesis, we leave various problems open for future research. Some of these are listed in the following, sorted by perceived importance within each part:

Part I

1. We have found ways to evaluate PerfMatch in XP-time on *almost* all individual substructures arising from the Graph Structure Theorem:
   - For bounded-genus graphs, we can proceed in time $4^{\gamma}n^{\mathcal{O}(1)}$ via Theorem 1.28.
   - For bounded-genus graphs with a bounded number of apices, we can apply brute-force in time $4^{\gamma}n^{k+\mathcal{O}(1)}$, and by Theorem 4.1, we cannot hope for an fpt-algorithm in the number of apices.
   - It can be verified that clique-sums of bounded-genus graphs with apices can also be evaluated in XP-time. We have not included this in the thesis.

   However, we failed to handle vortices. Even an algorithm for computing PerfMatch for a planar graph with one single vortex would greatly contribute towards an XP-algorithm for PerfMatch/hadw.

2. As we observed in Theorem 4.21, the #W[1]-hardness result for k-apex graphs does no longer hold if each apex can see only a bounded number of faces. This is the situation in torsos of graphs excluding a fixed graph from $\mathcal{A}1$, the class of 1-apex graphs. If we could solve the generalized vortices arising from [DHK09] in fpt-time, then an fpt-algorithm for PerfMatch/hadw$_{\mathcal{A}1}$ seems in reach.

Part II

1. Find tight lower bounds under #ETH or #SETH for the subgraph counting problems studied in Part II, in particular for the problem of counting k-matchings. That is, find or rule out an $f(k)n^{o(k)}$ algorithm for counting k-matchings under #ETH. A lower bound would carry over to various other counting problems. Note that the current lower bounds, which use the lower bound for #PartitionedSub from [Mar10], can only rule out $f(k)n^{o(k/\log k)}$ algorithms. However, that paper gives a lower bound for the *decision* version PartitionedSub, and it might be easier to obtain lower bounds for the counting version #PartitionedSub.

2. Extend the machinery of combined colorful signatures used in Section 5.2.2 to yield #W[1]-hardness for (edge-colorful) Holant problems other than counting (edge-colorful) k-matchings.

3. Prove a dichotomy theorem for the *edge-colorful* version of the subgraph counting problem $\#\mathsf{Sub}(\mathcal{H})$ on graph classes $\mathcal{H}$. Note that the complexity of the vertex-colorful version is completely determined by the maximum treewidth of $\mathcal{H}$, but an equally simple characterization does not seem to apply to the edge-colorful variant. For instance, we have observed that counting edge-colorful matchings is $\#\mathsf{W}[1]$-complete, but we can also give an fpt-algorithm for counting edge-colorful copies of a matching that has one additional vertex adjacent to every edge.

Part III

1. The block interpolation framework can be used to derive further lower bounds under #ETH. To avoid a long list of semi-interesting individual results, it would be nice to derive a meta-theorem from it, similar to the known sweeping #P-dichotomy theorems for Holant or #CSP problems. Can we uniformly prove lower bounds under #ETH for the problems that lie on the "hard" side of such a dichotomy?

2. Show that an algorithm with running time $(4-\epsilon)^{\gamma} \cdot n^{\mathcal{O}(1)}$ for evaluating PerfMatch on graphs of genus γ would refute the strong exponential time hypothesis #SETH, or find a faster algorithm. Note that an algorithm with running time $4^{\gamma} \cdot n^{\mathcal{O}(1)}$ exists by Theorem 1.28. Of course, a lower bound of $c^n n^{\mathcal{O}(1)}$ for PerfMatch on n-vertex graphs would also be highly interesting.

3. Give tight lower bounds for the remaining points of the Tutte polynomial that were not covered by our results.

Bibliography

[ACP87] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.

[Agr06] Manindra Agrawal. Determinant versus permanent. In *Proceedings of the 25th International Congress of Mathematicians, ICM 2006*, volume 3, pages 985–997, 2006.

[ALS91] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12(2):308–340, 1991.

[AR02] Vikraman Arvind and Venkatesh Raman. Approximation algorithms for some parameterized counting problems. In *ISAAC*, pages 453–464, 2002.

[Asa85] Takao Asano. An approach to the subgraph homeomorphism problem. *Theor. Comp. Sci.*, 38(0):249–267, 1985.

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[BC11] Markus Bläser and Radu Curticapean. The complexity of the cover polynomials for planar graphs of bounded degree. In *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, pages 96–107, 2011.

[BC12] Markus Bläser and Radu Curticapean. Weighted counting of k-matchings is #W[1]-hard. In *IPEC*, pages 171–181, 2012.

[BD07] Markus Bläser and Holger Dell. Complexity of the cover polynomial. In *ICALP 2007*, pages 801–812, 2007.

[BDH15] Andreas Björklund, Holger Dell, and Thore Husfeldt. The parity of set systems under random restrictions with applications to exponential time problems. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 231–242, 2015.

[BH74] James R. Bunch and John E. Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, 28(125):pp. 231–236, 1974.

[Big94] Norman Biggs. *Algebraic Graph Theory*. Cambridge University Press, 2nd edition, 1994.

[Bir12] George D. Birkhoff. A determinant formula for the number of ways of coloring a map. *Annals of Mathematics*, 14(1/4):pp. 42–46, 1912.

[Bjö12] Andreas Björklund. Counting perfect matchings as fast as Ryser. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 914–921, 2012.

[BMT14] Jacob D. Biamonte, Jason Morton, and Jacob W. Turner. Tensor Network Contractions for #SAT. *ArXiv e-prints*, May 2014.

[Bod96] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, December 1996.

[Bra08] Sergey Bravyi. Contraction of matchgate tensor networks on non-planar graphs. *ArXiv e-prints*, January 2008.

[Bul13] Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. *J. ACM*, 60(5):34, 2013.

[BW91] Graham Brightwell and Peter Winkler. Counting linear extensions is #P-complete. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 175–181, 1991.

[CC12] Jin-Yi Cai and Xi Chen. Complexity of counting CSP with complex weights. In *STOC 2012*, pages 909–920, 2012.

[CC14] Chandra Chekuri and Julia Chuzhoy. Polynomial bounds for the Grid-Minor Theorem. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 60–69, 2014.

[CCF+05] Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized np-hard problems. *Inf. Comput.*, 201(2):216–231, 2005.

[CDG+15] Xi Chen, Martin E. Dyer, Leslie Ann Goldberg, Mark Jerrum, Pinyan Lu, Colin McQuillan, and David Richerby. The complexity of approximating conservative counting csps. *J. Comput. Syst. Sci.*, 81(1):311–329, 2015.

[CE13] Erin W. Chambers and David Eppstein. Flows in one-crossing-minor-free graphs. *J. Graph Algorithms Appl.*, 17(3):201–220, 2013.

[CG14] Jin-Yi Cai and Aaron Gorenstein. Matchgates revisited. *Theory of Computing*, 10:167–197, 2014.

[CH90] Jin-Yi Cai and Lane A. Hemachandra. On the power of parity polynomial time. *Mathematical Systems Theory*, 23(2):95–106, 1990.

[Che10] Mike Chen. The complexity of checking whether two DAG have the same number of topological sorts. http://cstheory.stackexchange.com/questions/3105, November 2010.

[CHKX06] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346 – 1367, 2006.

[CHL10] Jin-yi Cai, Sangxia Huang, and Pinyan Lu. From Holant to #CSP and back: Dichotomy for Holant^c problems. In *Algorithms and Computation - 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I*, pages 253–265, 2010.

[CL07] Jin-Yi Cai and Pinyan Lu. Holographic algorithms: From art to science. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, STOC '07, pages 401–410, New York, NY, USA, 2007. ACM.

[CLX08] Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Holographic algorithms by Fibonacci gates and holographic reductions for hardness. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 644–653. IEEE Computer Society, 2008.

[CLX09a] Jin-Yi Cai, Pinyan Lu, and Mingji Xia. A computational proof of complexity of some restricted counting problems. In *TAMC 2009*, pages 138–149, 2009.

[CLX09b] Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Holant problems and counting CSP. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 715–724, New York, NY, USA, 2009. ACM.

[CLX10] Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Holographic algorithms with matchgates capture precisely tractable planar #CSP. *CoRR*, abs/1008.0683, 2010. Also appeared in FOCS 2010.

[CLX11] Jin-yi Cai, Pinyan Lu, and Mingji Xia. Dichotomy for Holant* problems of Boolean domain. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1714–1728, 2011.

[CM93] Bruno Courcelle and Mohamed Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theor. Comput. Sci.*, 109(1&2):49–82, 1993.

[CM14] Radu Curticapean and Dániel Marx. Complexity of counting subgraphs: Only the boundedness of the vertex-cover number counts. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 130–139, 2014.

[CMR01] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics*, 108(1-2):23–52, 2001.

[CTW08] Yijia Chen, Marc Thurley, and Mark Weyer. Understanding the complexity of induced subgraph isomorphisms. In *ICALP (1)*, pages 587–596, 2008.

[Cur13] Radu Curticapean. Counting matchings of size k is #W[1]-hard. In *ICALP (1)*, pages 352–363, 2013.

[Cur14] Radu Curticapean. Counting perfect matchings in graphs that exclude a single-crossing minor. *CoRR*, abs/1406.4056, 2014.

[Cur15] Radu Curticapean. Block interpolation: A framework for tight exponential-time counting complexity. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 380–392, 2015.

[CX] Radu Curticapean and Mingji Xia. Parameterizing the permanent: Genus, apices, minors, evaluation mod 2^k. In *56th IEEE Annual Symposium on Foundations of Computer Science, FOCS. To appear.*

[DF95] Rodney G. Downey and Michael R. Fellows. Parameterized computational feasibility. In *Feasible Mathematics II*, volume 13 of *Progress in Computer Science and Applied Logic*, pages 219–244. Birkhäuser Boston, 1995.

[DF99] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer, 1999.

[DGP07] Martin Dyer, Leslie Ann Goldberg, and Mike Paterson. On counting homomorphisms to directed acyclic graphs. *J. ACM*, 54, December 2007.

[DHK09] Erik D. Demaine, MohammadTaghi Hajiaghayi, and Ken-ichi Kawarabayashi. Approximation algorithms via structural results for apex-minor-free graphs. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, pages 316–327, 2009.

[DHM+14] Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlen. Exponential time complexity of the permanent and the Tutte polynomial. *ACM Transactions on Algorithms*, 10(4):21, 2014.

[DHN+04] E. Demaine, M. Hajiaghayi, N. Nishimura, P. Ragde, and D. Thilikos. Approximation algorithms for classes of graphs excluding single-crossing graphs as minors. *J. Comput. Syst. Sci.*, 69(2):166–195, 2004.

[DHT02] E. Demaine, M. Hajiaghayi, and D. Thilikos. 1.5-approximation for treewidth of graphs excluding a graph with one crossing as a minor. In *APPROX*, pages 67–80, 2002.

[DHT05] E. Demaine, M. Hajiaghayi, and D. Thilikos. Exponential speedup of fixed-parameter algorithms for classes of graphs excluding single-crossing graphs as minors. *Algorithmica*, 41(4):245–267, 2005.

[Die12] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[DJ04] Víctor Dalmau and Peter Jonsson. The complexity of counting homomorphisms seen from the other side. *Theor. Comput. Sci.*, 329(1-3):315–323, 2004.

[DL92] Paul Dagum and Michael Luby. Approximating the permanent of graphs with large factors. *Theor. Comput. Sci.*, 102(2):283–305, 1992.

[Edm87] Jack Edmonds. Paths, trees, and flowers. In *Classic Papers in Combinatorics*, Modern Birkhäuser Classics, pages 361–379. Birkhäuser Boston, 1987.

[Epp00] David Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3):275–291, 2000.

[ER93] Richard Ehrenborg and Gian-Carlo Rota. Apolarity and canonical forms for homogeneous polynomials. *European Journal of Combinatorics*, 14(3):157 – 181, 1993.

[Far79] Edward J. Farrell. An introduction to matching polynomials. *Journal of Combinatorial Theory, Series B*, 27(1):75 – 86, 1979.

[FG01] Markus Frick and Martin Grohe. Deciding first-order properties of locally tree-decomposable structures. *J. ACM*, 48(6):1184–1206, 2001.

[FG04] Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. *SIAM Journal on Computing*, (4):892–922, 2004.

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[FHT95] Sophie Fischer, Lane A. Hemaspaandra, and Leen Torenvliet. Witness-isomorphic reductions and the local search problem (extended abstract). In *Mathematical Foundations of Computer Science 1995, 20th International Symposium, MFCS'95, Prague, Czech Republic, August 28 - September 1, 1995, Proceedings*, pages 277–287, 1995.

[Fri04] Markus Frick. Generalized model-checking over locally tree-decomposable classes. *Theory Comput. Syst.*, 37(1):157–191, 2004.

[GGR14] Andreas Göbel, Leslie Ann Goldberg, and David Richerby. The complexity of counting homomorphisms to cactus graphs modulo 2. *TOCT*, 6(4):17:1–17:29, 2014.

[GGR15] Andreas Göbel, Leslie Ann Goldberg, and David Richerby. Counting homomorphisms to square-free graphs, modulo 2. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 642–653, 2015.

[GHLX11] Heng Guo, Sangxia Huang, Pinyan Lu, and Mingji Xia. The complexity of weighted Boolean #CSP modulo k. In *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*, pages 249–260, 2011.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[GJ08] Leslie Ann Goldberg and Mark Jerrum. Inapproximability of the Tutte polynomial. *Information and Computation*, 206(7):908–929, 2008.

[GJ14a] Leslie Ann Goldberg and Mark Jerrum. The complexity of approximately counting tree homomorphisms. *TOCT*, 6(2):8, 2014.

[GJ14b] Leslie Ann Goldberg and Mark Jerrum. The complexity of computing the sign of the Tutte polynomial. *SIAM J. Comput.*, 43(6):1921–1952, 2014.

[GL98] Anna Galluccio and Martin Loebl. On the theory of Pfaffian orientations. I. Perfect matchings and permanents. *Electronic Journal of Combinatorics*, 6, 1998.

[GLV13] Heng Guo, Pinyan Lu, and Leslie G. Valiant. The complexity of symmetric Boolean parity Holant problems. *SIAM Journal on Computing*, 42(1):324–356, 2013.

[Gro07] Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1):1, 2007.

[GSS01] Martin Grohe, Thomas Schwentick, and Luc Segoufin. When is the evaluation of conjunctive queries tractable? In *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 657–666, New York, NY, USA, 2001. ACM Press.

[GW13] Heng Guo and Tyson Williams. The complexity of planar Boolean #CSP with complex weights. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 516–527, 2013.

[Hal76] Rudolf Halin. S-functions for graphs. *Journal of Geometry*, 8(1-2):171–186, 1976.

[Har77] Robin Hartshorne. *Algebraic Geometry*, volume 52 of *Graduate Texts in Mathematics*. Springer, 1977.

[Her90] Ulrich Hertrampf. Relations among Mod-classes. *Theor. Comput. Sci.*, 74(3):325–328, 1990.

[HO02] Lane A. Hemaspaandra and Mitsunori Ogihara. *The Complexity Theory Companion*. Springer, 2002.

[Hof10] Christian Hoffmann. Exponential time complexity of weighted counting of independent sets. In *IPEC 2010*, pages 180–191, 2010.

[Hus11] Thore Husfeldt. Invitation to algorithmic uses of inclusion-exclusion. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, pages 42–59, 2011.

[IP01] Russel Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Computer and Sys. Sci.*, 63(4):512–530, 2001.

[JM14] Mark Jerrum and Kitty Meeks. The parameterised complexity of counting even and odd induced subgraphs. *CoRR*, abs/1410.3375, 2014.

[JM15a] Mark Jerrum and Kitty Meeks. The parameterised complexity of counting connected subgraphs and graph motifs. *J. Comput. Syst. Sci.*, 81(4):702–716, 2015.

[JM15b] Mark Jerrum and Kitty Meeks. Some hard families of parameterized counting problems. *TOCT*, 7(3):11, 2015.

[JS93] Mark Jerrum and Alistair Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22(5):1087–1116, 1993.

[JSV04] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM*, 51(4):671–697, 2004.

[JVW90] François Jaeger, Dirk L. Vertigan, and Dominic J.A. Welsh. On the computational complexity of the Jones and Tutte polynomials. *Mathematical proceedings of the Cambridge Philosophical Society*, 108(1):35–53, 1990.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York.*, pages 85–103, 1972.

[Kas61] Pieter W. Kasteleyn. The statistics of dimers on a lattice: I. The number of dimer arrangements on a quadratic lattice. *Physica*, 27(12):1209 – 1225, 1961.

[Kas67] Pieter W. Kasteleyn. Graph Theory and Crystal Physics. In *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, 1967.

[KC10] Michael Kowalczyk and Jin-Yi Cai. Holant problems for regular graphs with complex edge functions. In *27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010, March 4-6, 2010, Nancy, France*, pages 525–536, 2010.

[KKR12] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce A. Reed. The disjoint paths problem in quadratic time. *J. Comb. Theory, Ser. B*, 102(2):424–435, 2012.

[KLL13] Miroslaw Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. Counting and detecting small subgraphs via equations. *SIAM J. Discrete Math.*, 27(2):892–909, 2013.

[Kur30] Kazimierz Kuratowski. Sur le Problème des Courbes Gauches en Topologie. *Fundamenta Mathematicae*, 15:271–283, 1930.

[KW11] Ken-ichi Kawarabayashi and Paul Wollan. A simpler algorithm and shorter proof for the graph minor decomposition. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 451–458, 2011.

[Lan12] Joseph M. Landsberg. *Tensors: Geometry and Applications*. American Mathematical Society, 2012.

[Lau04] Alan J. Laub. *Matrix Analysis For Scientists And Engineers*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2004.

[Lin86] Nathan Linial. Hard enumeration problems in geometry and combinatorics. *SIAM Journal on Algebraic and Discrete Methods*, 7(2):331–335, 1986.

[Lit74] Charles Little. An extension of Kasteleyn's method of enumerating the 1-factors of planar graphs. In *Combinatorial Mathematics*, LNCS, pages 63–72. 1974.

[Liv09] Noam Livne. A note on #P-completeness of NP-witnessing relations. *Inf. Process. Lett.*, 109(5):259–261, 2009.

[LMS11] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the EATCS*, 84:41–71, 2011.

[LRT79] R. Lipton, D. Rose, and R. Tarjan. Generalized nested dissection. *SIAM Journal on Numerical Analysis*, 16(2):346–358, 1979.

[LY80] John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *J. Comput. Syst. Sci.*, 20(2):219–230, 1980.

[Mak04] Johann A. Makowsky. Algorithmic uses of the Feferman-Vaught theorem. *Annals of Pure and Applied Logic*, 126(1-3):159 – 213, 2004. Provinces of logic determined. Essays in the memory of Alfred Tarski. Parts I, II and III.

[Mak06] Johann A. Makowsky. From a zoo to a zoology: Descriptive complexity for graph polynomials. In *Logical Approaches to Computational Barriers, Second Conference on Computability in Europe, CiE 2006, Swansea, UK, June 30-July 5, 2006, Proceedings*, pages 330–341, 2006.

[Mar10] Dániel Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010.

[Mar12] Dániel Marx. A tight lower bound for planar multiway cut with fixed number of terminals. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, pages 677–688, 2012.

[McC06] Catherine McCartin. Parameterized counting problems. *Ann. Pure Appl. Logic*, 138(1-3):147–182, 2006.

[Mee14] Kitty Meeks. The challenges of unbounded treewidth in parameterised subgraph counting problems. *CoRR*, abs/1402.5857, 2014.

[MP14] Dániel Marx and Michal Pilipczuk. Everything you always wanted to know about the parameterized complexity of subgraph isomorphism (but were afraid to ask). In *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, pages 542–553, 2014.

[MRAG06] Johann A. Makowsky, Udi Rotics, Ilya Averbouch, and Benny Godlin. Computing graph polynomials on graphs of bounded clique-width. In *Graph-Theoretic Concepts in Computer Science, 32nd International Workshop, WG 2006, Bergen, Norway, June 22-24, 2006, Revised Papers*, pages 191–204, 2006.

[MRST97] William McCuaig, Neil Robertson, Paul D. Seymour, and Robin Thomas. Permanents, Pfaffian orientations, and even directed circuits (extended abstract). In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 402–405, 1997.

[Nob98] S. D. Noble. Evaluating the Tutte polynomial for graphs of bounded tree-width. *Combinatorics, Probability & Computing*, 7(3):307–321, 1998.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[RL08] B. Reed and Z. Li. Optimization and recognition for K5-minor free graphs in linear time. In *LATIN 2008: Theoretical Informatics*, pages 206–215. 2008.

[RS84] Neil Robertson and Paul D. Seymour. Graph minors. III. Planar tree-width. *J. Comb. Theory, Ser. B*, 36(1):49–64, 1984.

[RS86] Neil Robertson and Paul D. Seymour. Graph minors. V. Excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1):92–114, 1986.

[RS91] Neil Robertson and Paul D. Seymour. Excluding a graph with one crossing. In *Graph Structure Theory*, pages 669–676, 1991.

[RS93] Neil Robertson and Paul D. Seymour. Excluding a graph with one crossing. In *Proceedings of the AMS-IMS-SIAM Joint Summer Research Conference on Graph Minors, Held June 22 to July 5, 1991*, pages 669–675, 1993.

[RS95] Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.

[RS03] Neil Robertson and Paul D. Seymour. Graph minors. XVI. Excluding a non-planar graph. *J. Comb. Theory, Ser. B*, 89(1):43 – 76, 2003.

[RS04] Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *J. Comb. Theory, Ser. B*, 92(2):325–357, 2004.

[Rys63] Herbert J. Ryser. Combinatorial mathematics. *Number 14 in Carus Math. Monographs. Mathematical Association of America*, 1963.

[Sok05] Alan D. Sokal. The multivariate Tutte polynomial (alias Potts model) for graphs and matroids. In *Surveys in Combinatorics, 2005 [invited lectures from the Twentieth British Combinatorial Conference, Durham, UK, July 2005]*, pages 173–226, 2005.

[SS05] Alexander D. Scott and Alan D. Sokal. The repulsive lattice gas, the independent-set polynomial, and the Lovász Local Lemma. *Journal of Statistical Physics*, 118(5-6):1151–1261, 2005.

[STW14] Simon Straub, Thomas Thierauf, and Fabian Wagner. Counting the number of perfect matchings in K_5-free graphs. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 66–77, 2014.

[Tes00] Glenn Tesler. Matchings in graphs on non-orientable surfaces. *J. Comb. Theory, Ser. B*, 78(2):198–231, 2000.

[TF61] H. N. V. Temperley and Michael E. Fisher. Dimer problem in statistical mechanics - an exact result. *Philosophical Magazine*, 6(68):1478–6435, 1961.

[Tod91] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[Vad01] Salil P. Vadhan. The complexity of counting in sparse, regular, and planar graphs. *SIAM J. Comput.*, 31(2):398–427, 2001.

[Val79a] Leslie G. Valiant. Completeness classes in algebra. In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, STOC '79, pages 249–261, New York, NY, USA, 1979. ACM.

[Val79b] Leslie G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8(2):189–201, 1979.

[Val06] Leslie G. Valiant. Accidental algorithms. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 509–517, 2006.

[Val08] Leslie G. Valiant. Holographic algorithms. *SIAM J. Comput.*, 37(5):1565–1594, 2008.

[Vaz89] Vijay V. Vazirani. NC algorithms for computing the number of perfect matchings in $K_{3,3}$-free graphs and related problems. *Inf. Comput.*, 80(2):152–164, 1989.

[VV86] Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[Wag37] Klaus Wagner. Über eine Erweiterung des Satzes von Kuratowski. *(The author of the present thesis refuses to state the name of this journal)*, 1937.

[Wil13] Tyson Williams. When does "X is NP-complete" imply "#X is #P-complete"? http://cstheory.stackexchange.com/questions/16119, Januar 2013.

[WW13] Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013.

[XZZ07] Mingji Xia, Peng Zhang, and Wenbo Zhao. Computational complexity of counting problems on 3-regular planar graphs. *Theoretical Computer Science*, 384(1):111 – 125, 2007. Theory and Applications of Models of Computation.

Appendix

A. Original hardness proof for k-matchings

In this part of the appendix, we present the first #W[1]-hardness proof for #Match, as it appeared in [BC12, Cur13]. This proof is superseded by the results of Section 5.2.

The original proof consists of two parts, both of which are based around the notions of *partial cycle covers* and *partial path-cycle covers*. In the first part, taken from [BC12], an argument seen in [FG04] is adapted to yield a reduction from #Clique to counting partial cycle covers. In the second and more involved part, taken from [Cur13], some algebraic machinery is developed in order to reduce from counting partial cycle covers to counting matchings.

Definition A.1. Let $G = (V, E)$ be a digraph and $k \in \mathbb{N}$. A *k-partial path-cycle cover* C in G is a k-set $C \subseteq E$ that consists of a vertex-disjoint union of paths and cycles.[1]

The number of cycles in C is denoted by $\sigma(C)$, that of paths by $\rho(C)$, and that of vertices in G not hit by C by $\iota(C)$.[2]

We call C a k-partial *cycle* cover if $\rho(C) = 0$, that is, if C features no paths. The set of k-partial path-cycle covers with t paths in G is denoted by $\mathcal{PC}_{k,t}[G]$, that of k-partial path-cycle covers as $\mathcal{PC}_k[G]$ and that of k-partial cycle covers by $\mathcal{C}_k[G]$.

We denote the parameterized problem of counting k-partial path-cycle covers in an input graph G by #PCC, and that of counting k-partial cycle covers by #CC. It was shown in [BC12] that #CC is #W[1]-complete, and that a weighted generalization of #PCC is as well. From this, a simple graph transformation allowed to conclude #W[1]-hardness of counting weighted k-matchings. More precisely, for edge-weighted graphs with weights $w : E(G) \to \mathbb{Z}$, it was shown that it is #W[1]-hard to compute the quantity

$$\sum_{M \in \mathcal{M}_k[G]} \prod_{e \in M} w(e).$$

We will not include the reduction to this weighted problem into this thesis as it has been superseded by [Cur13]. Instead, we only prove #W[1]-hardness of #CC and reduce this to #Match. To show hardness of #CC, we introduce a combinatorial structure that could be described as a "union of closed walks without distinguished start vertices". For notational simplicity, we call such a structure a UCW:

Definition A.2. Let $G = (V, E)$ be a digraph. Let $(v_1, \ldots, v_k) \in V^k$ such that $(v_i, v_{i+1}) \in E$ for all $i < k$ and $(v_k, v_1) \in E$. Write $[v_1, \ldots, v_k]$ for the set of all cyclic shifts of $(v_1, \ldots, v_k)$ and call $W = [v_1, \ldots, v_k]$ a *CW* of *length* $\ell(W) := k$.

A UCW is a multiset $U = \{W_1, \ldots, W_t\}$ of CWs, and its length is $\ell(U) := \sum_{i=1}^{t} \ell(W_i)$.

[1] If k is not relevant in the context, we simply call C a *partial* path-cycle cover.

[2] Note that $\iota(C) = |V(G)| - |C| - \rho(C)$.

To each UCW, we associate a particular polynomial, its *type*, which is defined analogously to the types assigned to homomorphisms in [FG04]:

Definition A.3. Let W be a CW in G and let $v \in V(G)$. We write $f_W(v)$ for the number of appearances of v in W. For $U = \{W_1, \ldots, W_t\}$ a UCW, we set $f_U(v) := \sum_{i=1}^{t} f_{W_i}(v)$. Then the *type* θ_U of U is defined as the polynomial $\theta_U \in \mathbb{Z}[x]$ with

$$\theta_U(x) := \prod_{v \in V} (x)_{f_U(v)}.$$

Let Θ_k denote the set of all types of degree k. We write $\mathcal{U}_k[G, \theta]$ for the set of UCWs of length k and type θ in G.[3]

Note that Θ_k stands in canonical bijection with the number partitions of k, that is, the ways to express k as an unordered sum of positive numbers: Factorizing type polynomials into their linear factors allows to recover such partitions. It clearly holds that $|\Theta_k| \leq k^k$.

In analogy to the problem of counting typed directed cycles, which was proven to be #W[1]-hard in [FG04], we define the problem of counting typed UCWs in digraphs:

Problem A.4 (#typUCW)**.** Given as input a digraph $G = (V, E)$, a type θ and a number $k \in \mathbb{N}$, determine the number $\#\mathcal{U}_k[G, \theta]$. The parameter is k.

In Section A.1, we prove

$$\#\mathsf{Clique} \leq_{fpt}^{T} \#\mathsf{typUCW} \leq_{fpt}^{T} \#\mathsf{CC}, \tag{A.1}$$

and in Section A.2, we then proceed to prove the more involved reduction

$$\#\mathsf{CC} \leq_{fpt}^{T} \#\mathsf{Match},$$

which finishes the #W[1]-hardness proof for #Match.

A.1. Hardness of partial cycle covers

For ease of presentation, we will present the two reductions from (A.1) in reverse order, that is, we first show how to reduce #typUCW to #CC, and then show how to reduce #Clique to #typUCW. This allows to use the graph construction presented in Definition A.5 in both reductions.

A.1.1. From typed UCWs to partial cycle covers

Let $G = (V, E)$ be a digraph and let θ be a type of an UCW of total length k. We wish to count the UCWs of type θ in G, given oracle access to #CC. Our reduction is based on a graph transformation from [FG04], which was used to reduce the problem of counting typed cyclic walks to that of counting directed cycles.

[3] Note that types already specify the lengths of UCWs; we use the (redundant) subscript k to make their length explicit.

Definition A.5. [FG04, Proof of Lemma 23] Given a digraph $G = (V, E)$ and $\ell, m \in \mathbb{N}$, we define the graph $G_{\ell,m}$ as follows:

1. Replace each $v \in V$ by the "ladder at v", which consists of vertices

$$L_v := \{(v, i, j) \mid i \in [\ell], j \in [m]\}$$

and edges

$$\{((v, i, j), (v, i + 1, j')) \mid i \in [\ell],\ j, j' \in [m]\}.$$

2. Replace each edge $e = (u, v)$ in E by the "external edges at e"

$$P_e := \{((u, l, j), (v, 1, j')) \mid j, j' \in [m]\}.$$

By construction, every cycle in $G_{\ell,m}$ must pass through $\ell - 1$ ladder edges and one external edge in an alternating way. Thus, the length of every cycle (and hence the length of every partial cycle cover) in $G_{\ell,m}$ is a multiple of ℓ.

Given $k \in \mathbb{N}$, we can partition the partial cycle covers $C \in \mathcal{C}_{k\ell}[G_{\ell,m}]$ into classes by associating with every C a particular UCW, its so-called *projection* $\pi(C)$.

Definition A.6. Let $C \in \mathcal{C}[G_{\ell,m}]$ consist of cycles $C_1, \ldots, C_{\sigma(C)}$. For each C_i with $i \in [\sigma(C)]$ we define a particular CW $\pi(C_i)$ by contracting, for each $v \in V$, the ladder at v to the single vertex v. Then we define the *projection* of C as

$$\pi(C) := \{\pi(C_1), \ldots, \pi(C_{\sigma(C)})\}$$

and observe that $\pi(C) \in \mathcal{U}_k[G]$.

We also observe that the number of partial cycle covers with a certain projection U depends only on the type of U:

Proposition A.7. *For any $\ell, m \in \mathbb{N}$, any type θ and $U \in \mathcal{U}_k[G, \theta]$, the number*

$$b(U) := \{C \in \mathcal{C}_{k\ell}[G_{\ell,m}] \mid \pi(C) = U\}$$

satisfies

$$b(U) = \theta(m)^\ell.$$

Proof. Recall Definition A.3 for the quantity $f_U(v)$ for $v \in V$. Every element $C \in b(U)$ contains, for each $v \in V$, exactly $f_U(v)$ vertex-disjoint paths passing through the ladder at v. At each ladder, these paths can be chosen in $((m)_{f_U(v)})^\ell$ ways, and paths can be chosen independently at different ladders. Once all ladder paths are fixed, the choice of external edges is also fixed. In total, this proves that

$$b(U) = \prod_{v \in V} ((m)_{f_U(v)})^\ell = \theta(m)^\ell.$$

□

This can be used to prove the wanted reduction:

Lemma A.8. *We have* $\#\mathsf{typUCW} \leq^T_{fpt} \#\mathsf{CC}$. That is, given a graph G and a type $\theta \in \Theta_k$ as input, we can count the UCWs of type θ in G using an oracle for $\#\mathsf{CC}$.

Proof. The proof follows the steps shown in [FG04, Lemma 23]. Recall that Θ_k denotes the set of types with degree k; we list the elements of Θ as $\theta_1, \ldots, \theta_t$ with $t = |\Theta|$. Since $t \leq k^k$ and every polynomial $\theta \in \Theta$ satisfies $\deg(\theta) \leq k$, there exists some number $m \leq g(k)$, where g is computable and depends only on k, such that

$$\theta(m) \neq \theta'(m) \qquad \forall \theta, \theta' \in \Theta_k \text{ with } \theta \neq \theta'. \tag{A.2}$$

We compute such an m and use oracle calls to $\#\mathsf{CC}$ to compute

$$\alpha_\ell := \#\mathcal{C}_{k\ell}[G_{\ell,m}]$$

for $1 \leq \ell \leq |\Theta|$. We write β_θ for the number $|\mathcal{U}_k[G, \theta]|$ of UCWs of type θ in G, and by Proposition A.7, we can write

$$\alpha_\ell = \sum_{\theta \in \Theta_k} \beta_\theta \cdot \theta(m)^\ell.$$

We compute $\alpha_1, \ldots, \alpha_t$ with oracle calls to $\#\mathsf{CC}$ and obtain the equation system

$$\begin{pmatrix} \theta_1(m)^1 & \ldots & \theta_t(m)^1 \\ \vdots & & \vdots \\ \theta_1(m)^t & \ldots & \theta_t(m)^t \end{pmatrix} \begin{pmatrix} \beta_{\theta_1} \\ \vdots \\ \beta_{\theta_t} \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_t \end{pmatrix}. \tag{A.3}$$

The system matrix of (A.3) is a Vandermonde matrix on the values $\theta_1(m), \ldots, \theta_t(m)$, which are pairwise distinct by (A.2). By elementary linear algebra, this matrix therefore has full rank, and its entries can be computed by simple evaluations, so (A.3) can be solved uniquely for any value of β_θ. □

A.1.2. From cliques to typed UCWs

Let $G = (V, E)$ be a graph and let $k \in \mathbb{N}$ be fixed throughout this section. We describe how to compute the number of k-cliques in G with an oracle for $\#\mathsf{typUCW}$, adapting the reduction in [FG04, Lemma 25] in large parts and reusing some of its notation where appropriate.

First, let G' be the graph obtained from G by replacing each edge by a pair of antiparallel edges, followed by adding a self-loop to each vertex. The number of k-cliques in G is equal to the number of induced subgraphs in G' which are isomorphic to the complete digraph $K = K_k := ([k], [k]^2)$.

Let $\mathcal{H} = \mathcal{H}_k := \{H_1, H_2, \ldots, K\}$ be the set of graphs on k vertices, where isomorphic graphs are identified to one single representative, and the complete digraph K is defined as above. For $H \in \mathcal{H}$, let

$$x_H = \{U \subseteq V \mid G[U] \simeq H\}.$$

Our goal is to determine x_K, the number of k-cliques in G. To this goal, let

$$\gamma_\ell := \#\mathcal{U}[G', (x)_\ell^k] \quad \text{for } \ell \in \mathbb{N}$$

denote the number of UCWs of the "special" (or rather "useful") type $(x)_\ell^k$ in G'. This value can be computed with an oracle call to #typUCW, provided that $\ell \leq f(k)$. Writing

$$\beta_{H,\ell} := \#\mathcal{U}[H, (x)_\ell^k] \quad \text{for } H \in \mathcal{H} \text{ and } \ell \in \mathbb{N},$$

we observe that there is a linear combination

$$\gamma_\ell = \sum_{H \in \mathcal{H}} x_H \cdot \beta_{H,\ell}. \tag{A.4}$$

By computing the values $\{\beta_{H,\ell} \mid H \in \mathcal{H}, \ell \in \mathbb{N}\}$ via brute-force and querying the oracle for #typUCW to obtain the values $\{\gamma_\ell \mid \ell \in \mathbb{N}\}$, we can generate the linear combination (A.4) in indeterminates $\{x_H \mid H \in \mathcal{H}\}$ for $\gamma_1, \ldots, \gamma_L$, where $L \leq f(k)$ will be determined later. This yields the system

$$\begin{pmatrix} \beta_{H_1,1} & \cdots & \beta_{K,1} \\ \vdots & \ddots & \vdots \\ \beta_{H_1,L} & \cdots & \beta_{K,L} \end{pmatrix} \begin{pmatrix} x_{H_1} \\ \vdots \\ x_K \end{pmatrix} = \begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_L \end{pmatrix}. \tag{A.5}$$

A solution to (A.5) can be found in time polynomial in $|\mathcal{H}|, L, n$. While the system (A.5) does not necessarily feature full rank, we can show as in [FG04, Lemma 25] that there exists some $L \in \mathbb{N}$ such that the last column, which corresponds to the indeterminate x_K, is not contained in the span of all other columns. This implies by elementary linear algebra that *all* solutions to (A.5) agree on their values for x_K and that the value of x_K can thus be recovered from (A.5).

To prove the existence of L, we first need a technical lemma, which asserts that, with $\ell \in \mathbb{N}$ tending to infinity, deleting even a single edge from K makes the number of UCWs of the useful type $(x)_k^\ell$ negligible in comparison to those of K. A similar statement was shown in [FG04] for cycles instead of UCWs, but using a different proof approach:

Lemma A.9. *For all graphs $H \in \mathcal{H} \setminus \{K\}$, it holds that*

$$\lim_{\ell \to \infty} \frac{\beta_{H,\ell}}{\beta_{K,\ell}} = 0. \tag{A.6}$$

Proof. Let $\ell \in \mathbb{N}$. Recall Definition A.5 and consider $G_{1,\ell}$ with $G = K$, which is isomorphic to the complete graph $K_{k\ell}$. Every $k\ell$-partial cycle cover in $G_{1,\ell}$ uses all vertices of the graph (in particular, it uses all ℓ vertices at each ladder), so its projection according to Definition A.6 is some $U \in \mathcal{U}[K, (x)_\ell^k]$. Furthermore, by Proposition A.7, every $U \in \mathcal{U}[K, (x)_\ell^k]$ is the projection of exactly $(\ell!)^k$ many $k\ell$-partial cycle covers in $K_{1,\ell}$.

Consider $H \neq K$, so there exists some edge $e \in E(K) \setminus E(H)$. Then let s denote the number of cycle covers in $G_{1,\ell}$ and let t denote the number of cycle covers in $G_{1,\ell}$ that do

not use any of the external edges at e. Since every $U \in \mathcal{U}[K, (x)_\ell^k]$ has exactly $(\ell!)^k$ cycle covers in $K_{1,\ell}$ projecting to it, we have

$$\frac{\beta_{H,\ell}}{\beta_{K,\ell}} \leq \frac{t}{s}. \tag{A.7}$$

Let O_ℓ denote the $\ell \times \ell$ all-ones matrix. Then it is easily seen that

$$t = \operatorname{perm}(B \otimes O_\ell)$$

where $\otimes$ denotes the Kronecker product and B is defined to be O_k, but with one entry, say $B_{1,1}$, set to 0. We number rows and columns of $B \otimes O_\ell$ from 1 to $k\ell$. Any permutation $\sigma \in S_{k\ell}$ that contributes to $\operatorname{perm}(B \otimes O_\ell)$ maps every element from $[\ell]$ to an element from $[k\ell] \setminus [\ell]$, which gives $((k-1)\ell)_\ell$ choices for these elements, as opposed to $(k\ell)_\ell$ choices for an arbitrary permutation. Thus,

$$\frac{t}{s} = \frac{((k-1)\ell)_\ell}{(k\ell)_\ell},$$

which can be upper-bounded by

$$\prod_{i=0}^{\ell-1} \frac{k-1-i/\ell}{k-i/\ell} = \prod_{i=0}^{\ell-1} \left(1 - \frac{1}{k-i/\ell}\right) \leq \left(1 - \frac{1}{k}\right)^\ell.$$

Since k is fixed, this value converges to 0 for $\ell \to \infty$. □

Having proved Lemma A.9 for UCWs, the following lemma is shown similarly to [FG04, Lemma 25].

Lemma A.10. *For each $k \in \mathbb{N}$, there exists a computable $L \in \mathbb{N}$ such that*

$$(\beta_{K,1}, \ldots, \beta_{K,L})^T \notin \operatorname{span}\{(\beta_{H,1}, \ldots, \beta_{H,L})^T \mid H \in \mathcal{H}_k \setminus \{K_k\}\}.$$

In conclusion, we obtain that the equation system (A.5), constructed for the value of L obtained from Lemma A.10, admits a unique solution for the indeterminate x_K.

Lemma A.11. *We have* #Clique $\leq_{fpt}^T$ #typUCW.

Finally, by combining Lemmas A.8 and A.11, we obtain hardness of counting k-partial cycle covers.

Theorem A.12. *The problem* #CC *is* #W[1]*-complete.*

This theorem will serve as the reduction source for #W[1]-completeness of #Match in the following section.

A.2. From cycle covers to matchings

In the second (and technically more involved) part of the proof, we show how to reduce from counting partial cycle covers to counting matchings. This part will be based upon techniques from commutative algebra, for which we require the following definitions.

Preliminaries from commutative algebra

It will be very convenient to introduce specific notations for polynomials and slices of polynomials determined by a subset of their indeterminates.

Definition A.13. Let $\mathbf{x} = (x_1, \ldots, x_s)$ be a tuple of indeterminates and recall from Chapter 1 that $\mathbb{N}^{\mathbf{x}}$ is the set of monomials over $\mathbf{x}$. Given a multivariate polynomial $p \in \mathbb{Z}[\mathbf{x}]$ and $\nu \in \mathbb{N}^{\mathbf{x}}$, write $c(\nu) \in \mathbb{Z}$ for the coefficient of the monomial ν in p. This induces a linear combination

$$p = \sum_{\nu} c(\nu) \cdot \nu,$$

where only finitely many $c(\nu)$ are non-zero.

Let $\mathbf{x} = \mathbf{y} \dot\cup \mathbf{z}$ be a partition of the indeterminates of p. We can then consider $p \in (\mathbb{Z}[\mathbf{z}])[\mathbf{y}]$, that is, we may consider p as a polynomial over indeterminates $\mathbf{y}$ with coefficients from $\mathbb{Z}[\mathbf{z}]$. For $\nu \in \mathbb{N}^{\mathbf{y}}$, we then define the slice $[\nu]p$ as the uniquely determined polynomial $H_\nu \in \mathbb{Z}[\mathbf{z}]$ in the expansion

$$p = \sum_{\theta \in \mathbb{N}^{\mathbf{y}}} H_\theta \cdot \theta.$$

Crucial parts of our proof rely on *algebraic independence*, a notion from commutative algebra that generalizes linear independence. A general introduction to this topic is given in [Har77].

Definition A.14. Let $P = (p_1, \ldots, p_t)$ be a tuple of polynomials in some ring and let $\mathbf{y} = (\dot p_1, \ldots, \dot p_t)$ be a tuple of indeterminates, where each indeterminate corresponds to one polynomial in P.

An *annihilator* for P is a polynomial $A \in \mathbb{Z}[\mathbf{y}]$ which *annihilates* P, i.e., which satisfies $A(p_1, \ldots, p_t) \equiv 0$. If the only annihilator for P is the zero polynomial, we call P algebraically independent.

Remark A.15. In the previous definition, we wrote $\mathbf{y} = (\dot p_1, \ldots, \dot p_t)$ to highlight the correspondence between formal indeterminates and polynomials from the set P. In this chapter, expressions of the form $\dot p$ will always denote indeterminates.

Restricting the annihilator A to linear functions without mixed-variable terms yields an alternative definition of linear independence. Algebraic independence generalizes this by allowing "polynomial" combinations instead of only linear combinations.

We require only two ingredients from the theory of algebraic independence: The classical Jacobian criterion allows us to reduce algebraic independence to linear independence, and we will use this to test algebraic independence of a certain set of polynomials. A proof of Theorem A.16 can be found in [ER93].

Theorem A.16. *Let $P = \{p_1, \ldots, p_t\} \subseteq \mathbb{Z}[\mathbf{x}]$. Then P is algebraically independent iff* $\operatorname{rank}(JP) = t$*, where JP denotes the Jacobian matrix*

$$(JP)_{i,j} = \frac{\partial p_i}{\partial x_j}.$$

Furthermore, Lemma A.17 allows us to argue about annihilators of "almost-independent" sets. In the setting of this lemma, we have a set of "untainted" polynomials Q, a set of "tainted" polynomials P and a polynomial s that can be obtained as a linear combination of P. The set $P \cup Q$ is independent, whereas $\{s\} \cup P \cup Q$ obviously is not and thus admits nontrivial annihilators A, on indeterminates $\dot{s}, \mathbf{p}, \mathbf{q}$. However, if we pick a monomial $\nu \in \mathbb{N}^{\mathbf{q}}$, consider the slice $[\nu]A$ and substitute $\dot{s}$ by the linear combination of $\mathbf{p}$ that defined s, then we obtain the zero polynomial.

Lemma A.17. *Let $P = \{p_1, \ldots, p_r\}$ and $Q = \{q_1, \ldots, q_t\}$ be sets of polynomials such that $P \cup Q$ is algebraically independent, and let $s = p_1 + \ldots + p_r$.*[4]

Define indeterminates $\dot{s}$, $\mathbf{p} = (\dot{p}_1, \ldots, \dot{p}_r)$ and $\mathbf{q} = (\dot{q}_1, \ldots, \dot{q}_t)$, and define a ring $\mathfrak{O} := \mathbb{Z}[\dot{s}, \mathbf{p}, \mathbf{q}]$. Let $A \in \mathfrak{O}$ be an arbitrary annihilator for $\{s\} \cup P \cup Q$. Let $\nu \in \mathbb{N}^{\mathbf{q}}$ be arbitrary, and consider the slice $[\nu]A$, with $[\nu]A \in \mathbb{Z}[\dot{s}, \mathbf{p}]$. Then applying the substitution

$$\dot{s} := \dot{p}_1 + \ldots + \dot{p}_r$$

to $[\nu]A$ yields a polynomial $A_\nu \in \mathbb{Z}[\dot{\mathbf{p}}]$ with $A_\nu \equiv 0$.

Proof. Since A annihilates $\{s\} \cup P \cup Q$, we have

$$A(s, p_1, \ldots, p_r, q_1, \ldots, q_t) \equiv 0.$$

Considering $A \in (\mathbb{Z}[\dot{s}, \mathbf{p}])[\mathbf{q}]$, this equation can be rewritten as

$$\sum_{\nu \in \mathbb{N}^{\mathbf{q}}} ([\nu]A)(s, p_1, \ldots, p_r) \cdot \nu(q_1, \ldots, q_t) \equiv 0. \tag{A.8}$$

Recall that A_ν denotes $[\nu]A$ after substitution of $\dot{s}$ and observe that

$$([\nu]A)(s, p_1, \ldots, p_r) = A_\nu(p_1, \ldots, p_r)$$

since $\dot{s} := \dot{p}_1 + \ldots + \dot{p}_r$ and $s = p_1 + \ldots + p_r$. If $A_\nu \not\equiv 0$ for some ν, then (A.8) displays a nontrivial annihilator for $P \cup Q$ after substitution of $\dot{s}$, contradicting the algebraic independence of $P \cup Q$. □

Outline of the reduction

We prove #W[1]-hardness of #Match by a reduction from #CC. Given a partial path-cycle cover C, recall that $\rho(C)$ denotes the number of paths in C and recall that C is a t-partial

[4] In the definition of s, an arbitrary linear combination could be chosen instead of a mere sum. For ease of presentation, we chose to describe only the case of a sum.

cycle cover if $\rho(C) = 0$. Recall that the set of t-partial path-cycle covers in G with ρ paths is denoted by $\mathcal{PC}_{t,\rho}[G]$, whereas $\mathcal{PC}_t[G] := \bigcup_\rho \mathcal{PC}_{t,\rho}[G]$ and $\mathcal{PC}[G] := \bigcup_t \mathcal{PC}_t[G]$. For $t, \rho \in \mathbb{N}$, we write $m_{t,\rho} := |\mathcal{PC}_{t,\rho}[G]|$, provided that G is clear from the context.

The reduction proceeds as follows: We are given as input a directed graph G and $k \in \mathbb{N}$, and we wish to count the number of k-partial cycle covers in G. We are also given an oracle for #Match that can be queried about the numbers of K-matchings in arbitrary graphs, provided that $K \leq g(k)$, where g is computable. In our case, the issued queries will even satisfy $K \leq 3k$, and in fact, it will even turn out that our reduction can be carried out in polynomial time.

The proof begins in Section A.2.1 by introducing a particular graph transformation: First, we construct an undirected graph G' from G, together with a bijection $S : \mathcal{PC}_k[G] \to \mathcal{M}_k[G']$. Next, we apply gadgets to G' so as to obtain a graph H for which we can show that the quantity $|\mathcal{M}_K[H]|$ with $K = 3k$ can be expressed as a particular weighted sum over the matchings $M \in \mathcal{M}_k[G']$. The weight of M in this sum depends on the number of paths in its associated path-cycle cover $S^{-1}(M)$.

We proceed to show in Sections A.2.1 and A.2.2 that the weights in this sum allow to distinguish matchings $M \in \mathcal{M}_k[G']$ according to the number of paths in $S^{-1}(M)$ and finally use this in Section A.2.2.2 to recover the number of k-partial path-cycle covers with *zero* paths in G by oracle calls to #Match on H.

A.2.1. The graph construction

We will first present the "global" construction of the graph, which will feature a certain gadget (in fact, a matchgate) whose precise manifestation is not relevant at first. In the second part of this subsection, we will then proceed to describe the specific gadget used.

A.2.1.1. Global construction

We want to count k-partial cycle covers in a directed graph G with an oracle for #Match. Let $n = |V(G)|$. First, we define a graph $S(G)$ as in [BC11]:

Definition A.18. Given a directed graph $G = (V, E)$, replace each vertex $w \in V$ by vertices w^{in} and w^{out}, and replace each $(u, v) \in E$ by the undirected edge $\{u^{out}, v^{in}\}$. We call the resulting graph the bipartite *split graph* $S(G)$.

Let $G' = S(G)$. The graph G' is indeed bipartite, with bipartition into in- and out-vertices. Considering S as a function mapping from $E(G)$ to $E(G')$, it induces a bijection between $\mathcal{PC}_t[G]$ and $\mathcal{M}_t[G']$ for all $t \in \mathbb{N}$, as can be easily seen and is shown in [BC11]. Consider the left and middle part of Fig. A.1 on the next page for an example. We also observe the following:

Remark A.19. Let $C \in \mathcal{PC}_{t,\rho}[G]$. Since C has ρ paths, there are ρ vertices incident with only an incoming edge in C, another ρ vertices incident with only an outgoing edge, and $t - \rho$ vertices incident with both an incoming and an outgoing edge. The remaining $\iota(C) = n - t - \rho$ vertices are not incident with any edge in C.

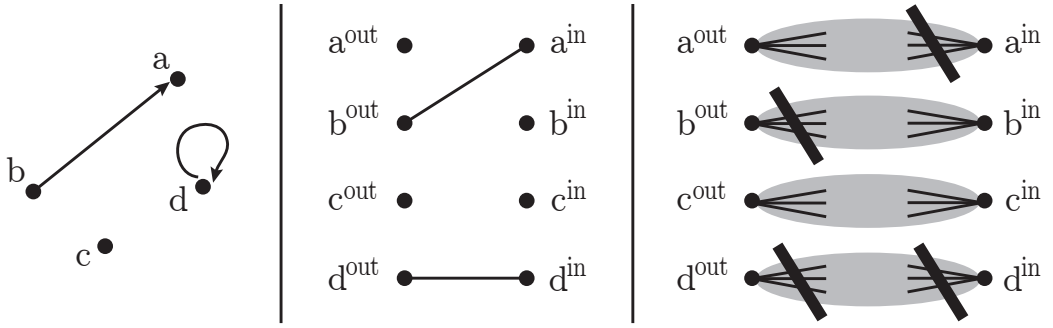


Figure A.1.: **(left)** A partial path-cycle cover C. **(middle)** The matching $M' = S(C)$. **(right)** For $w \in V$, the gray area between $\{w^{out}, w^{in}\}$ symbolizes $\mathcal{V}_w$. Cancelled edges indicate edges in $\mathcal{V}_w$ that cannot be included into M' since $\{w^{out}, w^{in}\}$ is in-blocked, out-blocked or blocked, as seen in the first, second and fourth pair, respectively.

This translates to $M = S(C)$ as follows: Consider pairs $\{w^{out}, w^{in}\} \subseteq V(G')$, for $w \in V(G)$. There are ρ such pairs with $w^{in} \notin \mathrm{usat}(M)$ and $w^{out} \in \mathrm{usat}(M)$, which we call *in-blocked*. There are another ρ pairs with $w^{out} \notin \mathrm{usat}(M)$ and $w^{in} \in \mathrm{usat}(M)$, which we call *out-blocked*. There are $t - \rho$ pairs with both $w^{out}, w^{in} \notin \mathrm{usat}(M)$, which we call *blocked*. The remaining $n - t - \rho$ pairs have $w^{out}, w^{in} \in \mathrm{usat}(M)$, and we call these pairs *free*. □

Roughly speaking, this implies the following: If we can distinguish matchings in G' according to how many pairs $\{w^{out}, w^{in}\}$ occur in the above states, then we can hope to distinguish path-cycle covers C, which correspond bijectively to M, by their numbers of paths.

In the remaining section, we present a particular construction that achieves exactly this, as will be proven in Section A.2.2. This construction uses a gadget, i.e., an undirected graph $\mathcal{V}$ with two special vertices u^{out} and u^{in} that can be inserted locally into G' to yield a graph $H = H(G)$.[5]

Definition A.20. Given a graph G, define a graph $H = H(G)$ as follows: First, let $G' = S(G)$. Then, for each $w \in V(G)$, add a fresh copy $\mathcal{V}_w$ of $\mathcal{V}$ to G', identify the vertex $w^{out} \in V(G')$ with $u^{out} \in V(\mathcal{V}_w)$ and identify $w^{in} \in V(G')$ with $u^{in} \in V(\mathcal{V}_w)$. Note that, by construction, G' appears as a subgraph in H.

Let $s \in V(G')$ with $s \in \{w^{out}, w^{in}\}$ for $w \in V(G)$. If $M \in \mathcal{M}[H]$ and s is matched in M, then $s \in e$ for some $e \in M$. Then either $e \in E(\mathcal{V}_w)$, in which case we call s *internally* matched, or $e \in E(G')$ and we call s *externally* matched. If s is externally matched, then all edges in M that stem from $E(\mathcal{V}_w)$ must be contained in $E(\mathcal{V}_w - s)$. Thus, when extending matchings $N \in \mathcal{M}[G']$ to $M \in \mathcal{M}[H]$ by including edges from $\mathcal{V}_w$, we have to distinguish the state of the pair $\{w^{out}, w^{in}\}$ in N.[6] This is illustrated in the right part of Figure A.1.

We account for this by associating four matching polynomials with the gadget $\mathcal{V}$, one for each of its states:

[5] After suitable extensions to the theory of matchgates, we could also view this gadget $\mathcal{V}$ as a matchgate.
[6] In other words, we are considering a *matchgate* at this point.

Definition A.21. Let $\mathcal{V}$ be a fixed graph containing vertices u^{out}, u^{in}. Then we define

$$\begin{aligned} F &:= M(\mathcal{V}), \\ V &:= M(\mathcal{V} - \{u^{out}\}), \\ U &:= M(\mathcal{V} - \{u^{in}\}), \\ B &:= M(\mathcal{V} - \{u^{out}, u^{in}\}). \end{aligned}$$

For $t, \rho \in \mathbb{N}$ with $\rho \leq t$ and $n - t - \rho \geq 0$, we define a polynomial $\mathsf{Mix}_{t,\rho} \in \mathbb{Z}[X]$ by

$$\mathsf{Mix}_{t,\rho} := B^{t-\rho} \cdot U^{\rho} \cdot V^{\rho} \cdot F^{n-t-\rho}.$$

The polynomials $\mathsf{Mix}_{t,\rho}$ are crucial in Section A.2.2 because the matching polynomial $M(H)$ can be written as a weighted sum over $C \in \mathcal{PC}[G]$ such that each C on t edges and ρ paths is weighted by $X^t \cdot \mathsf{Mix}_{t,\rho}$. This is stated in the following lemma, which can be proven with standard arguments. Recall that we use the notation $m_{t,\rho}(G) = |\mathcal{PC}_{t,\rho}[G]|$.

Lemma A.22. *Let G be a graph and let $H = H(G)$ as in Definition A.20. Then*

$$M(H) = \sum_{0 \leq \rho \leq t \leq n} m_{t,\rho}(G) \cdot X^t \cdot \mathsf{Mix}_{t,\rho}.$$

We close this subsection with a remark about the coefficients of B, U, V, F:

Remark A.23. Since there is exactly one empty matching, we have $[X^0]\mathsf{D} = 1$ for all $\mathsf{D} \in \{B, U, V, F\}$. Furthermore, it can be verified that

$$[X^1]F = [X^1](U + V - B)$$

provided $\{u, v\} \notin E(\mathcal{V})$. This will be the case for the gadget introduced in the following. □

A.2.1.2. The Venn gadget

We are ready to provide an explicit construction for the gadget $\mathcal{V}$: The *Venn gadget* $\mathcal{V}(\mathbf{x})$ is an undirected graph with special vertices u^{out} and u^{in}, as shown in Fig. A.2. Its precise manifestation depends upon a tuple of 11 parameters

$$\mathbf{x} = (a_\emptyset, a_u, a_v, a_{uv},\ b_\emptyset, b_u, b_v, b_{uv},\ c_u, c_v, c_{uv}) \in \mathbb{N}^{11}. \tag{A.9}$$

These parameters, which will be considered as indeterminates later, are named so as to reflect a particular set system that is represented by the gadget.

Definition A.24. Given a tuple $\mathbf{w} \in \mathbb{N}^{11}$, as specified in (A.9), the *Venn gadget* $\mathcal{V}(\mathbf{w})$ is constructed as follows from the empty graph:

1. Create $\sum \mathbf{w}$ fresh and unnamed vertices. Abusing notation, group these vertices into sets $a_\emptyset, \ldots, c_{uv}$ in the obvious way.
2. Create vertex u^{out}, adjacent to all of $(a_u \cup a_{uv}) \cup (b_u \cup b_{uv}) \cup (c_u \cup c_{uv})$.

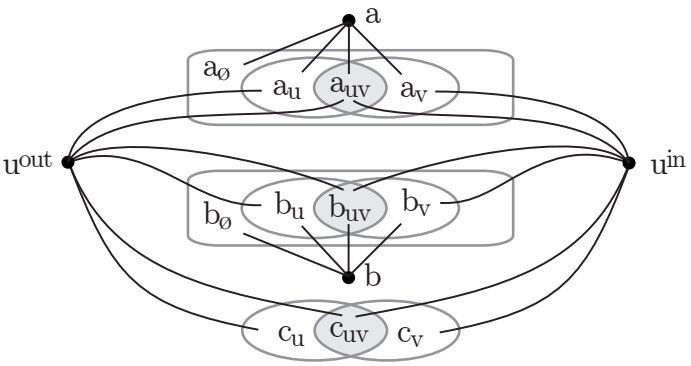


Figure A.2.: The Venn gadget $\mathcal{V}$ features named vertices u^{out}, u^{in}, a and b. The other vertices are partitioned into the disjoint sets $a_\emptyset, \ldots, c_{uv}$. In this figure, an edge leading from a special vertex w into a set S symbolizes that w is adjacent to all vertices in S.

3. Create vertex u^{in}, adjacent to all of $(a_v \cup a_{uv}) \cup (b_v \cup b_{uv}) \cup (c_v \cup c_{uv})$.

4. Create vertex a, adjacent to all of $a_\emptyset \cup a_u \cup a_v \cup a_{uv}$.

5. Create vertex b, adjacent to all of $b_\emptyset \cup b_u \cup b_v \cup b_{uv}$.

Remark A.25. The construction of $\mathcal{V}(\mathbf{w})$ for different $\mathbf{w} \in \mathbb{N}^{11}$ yields different graphs. Thus, using the gadget $\mathcal{V}(\mathbf{w})$ to construct the graph H in Definition A.20 in fact yields a graph $H = H(\mathbf{w})$.

When considering $\mathbf{w}$ as indeterminates, the matching polynomials B, U, V, F associated with $\mathcal{V}$, which were introduced in Definition A.21, are easily seen to be elements in $\mathbb{Z}[X, \mathbf{w}]$. Equivalently, we can define $\mathfrak{I} := \mathbb{Z}[\mathbf{w}]$ and say that $B, U, V, F \in \mathfrak{I}[X]$, where X denotes a formal generating variable. □

We now consider the coefficients of the polynomials $B, U, V, F \in \mathfrak{I}[X]$ from Remark A.25. Note that these coefficients are elements of $\mathfrak{I} = \mathbb{Z}[\mathbf{x}]$, and thus in turn polynomials. We show that the set of coefficients is "almost" algebraically independent, in the sense that it allows to invoke Lemma A.17.

First observe that $\deg(B) = 2$, that $\deg(U) = \deg(V) = 3$ and that $\deg(F) = 4$, as these are the maximum cardinalities of matchings counted by B, U, V, F, respectively. For $\mathsf{D} \in \{B, U, V, F\}$, abbreviate the i-th coefficient of D by D_i and note that $B_0 = V_0 = U_0 = F_0 = 1$ by Remark A.23. We will ignore these four coefficients from now on, for reasons that will become clear in Section A.2.2. Let $\mathcal{Y}$ be the set of all *other* coefficients of B, U, V, F. For convenience, we list these 12 coefficients as

$$\mathcal{Y} := \{B_1, B_2,\ U_1, U_2, U_3,\ V_1, V_2, V_3,\ F_1, F_2, F_3, F_4\}.$$

Additionally, let

$$\mathcal{B} := \{B_1, U_1, V_1, F_1\}$$

and note that $F_1 = U_1 + V_1 - B_1$ by Remark A.23, so $\mathcal{B}$ is linearly dependent. We will consider F_1 as a linear combination of the "tainted" set $P := \mathcal{B} \setminus \{F_1\}$, and we will consider the set $Q := \mathcal{Y} \setminus \mathcal{B}$ as "untainted". After computing the elements in $P \cup Q$ explicitly, we verify their algebraic independence in Section A.2.3 and obtain the following lemma:

Lemma A.26. *The set $P \cup Q$ is algebraically independent.*

We can now apply Lemma A.17 to obtain the following corollary. It states restrictions on certain slices that every annihilator for $\mathcal{Y}$ must satisfy, and which will be used in Section A.2.2.1.

Corollary A.27 (of Lemma A.17)**.** *Let P, Q be defined as above, and recall that $P \cup Q$ is algebraically independent and that $F_1 = U_1 + V_1 - B_1$.*

Define indeterminates $\dot{F}_1$, $\mathbf{p} = (\dot{B}_1, \dot{U}_1, \dot{V}_1)$ and $\mathbf{q}$, where $\mathbf{q}$ represents Q, and let $\mathbf{y} = (\dot{F}_1, \mathbf{p}, \mathbf{q})$. Let $\mathfrak{O} = \mathbb{Z}[\mathbf{y}]$ and let $A \in \mathfrak{O}$ annihilate $\mathcal{Y} = \{F_1\} \cup P \cup Q$. For any $b > 0$, let $\theta^* = (\dot{B}_2)^b$ and consider $[\theta^*]A \in \mathbb{Z}[\dot{F}_1, \mathbf{p}]$. Then applying the substitution

$$\dot{F}_1 := \dot{U}_1 + \dot{V}_1 - \dot{B}_1$$

to $[\theta^*]A$ yields a polynomial $A_{\theta^*} \in \mathbb{Z}[\mathbf{p}]$ with $A_{\theta^*} \equiv 0$.

A.2.2. Analysis of the graph construction

Recall that we wish to determine $m_{k,0}$, where $m_{t,\rho}$ denotes the number of t-partial path-cycle covers with ρ paths in G. We fix k and $K := 3k$. We also fix the indeterminates $\mathbf{y}$ and the "outer" ring $\mathfrak{O} = \mathbb{Z}[\mathbf{y}]$ as in Corollary A.27, as well as the twelve indeterminates $\mathbf{w}$ and the "inner" ring $\mathfrak{I} = \mathbb{Z}[\mathbf{w}]$ as in Remark A.25.

The indeterminates in $\mathbf{y}$ correspond to the coefficients $\mathcal{Y} \subseteq \mathfrak{I}[X]$ from Section A.2.1.2. We extend this view by considering the polynomials B, U, V, F and $\mathsf{Mix}^{(t,\rho)} \in \mathbb{Z}[X]$ from Definition A.21 formally as elements from $\mathfrak{O}[X]$ and write $\mathsf{Mix}^{\mathfrak{O}}_{t,\rho}$ to make this explicit:

Definition A.28. For $\mathsf{D} \in \{B, U, V, F\}$, let

$$\mathsf{D}^{\mathfrak{O}} = \sum_{i=1}^{\deg(\mathsf{D})} \dot{\mathsf{D}}_i X^i \in \mathfrak{O}[X].$$

Define $\mathsf{Mix}^{\mathfrak{O}}_{t,\rho} \in \mathfrak{O}[X]$ exactly as $\mathsf{Mix}^{\mathfrak{O}}_{t,\rho}$ in Definition A.21, but replace any D by $\mathsf{D}^{\mathfrak{O}}$.

Let $\mathsf{Mix}^{\mathfrak{O}} \in \mathfrak{O}^{(K+1)\times(K+1)}$ be the matrix whose entry at (t, ρ) is equal to $[X^{K-t}]\mathsf{Mix}^{\mathfrak{O}}_{t,\rho}$ for $0 \leq \rho \leq t \leq K$, and 0 else. Slightly abusing notation, we also write $\mathsf{Mix}^{\mathfrak{O}}$ for the set of entries appearing in the matrix $\mathsf{Mix}^{\mathfrak{O}}$.

We similarly define a matching polynomial $M^{\mathfrak{O}}(H) \in \mathfrak{O}[X]$ by formally replacing coefficients of Venn gadgets with indeterminates from $\mathbf{y}$. Extending Lemma A.22, we obtain:

Lemma A.29. *Let $H = H(G)$ according to Definition A.20. For matrices A, B of the same dimensions, let*

$$A \odot B := \sum_{ij} A_{ij} B_{ij}.$$

Then it holds that

$$[X^K]M^{\mathfrak{O}}(H) = \underbrace{\begin{pmatrix} [X^K]\mathsf{Mix}^{\mathfrak{O}}_{0,0} & \dots & [X^0]\mathsf{Mix}^{\mathfrak{O}}_{K,0} \\ & \ddots & \vdots \\ & & [X^0]\mathsf{Mix}^{\mathfrak{O}}_{K,K} \end{pmatrix}}_{=\mathsf{Mix}^{\mathfrak{O}}} \odot \begin{pmatrix} m_{0,0} & \dots & m_{K,0} \\ & \ddots & \vdots \\ & & m_{K,K} \end{pmatrix}.$$

This yields a formal "linear combination" of the quantities $m_{t,\rho}$ with coefficients from $\mathfrak{O}$. For $t = k$ and $0 \leq \rho \leq k$, the interesting quantities $m_{k,\rho}$ appear in it as

$$[X^K]M^{\mathfrak{O}}(H) = \ldots + m_{k,0}[X^{2k}]\mathsf{Mix}^{\mathfrak{O}}_{k,0} + \ldots + m_{k,k}[X^{2k}]\mathsf{Mix}^{\mathfrak{O}}_{k,k} + \ldots \tag{A.10}$$

In Section A.2.2.1, we substitute the polynomials $\mathcal{Y} \subseteq \mathfrak{I}$ from Section A.2.1.2 into the indeterminates $\mathbf{y}$, yielding a matrix $\mathsf{Mix}^{\mathfrak{I}} \in \mathfrak{I}^{(K+1)\times(K+1)}$. We show that, after this substitution, the polynomial

$$p^* := [X^{2k}]\mathsf{Mix}^{\mathfrak{I}}_{k,0}$$

associated with $m_{k,0}$ in (A.10) is *special*, in the sense that it cannot be written as a linear combination (with rational coefficients) of the other polynomials in $\mathsf{Mix}^{\mathfrak{I}}$.

In Section A.2.2.2, we show that a linear system of equations in the unknowns $m_{t,\rho}$ can be set up from (A.10) by evaluating the entries of $\mathsf{Mix}^{\mathfrak{I}}$ on distinct points $\xi \in \mathbb{N}^{11}$. These evaluations are performed using oracle calls on the graphs $H(\xi)$ obtained by the gadgets from Section A.2.1. The resulting system will feature $O(k^{11})$ linear equations, whose integer coefficients can be computed in time $n^{O(1)}$. Furthermore, the fact that p^* is special will imply that the system can be solved unambiguously for $m_{k,0}$.

A.2.2.1. The polynomial p^* is special

We consider expansions of the polynomials $p \in \mathsf{Mix}^{\mathfrak{O}}$ into monomials over $\mathbf{y}$ and use this to show that, after substitution of the coefficients $\mathcal{Y}$ from Section A.2.1.2 into the formal indeterminates $\mathbf{y}$, the polynomial $p^* = [X^{2k}]\mathsf{Mix}^{\mathfrak{I}}_{k,0}$ associated with $m_{k,0}$ satisfies the following:

Theorem A.30. *Let $\mathsf{Mix}^{\mathfrak{I}}$ denote the matrix obtained from $\mathsf{Mix}^{\mathfrak{O}}$ by substituting $\mathcal{Y}$ into $\mathbf{y}$ in every entry. Then the polynomial $p^* = [X^{2k}]\mathsf{Mix}^{\mathfrak{I}}_{k,0}$ is not contained in the span of the other entries in $\mathsf{Mix}^{\mathfrak{I}}$. Formally, if*

$$\sum_{0 \leq \rho \leq t \leq K} \alpha_{t,\rho} \cdot [X^{K-t}]\mathsf{Mix}^{\mathfrak{I}}_{t,\rho} \equiv 0, \tag{A.11}$$

with $\alpha_{t,\rho} \in \mathbb{Q}$ for all $0 \leq \rho \leq t \leq K$, then $\alpha_{k,0} = 0$.

This theorem will be proven at the end of this subsection. We first consider the polynomials in $\mathsf{Mix}^{\mathfrak{O}}$ and require some notation for the set of monomials (from $\mathbb{N}^{\mathbf{y}}$) appearing in such polynomials. Recall that $\mathfrak{O} = \mathbb{Z}[\mathbf{y}]$ and note that $[\theta]p \in \mathbb{Z}$ for $p \in \mathfrak{O}$ and $\theta \in \mathbb{N}^{\mathbf{y}}$.

Definition A.31. For $p \in \mathfrak{O}$, we denote the set of monomials with non-zero coefficients in p as

$$\mathfrak{M}[p] = \{\theta \in \mathbb{N}^{\mathbf{y}} \mid [\theta]p \neq 0\}.$$

For a set of polynomials $P \subseteq \mathfrak{O}$, we extend this to $\mathfrak{M}[P] = \bigcup_{p \in P} \mathfrak{M}[p]$. If $\theta \in \mathfrak{M}[P]$, we say that θ *appears* in P.

Our proof of Theorem A.30 proceeds as follows: We first identify a special monomial $\theta^* \in \mathbb{N}^{\mathbf{y}}$ and show that, among all entries of $\mathsf{Mix}^{\mathfrak{O}}$, the monomial θ^* appears only in the special polynomial p^*. We use this to show that, if p^* were contained in the span of the other polynomials in $\mathsf{Mix}^{\mathfrak{I}}$, then this would imply an annihilator for $\mathcal{Y}$ which violates a condition imposed by Corollary A.27.

To begin with, we define several quantities associated with monomials in $\mathbb{N}^{\mathbf{y}}$. Some of the upcoming definitions may seem awkwardly specific. However, we think that phrasing all definitions in their full generality is detrimental to the legibility of this part.

Definition A.32. Let $\theta \in \mathbb{N}^{\mathbf{y}}$, and for "convenience", observe that θ is of the form

$$\theta = (\dot{B}_1^{b_1}\dot{B}_2^{b_2})(\dot{U}_1^{u_1}\dot{U}_2^{u_2}\dot{U}_3^{u_3})(\dot{V}_1^{v_1}\dot{V}_2^{v_2}\dot{V}_3^{v_3})(\dot{F}_1^{f_1}\dot{F}_2^{f_2}\dot{F}_3^{f_3}\dot{F}_4^{f_4}).$$

We define a total degree

$$\mathrm{td}(\theta) := \sum_{i=1}^{4} i(b_i + u_i + v_i + f_i).$$

Let $\Theta := \mathfrak{M}[\mathsf{Mix}_{\mathfrak{O}}]$. For $\ell \in \mathbb{N}$, collect the monomials of total degree ℓ in Θ as

$$\Theta_\ell := \Theta \cap \{\theta \mid \mathrm{td}(\theta) = \ell\}.$$

For $\theta \in \mathbb{N}^{\mathbf{y}}$, define a tuple

$$\mathrm{occ}(\theta) := (\sum_i b_i, \sum_i u_i, \sum_i v_i, \sum_i f_i)$$

and write $\mathrm{occ}_B(\theta)$ for its first entry.

Example A.33. Let

$$\theta = \dot{B}_1^1\dot{B}_2^2\dot{U}_2^4\dot{V}_2^5\dot{F}_1^6.$$

Then

$$\mathrm{td}(\theta) = 1 \cdot (1+6) + 2 \cdot (2+4+5) = 29$$

and $\mathrm{occ}(\theta) = (3,4,5,6)$. Furthermore, we have $\mathrm{occ}_B(\theta) = 3$.

This notation is used for the statement of the following lemma, which follows from a straightforward application of the multinomial theorem.

Lemma A.34. *Let $0 \leq t \leq K$. For $a, b_1, \ldots, b_\ell \in \mathbb{N}$ with $s := \sum_i b_i \leq a$, let*

$$\binom{a}{b_1, \ldots, b_\ell} = \frac{a!}{b_1! \ldots b_\ell! (a-s)!}.$$

With $\theta \in \mathbb{N}^{\mathcal{Y}}$ written as a product as in Definition A.32, we have

$$[X^{K-t}]\mathsf{Mix}^{\mathfrak{O}}_{t,\rho} = \sum_{\theta \in \Theta_{K-t}} \underbrace{\binom{t-\rho}{b_1, b_2}\binom{\rho}{u_1, u_2, u_3}\binom{\rho}{v_1, v_2, v_3}\binom{n-t-\rho}{f_1, f_2, f_3, f_4}}_{=:\lambda_{t,\rho}(\theta)} \theta.$$

Corollary A.35. A monomial $\theta \in \Theta$ appears in $[X^{K-t}]\mathsf{Mix}^{\mathfrak{O}}_{t,\rho}$ iff its total degree satisfies $\mathrm{td}(\theta) = K - t$ and additionally $\lambda_{t,\rho}(\theta) \neq 0$, with $\lambda_{t,\rho}$ as defined in Lemma *A.34*. The second condition is true iff

$$\mathrm{occ}(\theta) \leq (t-\rho, \rho, \rho, n-t-\rho),$$

where $\leq$ is considered component-wise.

Our "special" monomial will be defined as $\theta^* := \dot{B}_2^k$ and we will show in the following lemma that it appears only in the previously defined special polynomial $[X^{2k}]\mathsf{Mix}^{\mathfrak{O}}_{k,0}$.

Lemma A.36. *If $\theta \in \Theta$ contains $\theta^* = \dot{B}_2^k$ as a factor, then $\theta = \theta^*$. Furthermore, if θ^* appears in $p \in \mathsf{Mix}^{\mathfrak{O}}$, then $p = p^*$. In fact, we have $[\theta^*]p^* = 1$.*

Proof. If $\theta \in \Theta$ contains $\dot{B}_2^k$, then $\mathrm{td}(\theta) \geq 2k$ by definition of the total degree td. Since $\theta \in \Theta$, it must appear in $[X^{K-t}]\mathsf{Mix}^{\mathfrak{O}}_{t,\rho}$ for some $0 \leq \rho \leq t \leq K$. Then $K - t \geq \mathrm{td}(\theta)$ by Corollary A.35. Recall that we fixed $K = 3k$, which implies that $t \leq k$. Since θ contains $\dot{B}_2^k$, we have $\mathrm{occ}_B(\theta) \geq k$. But by Corollary A.35, we also have $\mathrm{occ}_B(\theta) \leq t - \rho$.

The last two inequalities and $t \leq k$ imply $\rho = 0$ and $\mathrm{occ}_B(\theta) = k$. Thus θ appears only in p^*. But then $\mathrm{td}(\theta) = 2k$, and thus $\theta = \theta^*$.

Finally, we can use Lemma A.34 to obtain that

$$[\theta^*]p^* = \lambda_{k,0}(\theta^*) = 1,$$

proving all claims made. □

This allows us to finish the subsection with the promised proof of Theorem A.30.

Proof of Theorem A.30. Assume there were coefficients $\alpha_{t,\rho}$ satisfying (A.11) with $\alpha_{k,0} \neq 0$. With $\lambda_{t,\rho}(\theta)$ from Lemma A.34, write

$$[X^{K-t}]\mathsf{Mix}^{\mathfrak{O}}_{t,\rho} = \sum_{\theta \in \Theta} \lambda_{t,\rho}(\theta) \cdot \theta$$

and rearrange (A.11) to obtain an annihilator $A \in \mathfrak{O}$ for $\mathcal{Y}$ with

$$A := \left(\sum_{\theta \in \Theta} \theta \cdot \alpha_{k,0} \cdot \lambda_{k,0}(\theta) \right) + \sum_{\theta \in \Theta} \theta \cdot \sum_{\substack{0 \le \rho \le t \le K \\ (t,\rho) \neq (k,0)}} \alpha_{t,\rho} \cdot \lambda_{t,\rho}(\theta), \tag{A.12}$$

that is, $A(B_1, \dots, F_4) \equiv 0$. By Lemma A.36, the monomial $\theta^* = \dot{B}_2^k$ appears only within the parentheses, and with $\lambda_{k,0}(\theta^*) = 1$. Regrouping (A.12) yields

$$A = \alpha_{k,0} \cdot \theta^* + \sum_{\theta \neq \theta^*} \mu(\theta) \cdot \theta,$$

for new coefficients $\mu : \Theta \to \mathbb{Q}$. Also by Lemma A.36, the only monomial in A that contains θ^* is θ^* itself. Therefore, A is a nontrivial annihilator for the set $\mathcal{Y}$ from Section A.2.1.2, with the property that $[\theta^*]A = \alpha_{k,0}$ is non-zero. Corollary A.27 then leads to a contradiction: Since $[\theta^*]A \neq 0$ is constant, it is unaffected by the substitution $\dot{F}_1 := \dot{U}_1 + \dot{V}_1 - \dot{B}_1$, thus contradicting $A_{\theta^*} \equiv 0$ from Corollary A.27. □

A.2.2.2. Deriving linear equations

In this subsection, we complete the reduction. For this, we substitute the coefficients $\mathcal{Y}$ from Section A.2.1.2 into $\mathsf{Mix}^{\mathfrak{O}}$. By constructing the gadget $\mathcal{V}(\mathbf{w})$ for different values $\mathbf{w}$, we can evaluate the resulting polynomials $\mathsf{Mix}^{\mathfrak{I}}_{t,\rho}$ to yield integer values.

Definition A.37. For $\xi \in \mathbb{N}^{11}$, let $\mathsf{Mix}(\xi) \in \mathbb{Z}^{(K+1)\times(K+1)}$ be the matrix obtained from $\mathsf{Mix}^{\mathfrak{I}}$ by evaluating each of its entries at ξ. Recall that the elements of $\mathsf{Mix}^{\mathfrak{I}}$ are polynomials in $\mathfrak{I} = \mathbb{Z}[\mathbf{w}]$, so they can indeed be evaluated at ξ.

For a list of D vectors $\Xi = (\xi_1, \dots, \xi_D)$ with $\xi_i \in \mathbb{N}^{11}$ for $i \in [D]$, let $\mathsf{Mix}(\Xi) \in \mathbb{Z}^{D\times(K+1)^2}$ be the matrix whose i-th row contains the entries of $\mathsf{Mix}(\xi_i)$ as a row vector. We consider the columns of $\mathsf{Mix}(\Xi)$ to be indexed by pairs (t, ρ).

Remark A.38. If $|\Xi| \le n^{O(1)}$ and all entries of Ξ have bit-length $n^{O(1)}$, then $\mathsf{Mix}(\Xi)$ can be computed in time $n^{O(1)}$: By Definition A.21, each element in the matrix $\mathsf{Mix}^{\mathfrak{O}}_{t,\rho}$ is the product of n polynomials, each of degree ≤ 4. Any such element can therefore be computed in polynomial time, the coefficients $\mathcal{Y}$ can be substituted into it, and the resulting polynomials can be evaluated at any ξ_i, all in time $n^{O(1)}$.

In the following, we fix $\Xi = (\xi_1, \dots, \xi_D)$ with $D = (K+1)^{11}$ to the lexicographic enumeration of the grid $\{0, \dots, K\}^{11}$. Given a matrix $B \in \mathbb{Z}^{\ell \times b^2}$ whose columns are indexed by pairs (i, j), and another matrix $C \in \mathbb{Z}^{b\times b}$, we define the vector $B \odot C$, an element of $\mathbb{Z}^{\ell}$, via

$$(B \odot C)_t = \sum_{ij} B_{t,(i,j)} C_{ij}, \qquad \forall t \in [\ell].$$

In other words, the vector $B \odot C$ can be considered as the matrix-vector product of B and C', where C' is the result obtained from flattening C to a vector of length b^2.

With this notation, and using Lemma A.29, it can be checked that

$$\mathsf{Mix}(\Xi) \odot \begin{pmatrix} m_{0,0} & \dots & m_{K,0} \\ & \ddots & \vdots \\ & & m_{K,K} \end{pmatrix} = \begin{pmatrix} [X^K]M(H(\xi_1)) \\ \vdots \\ [X^K]M(H(\xi_D)) \end{pmatrix}, \tag{A.13}$$

with $H(\xi)$ for $\xi \in \mathbb{N}^{11}$ as defined in Remark A.25. Recall that the right-hand side of (A.13) counts the K-matchings in $H(\xi)$; since $K = 3k$, it can thus be evaluated with D oracle queries to #Match.

We consider (A.13) as a linear system of equations in the unknowns $m_{t,\rho}$, which we wish to solve for $m_{k,0}$. By Remark A.38, its system matrix $\mathsf{Mix}(\Xi)$ can be evaluated in polynomial time, which implies that a solution to (A.13) can also be found in time $n^{O(1)}$. The final and crucial step now consists of showing that *all* solutions to (A.13) agree on their values for $m_{k,0}$. While $\mathsf{Mix}(\Xi)$ does not have full rank, we can build upon Theorem A.30 to show that column $(k, 0)$ of $\mathsf{Mix}(\Xi)$ is not contained in the linear span of its other columns.

First, we require a generalization of the fact that every univariate polynomial p of maximum degree d that vanishes at $d+1$ points is in fact the zero polynomial. This follows from Lemma 1.38 about grid interpolation.

Lemma A.39. *Let $p \in \mathbb{Z}[x_1, \dots, x_s]$ be a polynomial with $\deg(p) \leq d$. If $p(\xi) = 0$ holds for all $\xi \in \{0, \dots, d\}^s$, then $p \equiv 0$.* □

From this, we obtain the last missing step.

Lemma A.40. *Let $A^{(t,\rho)}$ denote the column (t, ρ) of the matrix $\mathsf{Mix}(\Xi)$. If $\sum_{t,\rho} \alpha_{t,\rho} A^{(t,\rho)} = 0$ for coefficients $\alpha_{t,\rho} \in \mathbb{Q}$, then $\alpha_{k,0} = 0$.*

Proof. Observe that $\deg(p) \leq K$ holds for every polynomial $p \in \mathsf{Mix}^{\mathfrak{I}}$: All monomials θ appearing in $p \in \mathsf{Mix}^{\mathfrak{O}}$ satisfy $\mathrm{td}(\theta) \leq K$, and it can be verified that substituting $\mathcal{Y}$ into $\mathbf{y}$ yields polynomials of degree $\leq K$. Also recall that $\mathfrak{I} = \mathbb{Z}[\mathbf{x}]$ with $|\mathbf{x}| = 11$.

Assume there were coefficients $\alpha_{t,\rho}$ with $\alpha_{k,0} \neq 0$ and $\sum_{t,\rho} \alpha_{t,\rho} A^{(t,\rho)} = 0$. Then $q = \sum_{t,\rho} \alpha_{t,\rho} \cdot [X^{K-t}]\mathsf{Mix}^{\mathfrak{I}}_{t,\rho}$ vanishes on the grid $\{0, \dots, K\}^{11}$, so $q \equiv 0$ by Lemma A.39. This however contradicts Theorem A.30 because $\alpha_{k,0} \neq 0$. □

Hence, in conclusion, we can set up the system (A.13) with oracle calls to counting $3K$-matchings in general graphs, and we can solve it for the value of $m_{k,0}$, that is, the number of k-partial cycle covers in the original graph G.

A.2.3. Omitted calculations

In this section, we provide the omitted calculations from Section A.2.1.2. To simplify the analysis, we will not consider the polynomials B, U, V, F, but rather related polynomials B, U', V', F', from which the first four polynomials can be obtained via linear combinations. We will furthermore show that such linear combinations preserve algebraic independence.

Let U' be defined as U, with the restriction that only matchings saturating u^{out} are counted in the sum that defines U. Likewise define V' as the restriction of V to matchings

saturating u^{in} and F' as the restriction of F to matchings saturating both u^{out} and u^{in}. Then we clearly have

$$\begin{aligned} U &= B + U' \\ V &= B + V' \\ F &= B + U' + V' + F'. \end{aligned}$$

For $\mathsf{D} \in \{B, U', V', F'\}$, abbreviate $\mathsf{D}_i := [X^i]B$. We require a simple corollary from Theorem A.16:

Corollary A.41. *Let $P = \{p_1, \dots, p_r\}$ be a set of algebraically independent polynomials. For arbitrary $i \in [r]$ and $\lambda_1, \dots, \lambda_r \in \mathbb{Z}$ with $\lambda_i \neq 0$, replace p_i by $\sum_{j=1}^{r} \lambda_j p_j$ to obtain a new set P'. Then P' is algebraically independent.*

Proof. The claim follows from linearity of the differentiation operator together with the invariance of the rank under elementary row operations. □

In the following, we will establish the algebraic independence of the polynomials

$$\mathcal{H} := \{B_1, B_2,\ U'_1, U'_2, U'_3,\ V'_1, V'_2, V'_3,\ F'_2, F'_3, F'_4\},$$

(note that F'_1 is missing in this set), and will then use Corollary A.41, which implies Lemma A.26. Recall that we are given indeterminates

$$\mathbf{w} = (a_\emptyset, a_u, a_v, a_{uv},\ b_\emptyset, b_u, b_v, b_{uv},\ c_u, c_v, c_{uv}).$$

We first abbreviate certain polynomial combinations of these indeterminates. This is only to facilitate reading; no new indeterminates are introduced in this process.

- Let $a^*_u := a_u + a_{uv}$, let $b^*_u := b_u + b_{uv}$ and let $c^*_u := c_u + c_{uv}$.
  Let $a^*_v := a_v + a_{uv}$, let $b^*_v := b_v + b_{uv}$ and let $c^*_v := c_v + c_{uv}$.
- Let $u := a^*_u + b^*_u + c^*_u$ and let $v := a^*_v + b^*_v + c^*_v$.
- Let $a := a_\emptyset + a_u + a_v + a_{uv}$, let $b := b_\emptyset + b_u + b_v + b_{uv}$, and let $c := c_u + c_v + c_{uv}$.
- Let $a_{\mathtt{both}} := a^*_u a^*_v - a_{uv}$, let $b_{\mathtt{both}} := b^*_u b^*_v - b_{uv}$ and let $c_{\mathtt{both}} := c^*_u c^*_v - c_{uv}$.

Extending the abuse of notation from Definition A.24 that identified the indeterminate a_u with a set a_u, we speak of, e.g., the set $a^*_u = a_u + a_{uv}$ as the set $a^*_u = a_u \cup a_{uv}$.

The blocked Venn gadget

If we cannot include u^{in} or u^{out}, we can only match vertices a and b. Thus

$$\begin{aligned} B_1 &= a + b, \\ B_2 &= ab. \end{aligned}$$

The partly blocked Venn gadget

In the matching polynomial U', we consider all matchings of $\mathcal{V}$ that include an edge $\{u^{out}, w\}$ with $w \in V(\mathcal{V})$, but leave u^{in} unsaturated. These matchings can be grouped according to whether w is contained in a, b or c. The same holds for V', but with the roles of u^{out} and u^{in} reversed. We will show in the following that

$$\begin{aligned} U'_1 &= u, \\ U'_2 &= c^*_u B_1 + (a^*_u + b^*_u)(B_1 - 1), \\ U'_3 &= c^*_u B_2 + a^*_u (a-1)b + b^*_u (b-1)a, \end{aligned}$$

$$\begin{aligned} V'_1 &= v, \\ V'_2 &= c^*_v B_1 + (a^*_v + b^*_v)(B_1 - 1), \\ V'_3 &= c^*_v B_2 + a^*_v (a-1)b + b^*_v (b-1)a. \end{aligned}$$

An intuitive explanation for the expressions U'_1, U'_2, U'_3 follows. The argument for V'_1, V'_2, V'_3 is of course symmetric.

1. In U'_1, we can only choose an edge incident with u^{out}, which gives u choices.
2. In U'_2, we distribute two edges e_1, e_2, of which e_1 is incident with u^{out}.
   a) Choose e_1 into c^*_u. The remaining graph is just a blocked gadget, so there are B_1 ways to choose a 1-matching in it.
   b) Choose e_1 into $a^*_u \cup b^*_u$. Now match either the special vertex a or b. Among the total B_1 possible partners, one is already matched by e_1. This gives $(B_1 - 1)$ choices.
3. In U'_3, we distribute three edges e_1, e_2, e_3, of which e_1 is incident with u.
   a) Choose e_1 into c^*_u. The remaining graph is just a blocked gadget, so there are B_2 ways to choose a 2-matching in it.
   b) Choose e_1 into a^*_u. Distribute e_2 and e_3 among a and b. Since e_1 reaches into a, there are $(a-1)b$ choices.
   c) Choose e_1 into b^*_u. Distribute e_2 and e_3 among a and b. Since e_1 reaches into b, there are $(b-1)a$ choices.

The free Venn gadget

In the matching polynomial F', we consider all matchings of $\mathcal{V}$ that include edges $\{u^{in}, w_u\}$ and $\{u^{out}, w_v\}$ with $w_u, w_v \in \mathcal{V}$. These matchings can be grouped according to whether w_u and w_v are each contained in the sets a, b or c.

We write the respective numbers in tabular form: The i-th row corresponds to the vertex u^{out} being matched into the i-th entry of (a^*_u, b^*_u, c^*_u), and likewise for columns and the

vertex u^{in}.

$$
\begin{array}{rcccccccc}
F_1' & = & 0 & & & & & & \\
 & & & a_{\texttt{both}} & + & a_u^* b_v^* & + & a_u^* c_v^* & \\
F_2' & = & + & b_u^* a_v^* & + & b_{\texttt{both}} & + & b_u^* c_v^* & \\
 & & + & c_u^* a_v^* & + & c_u^* b_v^* & + & c_{\texttt{both}} & \\
 & & & a_{\texttt{both}}(a+b-2) & + & a_u^* b_v^*(a+b-2) & + & a_u^* c_v^*(a+b-1) & \\
F_3' & = & + & b_u^* a_v^*(a+b-2) & + & b_{\texttt{both}}(a+b-2) & + & b_u^* c_v^*(a+b-1) & \\
 & & + & c_u^* a_v^*(a+b-1) & + & c_u^* b_v^*(a+b-1) & + & c_{\texttt{both}}(a+b) & \\
 & & & a_{\texttt{both}}(a-2)b & + & a_u^* b_v^*(a-1)(b-1) & + & c_v^* a_u^*(a-1)b & \\
F_4' & = & + & b_u^* a_v^*(a-1)(b-1) & + & b_{\texttt{both}} a(b-2) & + & c_v^* b_u^* a(b-1) & \\
 & & + & c_u^* a_v^*(a-1)b & + & c_u^* b_v^* a(b-1) & + & c_{\texttt{both}} ab &
\end{array}
$$

The formulas can be verified by a simple case distinction, which is similar to the one given for the partly blocked Venn gadget in the previous subsection. We have now computed all polynomials in $\mathcal{H}$.

Using a computer algebra system, such as MATLAB, we can verify that the Jacobian matrix $J\mathcal{H}$ indeed has full rank.

Notes

The author wishes to thank

- Mingji Xia for sharing ideas that were crucial for constructing the Venn gadget,
- Markus Bläser for mentioning algebraic independence in the right moment, and
- an anonymous reviewer, whose comments helped improving the presentation of this material.

B. Existence of k-matching gadgets

The following material appeared in [CM14] and was solely authored by Dániel Marx, without any contribution by the author of the present thesis. This material is included only for the sake of completeness.

In the following, we prove Theorem 6.4 on page 145. Please recall the definition of a k-matching gadget.

Definition B.1. Let H be a graph, M be an induced k-matching in H, and let $C := V(H) \setminus V(M)$. We say that (H, M) is a *k-matching gadget* if whenever an isomorphism f from $H[C]$ to $H[C']$ for some $C' \subseteq V(H)$ satisfies the conditions

(C1) $H \setminus C'$ has no isolated vertex,

(C2) $H \setminus C'$ is bipartite, and

(C3) f is boundary preserving,

then it is also true that $H \setminus C'$ is a k-matching, i.e., $H \setminus C'$ is isomorphic to the graph on $2k$ vertices that contains k vertex-disjoint edges.

It will be convenient to know that if a k-matching gadget exists, then a k_0-matching gadget also exists for every $k_0 < k$. This is not obvious from the definition and requires a nontrivial proof (which also serves as an illustration of Definition B.1 and how it is used, e.g., in the proof of Claim B.6 in the next section).

Lemma B.2. *If (H, M) is a k-matching gadget and $M_0 \subseteq M$ is a k_0-matching, then (H, M_0) is a k_0-matching gadget.*

Proof. Let $C = V(H) \setminus V(M)$ and $C_0 = V(H) \setminus V(M_0)$; we have $C \subseteq C_0$. Let f_0 be an isomorphism from $H[C_0]$ to $H[C_0']$ satisfying (C1)–(C3) of Definition B.1 (see Figure B.1). We have to show that $H \setminus C_0'$ is a matching (and then clearly it is a k_0-matching, since $H \setminus C_0'$ has the same size as $H \setminus C_0$). Let f be the restriction of f_0 to C and let $C' = f(C)$; then f is an isomorphism from $H[C]$ to $H[C']$.

We claim that f satisfies (C1)–(C3) of Definition B.1 with respect to (H, M).

(C1) Suppose that there is an isolated vertex $v \in H \setminus C'$. As $H[C_0' \setminus C']$ is isomorphic to $H[C_0 \setminus C]$, which induces a matching, we have $v \notin C_0'$. By (C1) on f_0, we have that there is no isolated vertex in $H \setminus C_0'$, a contradiction.

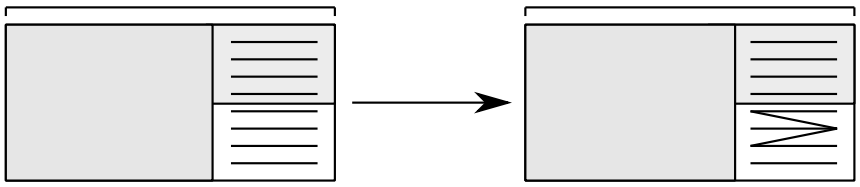


Figure B.1.: Proof of Lemma B.2.

(C2) The graph $H \setminus C'_0$ is bipartite by (C2) on f_0. The graph $H[C'_0 \setminus C']$ is isomorphic to $H[C_0 \setminus C]$, which induces a matching, hence bipartite. There are no edges between $C_0 \setminus C$ and $V(H) \setminus C_0$ (as there are no edges between M_0 and $M \setminus M_0$). Thus the fact that f_0 is boundary preserving implies that there is no edge between $C'_0 \setminus C'$ and $V(H) \setminus C'_0$. It follows that $H \setminus C'$ is bipartite.

(C3) Consider first a vertex $v \in C \setminus \partial_H(C)$. As every neighbor of v is in C, every neighbor of $f(v)$ is in C', that is, $f(v) \in C' \setminus \partial_H(C')$. Consider now a vertex in $v \in \partial_H(C)$. If v has a neighbor $u \in C_0 \setminus C$, then $f(u) \in C'_0 \setminus C'$ is a neighbor of $f(v)$ outside C', implying $v \in \partial_H(C')$. Finally, if v has a neighbor $u \notin C_0$, then $v \in \partial_H(C_0)$ and hence (as f_0 is boundary-preserving by property (C3)) we have $f(v) \in \partial_H(C'_0)$. Since $f(v)$ is in C', we also have $f(u) \in \partial_H(C')$.

As (H, M) is a k-matching gadget and f satisfies (C1)–(C3) of Definition B.1, it follows that $H \setminus C'$ is a matching. As $H[C'_0 \setminus C']$ is isomorphic to the matching $H[C_0 \setminus C]$, this is only possible if $H \setminus C'_0$ is also a matching. □

The following lemma shows as an example a simple condition that guarantees the correctness of a k-matching gadget.

Lemma B.3. *Let M be an induced k-matching in a graph H such that every vertex of $C := V(H) \setminus V(M)$ is adjacent to at most one vertex of $V(M)$. Then (M, H) is a k-matching gadget.*

Proof. Suppose that f is an isomorphism from $H[C]$ to $H[C']$ for some $C' \subseteq V(H)$ satisfying (C1)–(C3) of Definition B.1, but $H \setminus C'$ is not a matching. As $H[C]$ and $H[C']$ are isomorphic, the number of edges in $H[C]$ and $H[C']$ are the same. Every vertex $v \in C$ has at most one edge to $V(H) \setminus C$ and if v has such an edge (that is, $v \in \partial_H(C)$), then (C3) implies that $f(v) \in \partial_H(C')$ has at least one edge to $V(H) \setminus C'$. Therefore, the number of edges between C' and $V(H) \setminus C'$ is at least the number of edges between C and $V(H) \setminus C$. It follows that the number of edges in $H \setminus C'$ is at most the number of edges in $H \setminus C$, that is, at most k. Therefore, the $2k$-vertex graph $H \setminus C'$ has at most k edges and (C1) implies that it does not have isolated vertices.[1] This is only possible if $H \setminus C'$ is a k-matching. □

[1] This the point where we crucially use (C1) of Definition B.1.

The condition in Lemma B.3 can be also formulated as requiring the following two properties:

(P1) no two edges of M have a common neighbor in C and

(P2) the two endpoints of an edge of M have no common neighbor in C.

As we shall see, condition (P1) is usually easy to achieve in the graphs we care about (by making M somewhat smaller), but we will have more difficulty in ensuring condition (P2).

B.1. Bounded-degree graphs

The goal of this section is to prove Theorem 6.4, the existence of k-matching gadgets, for the special case of graph classes $\mathcal{H}$ with bounded maximum degree and unbounded vertex-cover number, or equivalently, containing graphs with arbitrarily large matchings. The results in Sections B.2 and B.3 for other graph classes are based on this result for bounded-degree graphs. The basic idea is that in bounded-degree graphs we are close to the situation described by Lemma B.3: clearly, the two endpoints of an edge in the matching can have only a bounded number of common neighbors; in this sense property (P2) "almost holds." We choose a candidate (H, M) for the k-matching gadget and see how it can fail. If for every C' satisfying (C1)–(C3), the graph $H \setminus C'$ still has many components of size 2 (so it is "almost a matching"), then we can extract a correct k'-matching gadget for some relatively large $k' < k$. Suppose therefore that (H, C) "spectacularly fails": $H \setminus C'$ has only few components of size 2. As $H \setminus C'$ has no isolated vertices, this is only possible if $H \setminus C'$ has many more edges than the k-matching M. Then we argue that now the total degree on the boundary of C' is much smaller than on the boundary of C, and we can use this to find an induced matching in $H \setminus C'$ where the endpoints of the edges have strictly fewer common neighbors than in M. As the graph has bounded degree, repeating this argument a constant number of times eventually leads to a matching where the endpoints of the edges have no common neighbors, hence Lemma B.3 can be invoked.

In a bounded-degree graph, any sufficiently large set of edges contains a large matching and in fact a large induced matching: we can greedily select edges and we need to throw away only a bounded number of edges after each selection. Moreover, in order to move closer to the situation described in Lemma B.3, we may also satisfy the requirement that the selected edges have no common neighbors (but it is possible that the two endpoints of an edge have common neighbors).

Lemma B.4. *Let F be a set of edges in a graph G with maximum degree D.*

1. *There is an induced matching $M' \subseteq F$ of size at least $|F|/(2D^2)$.*
2. *There is an induced matching $M'' \subseteq F$ of size at least $|F|/(2D^3)$ such that every vertex of $V(G) \setminus V(M'')$ is adjacent to at most one edge of M''.*

Proof. (1) Let uv be an edge of F. As the graph has maximum degree at most D, there are less than $2D^2$ edges having an endpoint in the closed neighborhood of $\{u, v\}$. We

perform the following greedy procedure: we select an arbitrary edge uv of F into M' and then remove every edge from F that has an endpoint in the closed neighborhood of $\{u, v\}$. In each step, we reduce the size of F by at most $2D^2$. Therefore, the procedure runs for at least $|F|/(2D^2)$ steps, producing an induced matching of the required size.

(2) Observe that there are less than $2D^3$ edges of F at distance at most 2 from uv. Then a greedy algorithm can produce a set of edges of size at least $|F|/(2D^3)$ such that the edges are at pairwise distance at least 3, that is, M' is an induced matching and every vertex outside M is adjacent to at most one edge of M. □

For bounded-degree graphs, Lemma B.4 implies that there is not much difference between having a large set of edges, a large matching, a large induced matching, or a large induced matching satisfying the requirement that every vertex outside the matching is adjacent to at most one edge of the matching.

Lemma B.5. *There is a function $f_d(k_0, D)$ such that the following holds. If H is a graph with maximum degree at most D and contains a matching of size at least $f_d(k_0, D)$, then there is a k_0-matching gadget (H, M_0).*

Proof. We prove the following statement by induction on c:

> There is a function $f'_d(c, k_0, D)$ such that the following holds. If H is a graph with maximum degree at most D and having a set F of at least $f'_d(c, k_0, D)$ edges such that u and v have at most c common neighbors for every $uv \in F$, then there is a k_0-matching gadget (H, M_0).

If this statement is true for every c, then we can set $f_d(k_0, D) = f'_d(D, k_0, D)$: if the graph has maximum degree of H is D, then it is clear that every edge of a matching has the property that the endpoints have at most D common neighbors.

For $c = 0$, we can set $f'_d(0, k_0, D) = 2k_0D^3$. Then Lemma B.4(2) implies that F contains an induced matching M_0 of size k_0 such that every vertex of $V(H) \setminus V(M_0)$ is adjacent to at most one edge of M_0. Furthermore, $c = 0$ implies that every vertex of $V(H) \setminus V(M_0)$ can be adjacent to at most one endpoint of at most edge of M_0. Therefore, Lemma B.3 implies that (H, M_0) is a k_0-matching gadget.

Suppose now that the statement is true for $c - 1$ with some value of $f'_d(c - 1, k_0, D)$; we show that the statement is true for c with

$$f'_d(c, k_0, D) = 2D^3(5c \cdot f'_d(c - 1, k_0, D) + 100k_0).$$

If F has at least this size, then Lemma B.4(2) implies that there is an induced matching $M \subseteq F$ of size at least $5c \cdot f'_d(c - 1, k_0, D) + 100k_0$ such that every vertex in $V(H) \setminus V(M)$ is adjacent to at most one edge of M.

If (H, M) is a matching gadget, then we are done by Lemma B.2 as $|M| \geq k_0$. Otherwise, let $C := V(H) \setminus V(M)$ and let f be an isomorphism from $H[C]$ to $H[C']$ satisfying (C1)–(C3) of Definition B.1 such that $H \setminus C'$ is not a matching. For any graph G, let us denote by $\kappa_2(G)$ the number of components of G with exactly two vertices. Let us choose f such that $k' := \kappa_2(H \setminus C')$ is minimum possible.

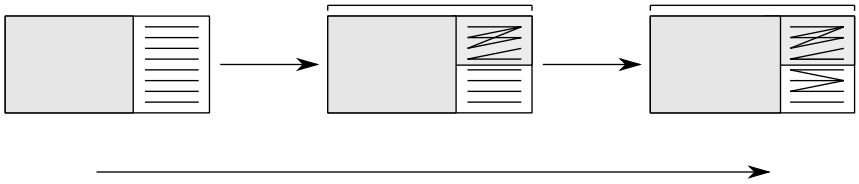


Figure B.2.: Proof of Claim B.6 in Lemma B.5.

Claim B.6. If $k' \geq k_0$, then there is a k_0-matching gadget (H, M_0).

Proof. Let M_0 be the induced matching obtained as the union of all the $k' = \kappa_2(H \setminus C') \geq k_0$ components of $H \setminus C'$ having two vertices each (see Figure B.2). We show that (H, M_0) is a k'-matching gadget; then the existence of a k_0-matching gadget follows from Lemma B.2.

Let $C_0 := V(H) \setminus V(M_0)$; we have $C' \subseteq C_0$. Let us point out that there is no edge between $V(M_0)$ and $V(H) \setminus (C' \cup M_0) = C_0 \setminus C'$: the edges of M_0 are components of $V(H) \setminus C'$. Suppose that f_0 is an isomorphism from $H[C_0]$ to some $H[C_0']$ showing that (H, M_0) is not a k'-matching gadget: f_0 satisfies properties (C1)–(C3), but $H \setminus C_0'$ is not a matching. Let $f^* = f_0 \circ f$ (that is, applying f_0 after f) be the isomorphism from $H[C]$ to $H[C^*]$ with $C^* = f_0(f(C)) = f_0(C') \subseteq C_0'$. Let us verify that f^* satisfies the three conditions of Definition B.1 with respect to (H, M):

(C1) Consider a vertex $v \in H \setminus C^*$. If $v \notin C_0'$, then property (C1) of f_0 implies that v is not isolated in $H \setminus C_0'$ and hence it is not isolated in $H \setminus C^*$ either. Suppose therefore that $v \in C_0' \setminus C^*$, that is, there is a $u \in C_0 \setminus C'$ with $f_0(u) = v$. As $H \setminus C'$ has no isolated vertex, there has to be a neighbor w of u in $H \setminus C'$. If $w \in C_0 \setminus C'$, then $f_0(w)$ is a neighbor of $f_0(u) = v$ in $H \setminus C^*$. Suppose now that $w \notin C_0$, which implies that u is in $\partial_H(C_0)$. Therefore, property (C3) of f_0 implies that $f_0(u)$ has a neighbor outside C_0', hence $f_0(u)$ has a neighbor in $H \setminus C^*$.

(C2) By (C2) on f, the graph $H[C_0 \setminus C]$ is bipartite. Mapping f_0 is an isomorphism between $H[C_0]$ and $H[C_0']$ that maps $C_0 \setminus C$ to $C_0' \setminus C^*$, thus $H[C_0' \setminus C^*]$ is bipartite as well. We have observed above that there is no edge between $C_0 \setminus C'$ and M_0, thus (C3) on f_0 implies that there is no edge between $C_0' \setminus C^*$ and $V(H) \setminus C_0'$. Finally, $V(H) \setminus C_0'$ is bipartite by (C2) on f_0, thus we get that $H \setminus C^*$ is bipartite.

(C3) Let v be a vertex in C. By (C3) on f, we have that $v \in \partial_H(C)$ if and only if $f(v) \in \partial_H(C')$. As there are no edges going from $C_0 \setminus C'$ to outside C_0, we have $\partial_H(C') = \partial_H(C_0)$. Furthermore, by (C3) on f_0, we have that $u \in \partial_H(C_0)$ if and only if $f_0(u) \in \partial_H(C_0')$. Putting together, we have that $v \in \partial_H(C)$ if and only if $f^*(v) = f_0(f(v)) \in \partial_H(C_0')$.

We claim that $\kappa_2(H \setminus C^*) < \kappa_2(H \setminus C')$, contradicting the minimal choice of f. We have stated above that there is no edge between $C_0 \setminus C'$ and $V(M_0)$; as f_0 is boundary preserving, it also follow that there is no edge between $C_0' \setminus C^*$ and $V(H) \setminus C_0'$. Therefore,

the components of $H \setminus C^*$ are exactly the components of $H[C'_0 \setminus C^*]$ and the components of $H \setminus C'_0$. Recall that $H[C'_0 \setminus C^*]$ is isomorphic to $H[C_0 \setminus C']$, which has no 2-vertex component by the definition of M_0. As $H \setminus C'_0$ is not a matching, we have that $H \setminus C'_0$ (and hence $H \setminus C^*$) has strictly less than $|M_0|$ 2-vertex components, that is, $\kappa_2(H \setminus C^*) < \kappa_2(H \setminus C')$. ⌟

In the following, we suppose that $\kappa_2(H \setminus C') < k_0$. Let $s = 10c \cdot f'_d(c-1, k_0, D) + 200k_0$ be the number of vertices of $H \setminus C$ and $H \setminus C'$. Note that $H \setminus C = M$ is an induced matching having exactly $s/2$ edges. We can give a lower bound on the number of edges of $H \setminus C'$ by giving an upper bound on the number of additional edges required to make the graph connected. By (C1) of f, every component has size at least two in $H \setminus C'$.[2] There are less than k_0 components with exactly two vertices, thus by adding at most k_0 edges, we can ensure that every component contains at least three vertices. Then there are at most $s/3$ components with at least three vertices, hence we can connect them with $s/3$ additional edges. It follows that there are at least $s - (s/3 + k_0) = 2s/3 - k_0 \geq 0.6s$ edges in $H \setminus C'$ (using $s \geq 200k_0$). As the number of edges in $H[C]$ and $H[C']$ are the same and $H \setminus C$ has $0.5s$ edges, it follows that the number of edges going out of C' is less than the number of edges going out of C by at least $0.1s$.

Let $B := \partial_H(C)$ and $B' := \partial_H(C')$; by (C3) of Definition B.1, we have $|B| = |B'|$. Recall that every vertex of B has either 1 or 2 edges to $V(M)$. As each of the $s/2$ edges of M has at most c common neighbors in B, there are at most $c(s/2)$ vertices of B with two edges to $V(M)$, implying that there are at most $|B| + c(s/2)$ edges going out of C. Therefore, there are at most $|B| + c(s/2) - 0.1s$ edges going out of C'. Every vertex of $B' = \partial_H(C')$ has at least one edge going to $V(M)$. Let us remove one such edge from each vertex of B', then a set T of at most $c(s/2) - 0.1s$ edges remain.[3] Let $B'_{\geq 2}$ be the subset of B' containing those vertices that have at least two edges going to $V(H) \setminus C'$. Then the total number of edges going from $B'_{\geq 2}$ to $V(H) \setminus C'$ is at most $2|T| \leq cs - 0.2s = (c - 0.2)s$: each vertex of $B'_{\geq 2}$ has at least one edge in T and, in addition to that, can have at most one edge not in T.

As the number of edges between $B'_{\geq 2}$ and $V(H) \setminus C'$ is at most $(c - 0.2)s$, at most $(c - 0.2s)s/c$ vertices of $V(H) \setminus C'$ can have at least c neighbors in $B'_{\geq 2}$. Let X be the set of vertices in $V(H) \setminus C'$ with at most $c - 1$ neighbors in $B'_{\geq 2}$, we have that $|X| \geq s - ((c - 0.2)s)/c = 0.2s/c$. By (C1) of Definition B.1 on f, there are no isolated vertices in $H \setminus C'$. For each vertex in X, let us select an edge of $H \setminus C'$ incident to it; this way, we select a set F^* of least $|X|/2 \geq 0.1s/c \geq f'_d(c-1, k_0, D)$ distinct edges. Consider an edge uv in F^*. Vertices u and v have no common neighbors in $H \setminus C'$: this would contradict (C2) of Definition B.1 stating that $H \setminus C'$ is bipartite.[4] Therefore, every such common neighbor is in $B'_{\geq 2}$. One of u and v is in X, which means that it has at most $c - 1$ neighbors in $B'_{\geq 2}$, implying that u and v can have at most $c - 1$ common neighbors. As $|F^*| \geq f'_d(c-1, k_0, D)$, the induction assumption implies that there is a k_0-gadget. □

[2] This the point where we crucially use (C1) of Definition B.1.
[3] This the point where we crucially use (C3) of Definition B.1: we need that $|B| = |B'|$.
[4] This the point where we crucially use (C2) of Definition B.1.

B.2. Graphs with no large subdivided stars

A *subdivided ℓ-star* consists of a center vertex v and ℓ paths of length 2 starting at v that do not share any vertex other than v. We denote by $\psi(v)$ the largest integer ℓ such that v is the center of a subdivided ℓ-star. We denote by $\psi(G)$ the maximum of $\psi(v)$ for every $v \in V(G)$. The goal of this section is to prove Theorem 6.4, the existence of k-matching gadgets, for graphs where $\psi(G)$ is bounded.

We develop a technology that allows us to "ignore" certain sets Q of vertices: if $H \setminus Q$ has a k-matching gadget, then so does H. This works for sets Q where the vertices have some characteristic property (e.g., based on degrees) that allows us to distinguish them from the vertices not in Q (see below). We use this technique to reduce the problem to bounded-degree graphs. If we have a large induced matching where every vertex has small degree, then we define Q to be the vertices of "large degree." Now $H \setminus Q$ is clearly a bounded-degree graph and hence Lemma B.5 can be invoked. Suppose therefore that we have an induced matching where every vertex has large degree. Then we define Q to be the vertices of "small degree." Somewhat unexpectedly, $H \setminus Q$ is a bounded-degree graph also in this case: this follows from the fact that if $\psi(G)$ is bounded, then a vertex cannot have many neighbors of large degree.

Proposition B.7. *Every vertex $v \in V(G)$ has at most $\psi(v)$ neighbors with degree at least $2\psi(v) + 2$.*

Proof. Let $\ell = \psi(v) + 1$ and suppose that vertex v is adjacent to vertices $\alpha_1, \ldots, \alpha_\ell$, each having degree at least 2ℓ. Then for every $1 \leq i \leq \ell$, as the degree of α_i is at least 2ℓ, we can find a vertex β_i that is adjacent to α_i, but is not in the set $\{v, \alpha_1, \ldots, \alpha_\ell, \beta_1, \ldots, \beta_{i-1}\}$ (note that this set of at most 2ℓ vertices contains less than 2ℓ vertices different from α_i). This creates a subdivided ℓ-star centered at v, a contradiction. □

Therefore, we can reduce the problem to bounded-degree graphs also in the case of a matching with large degree vertices. Finally, if we have a large induced matching with "mixed" edges, that is, each having both a small-degree and a large-degree endpoint, then we can reduce to one of the previous two cases by looking at the common neighbors of the endpoints.

The following definition will be crucial for the clean treatment of the problem. We show that if a set is "well identifiable" (for example, based on degrees etc.) then we can remove it from the graph and it is sufficient to show that the remaining part of the graph has a k-matching gadget. The definition formulates this condition as invariance under certain isomorphisms.

Definition B.8. Let H be a graph and let $X \subseteq C \subseteq V(H)$ be two subsets of vertices. We say that X is a *strong set* with respect to C if whenever f is a boundary-preserving isomorphism from $H[C]$ to $H[C']$ for some $C' \subseteq V(H)$, then $f(X) = X$ (in particular, this implies $X \subseteq C'$).

Observe that $f(X)$ and X have the same size, thus to prove $f(X) = X$, it is sufficient to prove $f(X) \subseteq X$, that is, $v \in X$ implies $f(v) \in X$.

As a simple example, suppose that every vertex in H has either degree at most d or degree at least $d+2k+1$ and $M \subseteq H$ is a k-matching with every vertex having degree at most d in H. Let $C = V(H) \setminus V(M)$ and let $X \subseteq C$ be the set of vertices with degree at least $d+2k+1$. Then X is a strong set: every vertex $x \in X$ has at least $d+2k+1-|V(M)| = d+1$ neighbors in C, hence $f(v)$ has at least $d+1$ neighbors in C', implying that $f(v) \in X$ (as we assumed that degree larger than d implies that the degree is at least $d+2k+1$). In fact, it is sufficient to enforce the degree requirement only for vertices $v \in \partial_H(C)$: it is sufficient if we require that the degree of every vertex in $\partial_H(C)$ is either at most d or at least $d+2k+1$, but the degrees of the vertices in $C \setminus \partial_H(C)$ can be arbitrary. This is sufficient, as if $v \in C \setminus \partial_H(C)$, then every neighbor of v is in C and (C3) of f implies that every neighbor of $f(v)$ is in C', hence (as $H[C]$ and $H[C']$ are isomorphic) vertices v and $f(v)$ have exactly the same degree.

We show now that removing a strong set disjoint from M does not affect whether a k-matching gadget is correct.

Lemma B.9. *Let H be a graph containing an induced k-matching M, let $C := V(H) \setminus V(M)$, and let $X \subseteq C$ be a strong set with respect to C. If $(H \setminus X, M)$ is a k-matching gadget, then so is (H, M).*

Proof. Suppose that f is an isomorphism from $H[C]$ to $H[C']$ for some $C' \subseteq V(H)$ satisfying (C1)–(C3) of Definition B.1, but $H \setminus C'$ is not a matching. Let $H^* = H \setminus X$ and let f^* be the restriction of f to $C \setminus X$. As X is a strong set, we have $f(X) = X$ and hence $f(C \setminus X) = C' \setminus X$, that is, f^* induces an an isomorphism from $H^*[C \setminus X]$ to $H^*[C' \setminus X]$. Observe that $H^* \setminus (C' \setminus X) = H \setminus C'$ has no isolated vertex and bipartite (as f satisfies properties (C1) and (C2)). Furthermore, f^* is boundary preserving (as $\partial_{H \setminus X}(C \setminus X) = \partial_H(C)$ and $\partial_{H \setminus X}(C' \setminus X) = \partial_H(C')$). Therefore, the fact that (H^*, M) is a k-matching gadget implies that $H^* \setminus (C' \setminus X) = H \setminus C'$ is a k-matching, what we had to show. □

Similarly to bounded-degree graphs (Lemma B.4), we can use a bound on $\psi(H)$ to argue that not too many edges can be in the neighborhood of an edge and therefore a large set of edges implies a large induced matching. However, all we need now is that a large induced matching implies that there is a large induced matching such that every vertex outside the matching is adjacent to at most one edge of the matching.

Lemma B.10. *Let M be an induced matching of size at least $2kL^2$ in a graph H with $\psi(G) \leq L$. Then there is an $M' \subseteq M$ of size at least k such that every vertex of $V(G) \setminus V(M')$ is adjacent to at most one edge of M'.*

Proof. Let uv be an edge of M. As M is an induced matching, every other edge of M is at distance at least 2 from uv. We claim that at most $2L^2$ edges of M are at distance exactly 2 from uv. Assume, without loss of generality, that there is a set $M^* \subseteq M$ of at least L^2 edges different from uv at distance exactly 2 from u. If a neighbor w of u is adjacent to L distinct edges of M, then there is a subdivided L-star centered at w, a contradiction. Thus every neighbor of u is adjacent to at most L distinct edges of M, which means that there

are at least $|M^*|/L = L$ neighbors of u, each adjacent to a distinct edge of M^*. Then u is the center of an L-star, a contradiction. Thus we have shown that for every edge of M, there are at most $2L^2$ edges at distance exactly 2 from it. This means that we can greedily select a subset $M' \subseteq M$ of edges at pairwise distance is more than 2, that is, M' is an induced matching and every vertex outside M' is adjacent to at most one edge of M'. □

Recall the example after Definition B.8: if there is a sufficiently large "gap" in the degrees of the vertices of $N(V(M))$ for a matching M, then we can define a strong set simply based on the degrees. The following lemma creates such a gap of arbitrary large size by throwing away at most half of the edges of a matching.

Lemma B.11. *Let F be an induced matching in a graph H with $\psi(H) \leq L$. For every $x \geq 2L+2$, $y \geq 1$, there is an induced matching $F' \subseteq F$ of size at least $|F|/2$ and an $x \leq g \leq x + 4(2L+2)y$ such that $N(V(F'))$ has no vertex whose degree in H is in the range $\{g, \ldots, g+y-1\}$.*

Proof. Let $B := N(V(F))$ and let B_i be the subset of B containing those vertices that have degree exactly i in H and let x_i be the number of edges between B_i and $V(F)$. Let $X_j = \sum_{i=x+jy}^{x+(j+1)y-1} x_i$. Proposition B.7 implies that for every $i \geq x \geq 2L+2$, every vertex in $V(F)$ has at most $2L+2$ neighbors in B_i, thus $\sum_{j=0}^{4(2L+2)-1} X_j = \sum_{i=x}^{x+4(2L+2)y-1} x_i \leq (2L+2) \cdot 2|F|$. Thus by an averaging argument, there is a $0 \leq j^* \leq 4(2L+2)$ such that $X_{j^*} \leq |F|/2$. Therefore, if we construct a matching $F' \subseteq F$ by throwing away every edge of F adjacent to B_i for some $x + j^*y \leq i \leq x + (j^*+1)y - 1$, then we get a matching F' of size at least $|F|/2$ such that no vertex in $N(V(F')) \subseteq B$ (recall that F is an induced matching) has degree in the range $\{x + j^*y, \ldots, x + j^*y + y - 1\}$. That is, the statement is true with $g = x + j^*y$. □

Now we are ready to prove that main result for graphs not having large subdivided stars. The proof uses Lemma B.9 to remove a set of vertices, making the graph bounded degree, and then the bounded-degree result Lemma B.5 can be invoked.

Lemma B.12. *There is a function $f_s(k_0, L)$ such that if graph H with $\psi(H) \leq L$ has an induced matching of size $f_s(k_0, L)$, then there is a k_0-matching gadget (H, M_0).*

Proof. We define the following constants (the function f_d is from Lemma B.5):

$$\begin{aligned} k_H &= 2f_d(k_0, L) \\ D_1 &= 2L + 2 + 8(2L+2)k_H + 2k_H \\ D_2 &= D_1 + 8(2L+2)L \\ k_L &= 2f_d(k_0, D_2) \\ k_X &= 2k_0L^2 + 2D_1^2 k_L + 2L^2 k_H \\ f_s(k_0, L) &= k_H + k_L + k_X. \end{aligned}$$

Let M be an induced matching of size $f_s(k_0, L)$ in H. The edges of M are of three types: either both endpoint have degree at most D_1, or only one of them, or neither. We show that

if M contains a large number of edges from any of the three types, then the k_0-matching gadget exists. In the case when there is a large matching with degrees bounded by D_1, then we identify a strong set containing the high-degree vertices, remove them by applying Lemma B.9, and invoke Lemma B.5 on the remaining bounded-degree graph.

Claim B.13. If there is an induced matching F of k_L edges in H such that the endpoints of these edges have degree at most D_1 in H, then there is a k_0-matching gadget.

Proof. Let us apply Lemma B.11 on F with $x = D_1 + 1$ and $y = 2L$. Then we get a matching $F' \subseteq F$ and a $D_1 + 1 \leq g \leq D_1 + 1 + 8(2L + 2)L = D_2 + 1$ such that there is no vertex in $N(V(F'))$ with degree in the range $\{g, \ldots, g+2L-1\}$. Let Q be the set of vertices with degree at least g (observe that $g \geq D_1 + 1$ implies that $Q \cap V(F) = \emptyset$). We claim that Q is a strong set with respect to $C := V(H) \setminus V(F')$. Let f be a boundary-preserving isomorphism from $H[C]$ to $H[C']$. Consider a vertex $v \in Q$. If $v \notin \partial_H(C)$, then every neighbor of v is in C, hence $f(v)$ has at least g neighbors in C', implying $f(v) \in Q$. Suppose now that $v \in \partial_H(C)$. If v is adjacent to $L + 1$ distinct edges of F', then v is the center of an $(L + 1)$-star, a contradiction. Therefore, there are at most $2L$ edges between v and $V(F')$. As $v \in N(V(F'))$, the way F' was defined by Lemma B.11 ensures that the degree of v is at least $g + 2L$, thus it has at least g neighbors inside C. It follows that $f(v)$ has at least g neighbors inside C' and hence $f(v) \in Q$. We have shown that $v \in Q$ implies $f(v) \in Q$ and hence Q is a strong set. The graph $H \setminus Q$ has maximum degree at most $g - 1 \leq D_1 + 8(2L + 2)L = D_2$ and has a matching F' of size $|F'| \geq |F|/2 = f_d(k_0, D_2)$. Therefore, Lemma B.5 implies that there is a k_0-matching gadget in $H \setminus Q$ and then it follows by Lemma B.9 that H has a k_0-matching gadget as well. ⌟

In the case when every vertex of the matching has high degree in H, then we identify a strong set containing all the low-degree vertices. We argue that after removing this strong set, the remaining graph has bounded degree: as we have a bound on $\psi(H)$, Proposition B.7 implies that a vertex cannot have many high-degree neighbors. Therefore, we are again in a situation when Lemma B.5 can be invoked.

Claim B.14. If there is an induced matching F of k_H edges in H such that the endpoints of these edges have degree at least D_1, then there is a k_0-matching gadget.

Proof. Let us apply Lemma B.11 on F with $x = 2L + 2$ and $y = 2|F|$. Then we get a matching $F' \subseteq F$ and a $2L + 2 \leq g \leq 2L + 2 + 8(2L + 2)|F|$ such that there is no vertex in $N(V(F'))$ with degree in the range $\{g, \ldots, g + 2|F| - 1\}$. Let Q be the set of vertices with degree less than $g + 2|F|$ (observe that $g + 2|F| \leq 2L + 2 + 8(2L + 2)|F| + 2|F| = D_1$ implies that $Q \cap V(F) = \emptyset$). We claim that Q is a strong set with respect to $C := V(H) \setminus V(F')$. Let f be a boundary-preserving isomorphism from $H[C]$ to $H[C']$. Consider a vertex $v \in Q$. If $v \notin \partial_H(C)$, then every neighbor of v is in C. As f is boundary preserving, we have $f(v) \notin \partial_H(C')$, hence every neighbor of $f(v)$ is in C'. Furthermore, f is an isomorphism between $H[C]$ and $H[C']$, hence $f(v)$ has the same degree as v, that is, less than $g + 2|F|$, implying $f(v) \in Q$. Suppose now that $v \in \partial_H(C)$. By the way F' was defined by Lemma B.11, the degree of v is actually less than g, thus it has at most g

neighbors inside C. It follows that $f(v)$ has at most g neighbors inside C'. Clearly, v can have at most $2|F'| \leq 2|F|$ neighbors outside C', hence $f(v) \in Q$. We have shown that $v \in Q$ implies $f(v) \in Q$ and hence Q is a strong set. The graph $H \setminus Q$ has maximum degree at most L: otherwise H has a vertex with more than L neighbors with degree at least $g + 2|F| \geq 2L + 2$ adjacent to it, contradicting Proposition B.7. As F' is a matching of size $|F'| \geq |F|/2 = f_d(k_0, L)$, Lemma B.5 implies that there is a k_0-matching gadget in $H \setminus Q$. It follows by Lemma B.9 that H has a k_0-matching gadget as well. ⌟

The remaining case is when we have a large induced matching where each edge has both a high-degree and a low-degree endpoint. If we have many edges where the endpoints have no common neighbors, then we can invoke Lemma B.3. Otherwise, if many of the edges have low-degree common neighbors, then we can use this common neighbor and the low-degree endpoint to create a matching where every vertex has low degree and apply Claim B.13. Similarly, if many of the edges have high-degree common neighbors, then we can create a matching with high-degree vertices and apply Claim B.14.

Claim B.15. If there is a matching F of k_X edges in H such that the every edge in F has an endpoint with degree at most D_2 and a degree at least D_1, then there is a k_0-matching gadget.

Proof. Let $F_0 \subseteq F$ contain those edges uv in F for which u and v have no common neighbors. If $|F_0| \geq 2k_0L^2$, then Lemma B.10 implies that there is an induced matching $F_0' \subseteq F_0$ of size k_0 such that every vertex of $V(H) \setminus V(F_0')$ is adjacent to at most one edge of F_0'. In fact, as the endpoints of the edges in F_0' have no common neighbors, every vertex of $V(H) \setminus V(F_0')$ is adjacent to at most one vertex of $V(F_0')$. Then Lemma B.3 implies that there is a k_0-matching gadget.

Suppose therefore that $F \setminus F_0$ has size at least $k_X - 2k_0L^2 = 2D_1^2k_L + 2Lk_H$. For every edge $e \in F$, let us pick a common neighbor w_e of the endpoints of e and let us partition $F \setminus F_0$ into F_L and F_H depending on whether w_e has degree less than D_1 or at least D_1, respectively. If $|F_L| \geq 2D_1^2k_L$, then, for every $e \in F_L$, let F_L^* contain the edge between w_e and the endpoint of e with degree at most D_1. As every vertex of F_L^* has degree at most D_1 in H, Lemma B.4(1) implies that we can select a subset of F_L^* of size at least $|F_L^*|/(2D_1^2) \geq k_L$ that forms an induced matching, and then Claim B.13 implies that there is a k_0-matching gadget.

Otherwise, we have that $|F_H| \geq 2L^2k_L$ and, for every $e \in F_H$, we let F_H^* contain the edge between w_e and the endpoint of e with degree at least D_1. As every endpoint of every edge in F_H^* has degree at least $D_1 \geq 2L + 2$, Proposition B.7 implies that the graph induced by these vertices has maximum degree at most L. Thus by Lemma B.4(1), we can select a subset of F_H^* that forms an induced matching of size at least $|F_L^*|/(2L^2) \geq k_H$, and then Claim B.14 implies that there is a k_0-matching gadget. ⌟

Claims B.13–B.15 prove the lemma: if M has size at least $f_s(k_0, L)$, then at least one the three cases hold. □

B.3. Bounded-treewidth graphs

In this section, we complete the proof of Theorem 6.4 by showing that if a bounded-treewidth graph has large vertex-cover number, then it contains a k-matching gadget. The proof presented below finds an induced matching such that $\psi(v)$ is bounded for every vertex v of the matching, that is, there are no large subdivided stars centered on them. Then we define Q to be the set of vertices with large ψ-number (this require some care) and use the technology developed in Section B.2 (Lemma B.9) to argue that it is sufficient to find a k-matching gadget in $H \setminus Q$. Clearly, $\psi(H \setminus Q)$ is bounded, hence Lemma B.12 can be invoked.

Lemma B.16. *Let w and k be integers and let H be a graph of treewidth at most w and vertex cover number greater than $3k(w+1)$. Then there is an induced matching $M = \{u_1v_1, \ldots, u_kv_k\}$ such that $\psi(u_i), \psi(v_i) \leq 2(w+1)$ for every $1 \leq i \leq k$.*

Proof. Consider a rooted tree decomposition (T, B) of H having width w. For every $t \in V(T)$, we denote by H_t the graph induced by the union of every bag $B_{t'}$ for every descendant t' of t (including t itself). For $i = 0, 1, \ldots,$ we iteratively construct an induced matching M_i and a subset X_i of $V(T)$; initially, $M_0 = X_0 = \emptyset$. We define (see Figure B.3)

- $S_i := \bigcup_{t \in X_i} B_t$,
- $V_i := \bigcup_{t \in X_i} \bigcup_{t' \text{ is a descendant of } t} B_t$,
- $\widehat{X}_i \subseteq X_i$ to be the maximal elements of X_i (i.e., those vertices of X_i that have no proper ancestor in X_i), and
- $\widehat{S}_i := \bigcup_{t \in \widehat{X}_i} B_t$.

Note that if a vertex v of V_i has a neighbor outside V_i, then $v \in \widehat{S}_i$, that is, $\widehat{S}_i$ separates $V_i \setminus \widehat{S}_i$ from $V(H) \setminus V_i$.

We maintain the following invariant properties:

1. $H[V_i \setminus S_i]$ has a vertex cover C_i of size at most $(w+1)(3|M_i| - |X_i| - |\widehat{X}_i|)$.
2. Each edge in M_i is in a different component of $H[V_i \setminus S_i]$ (in particular, S_i is disjoint from $V(M_i)$).
3. $\psi(u), \psi(v) \leq 2(w+1)$ for every edge $uv \in M_i$.

For $i = 0$, all three conditions hold vacuously. If $|M_{i-1}| \geq k$, then we stop the process: a subset of k edges of M_{i-1} is the required induced matching. Otherwise, given M_{i-1} and X_{i-1}, we compute M_i and X_i the following way. Let us choose a node t^* such that $H_{t^*} \setminus (B_{t^*} \cup V_{i-1})$ contains at least one edge and the distance of t^* from the root r is maximum possible. We claim that at least one such node exists; in particular, the root r is such a node. Otherwise, if $H_r \setminus (B_r \cup V_{i-1})$ has no edge, then $B_r \cup S_{i-1} \cup C_{i-1}$ is covers of $H_r = H$ (note that $\widehat{S}_{i-1} \subseteq S_i$ covers every edge between V_{i-1} and $V(H) \setminus V_{i-1}$, and C_{i-1} is covers every edge in $H[V_{i-1} \setminus S_{i-1}]$) and its size is at most $w + 1 + (w+1)|X_{i-1}| +$

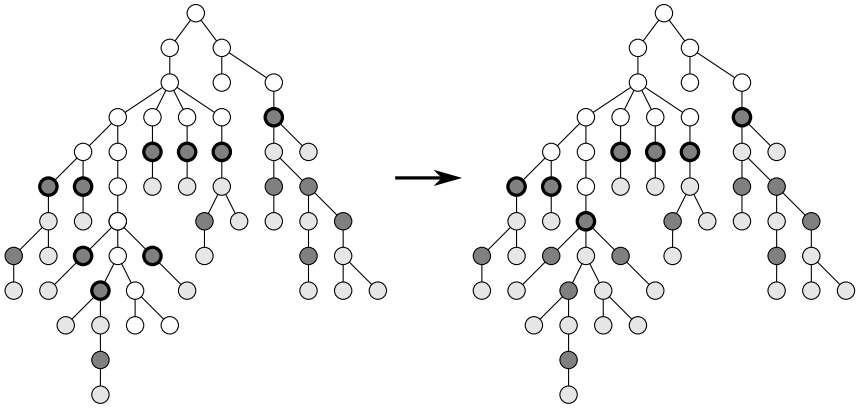


Figure B.3.: Case 1 in the proof of Lemma B.16 with $p = 4$ and $j = 1$. On the left, shaded nodes show V_{i-1}, dark shaded nodes show X_{i-1}, dark circled nodes show $\widehat{X}_{i-1}$. We have $|X_i| = |X_{i-1}| + 1$ and $|\widehat{X}_i| = |\widehat{X}_{i-1}| - 2 < |\widehat{X}_{i-1}|$.

$(w+1)(3|M_{i-1}| - |X_{i-1}| - |\widehat{X}_{i-1}|) \leq 3k(w+1)$ (using property 1 on the size of C_{i-1} and $|M_{i-1}| < k$), contradicting our assumption on the vertex cover number of H. Let $t_1, \ldots,$ t_p be the children of t^* for which $H_{t_j} \setminus (B_{t^*} \cup V_{i-1})$ contains at least one edge. Note that $p \geq 1$: if there is an edge in $H_{t^*} \setminus (B_{t^*} \cup V_{i-1})$, then it has to appear in H_{t_j} for some child t_j of t^*. We consider two cases.

Case 1: there is a t_j that has more than one descendants in $\widehat{X}_{i-1}$ (see Figure B.3). Let $\widehat{X}'_{i-1}$ be the descendants of t_j in $\widehat{X}_{i-1}$. Let us find a node t that is at maximum distance from the root and has at least two descendants in $\widehat{X}'_{i-1}$ (it is possible that $t = t_j$). In other words, for every pair of nodes in $\widehat{X}'_{i-1}$, we find the least common ancestor of the two nodes and we take t to be a node of maximum distance from the root among these common ancestors. We let $M_i = M_{i-1}$ and $X_i = X_{i-1} \cup \{t\}$. Observe that t_j is an ancestor of t and therefore t^* is a proper ancestor of t. Thus by the choice of t^*, there is no edge in $H_t \setminus (B_t \cup V_{i-1})$. Therefore, $C_i := C_{i-1}$ is a vertex cover of $H[V_i \setminus S_i]$: edges with both endpoint in V_{i-1} are covered by C_i, edges with exactly one endpoint in V_{i-1} have one endpoint in $\widehat{S}_{i-1} \subseteq S_i$, and edges with both endpoints in $V_i \setminus V_{i-1}$ have one endpoint in $B_t \subseteq S_i$. Observe that the descendants of t in $\widehat{X}'_{i-1}$ (there are at least two such nodes) are no longer maximal nodes in $\widehat{X}_i$ after adding t and we have added only one new node to X_i, hence $|\widehat{X}_i| \leq |\widehat{X}_{i-1}| - 1$. Together with $|M_i| = |M_{i-1}|$ and $|X_i| = |X_{i-1}| + 1$, this implies that $3|M_i| - |X_i| - |\widehat{X}_i| \geq 3|M_{i-1}| - |X_{i-1}| - |\widehat{X}_{i-1}|$. Therefore, C_i satisfies the size bound of property 1. For property 2, observe first that $S_i \cap V_{i-1} \subseteq S_{i-1}$ (if a vertex appears in B_t and V_{i-1}, then it has to appear in a bag of $\widehat{X}_{i-1}$) and therefore the fact that S_{i-1} is disjoint from $V(M_{i-1})$ implies that S_i disjoint from $V(M_i)$. Together with $S_{i-1} \subseteq S_i$, it follows that the edges in M_i are indeed in different components of $H[V_i \setminus S_i]$. Property 3 follows from $M_i = M_{i-1}$.

Case 2: Every t_j has at most one descendant in $\widehat{X}_{i-1}$ (Figure B.4). For every $1 \leq j \leq p$, we let $u_j v_j$ to be an edge of $H_{t_j} \setminus (B_{t^*} \cup V_{i-1})$. We let $M_i = M_{i-1} \cup \bigcup_{j=1}^{p} \{u_j v_j\}$ and $X_i = X_{i-1} \cup \{t^*\}$. To prove property 1, we show that $C_i := C_{i-1} \cup \bigcup_{j=1}^{p} B_{t_j}$ is a vertex

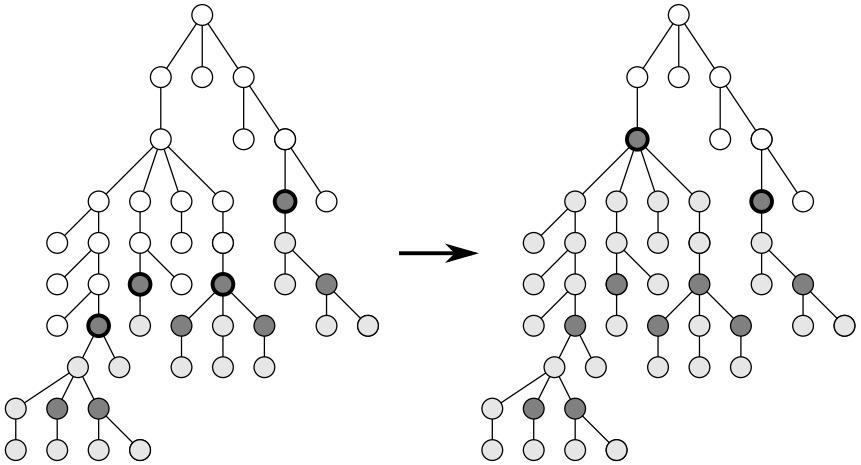


Figure B.4.: Case 2 in the proof of Lemma B.16 with $p = 3$. On the left, shaded nodes show V_{i-1}, dark shaded nodes show X_{i-1}, circled nodes show $\widehat{X}_{i-1}$. We have $|X_i| = |X_{i-1}| + 1$ and $|\widehat{X}_i| = |\widehat{X}_{i-1}| - 2 \leq |\widehat{X}_{i-1}| + 1$.

cover of $H[V_i \setminus S_i]$. Consider an edge $u'v'$ of $H[V_i \setminus S_i]$ not covered by C_i. If both u' and v' are in $V_{i-1} \setminus S_{i-1}$, then $u'v'$ is covered by C_{i-1}; if, say, $u' \in V_{i-1}$ and $v' \notin V_{i-1}$, then $u' \in \widehat{S}_{i-1} \subseteq S_{i-1}$. Therefore, we may assume that $u'v'$ is an edge of $H \setminus V_{i-1}$. Neither u' nor v' can be in $B_{t^*} \subseteq S_i$. Thus $u'v'$ is an edge of $H_{t^*} \setminus (B_{t^*} \cup V_{i-1})$. By the way we defined $t_1, \ldots, t_p$, this means that $u'v'$ is an edge of $H_{t_j} \setminus (B_{t^*} \cup V_{i-1})$ for some $1 \leq j \leq p$. As $B_{t_j} \subseteq C_i$, it is in fact an edge of $H_{t_j} \setminus (B_{t_j} \cup V_{i-1})$ as well. However, this contradicts the choice of t^*. Let us prove that the size bound of property 1 holds for C_i. As we add only the single new node t^* to X_i, we have $|X_i| = |X_{i-1}| + 1$ and $|\widehat{X}_i| \leq |\widehat{X}_{i-1}| + 1$ (note that the equality $|\widehat{X}_i| = |\widehat{X}_{i-1}| + 1$ is possible, but only if t^* has no descendant in X_{i-1}). Together with $|M_i| = |M_{i-1}| + p$, it follows that

$$\begin{aligned}
|C_i| &\leq |C_{i-1}| + p(w+1) \\
&\leq (w+1)(3|M_{i-1}| - |X_{i-1}| - |\widehat{X}_{i-1}|) + p(w+1) \\
&\leq (w+1)(3(|M_i| - p) - (|X_i| - 1) - (|\widehat{X}_i| - 1)) + p(w+1) \\
&= (w+1)(3|M_i| - |X_i| - |\widehat{X}_i| - 2p + 2) \\
&\leq (w+1)(3|M_i| - |X_i| - |\widehat{X}_i|),
\end{aligned}$$

where we used the property 1 on C_{i-1} in the first inequality and $p \geq 1$ in the last inequality.

To prove property 2, observe first that every $u_j v_j$ is disjoint from B_{t^*} by definition and $B_{t^*} \cap V_{i-1} \subseteq \widehat{S}_{i-1}$, thus S_i is disjoint from $V(M_i)$. Notice that $\widehat{S}_{i-1} \subseteq S_{i-1} \subseteq S_i$ separates V_{i-1} from $V_i \setminus V_{i-1}$ and every edge of $M_i \setminus M_{i-1}$ is in $V_i \setminus V_{i-1}$ Therefore, no edge of M_{i-1} can be in the same component of $H \setminus S_i$ as an edge $M_i \setminus M_{i-1}$. Furthermore, the edges of $M_i \setminus M_{i-1}$ are separated by $B_{t^*} \subseteq S_i$. To prove property 3, suppose that, for $\ell = 2(w+1) + 1$, there is a subdivided ℓ-star centered at u_j (the argument is the same for

v_j); let $u_j\alpha_1\beta_1, \ldots, u_j\alpha_\ell\beta_\ell$ be the paths of length 2 starting at u_j. Node t_j can have at most one descendant in $\widehat{X}_{i-1}$. If there is such a descendant $x_j \in \widehat{X}_{i-1}$, then there is an $1 \leq q \leq \ell$ such that $\alpha_q, \beta_q \notin B_{t_j} \cup B_{x_j}$; if there is no such descendant, then let us choose q such that $\alpha_q, \beta_q \notin B_{t_j}$. It follows that α_q and β_q only appear in bags that are proper descendants of t_j, but they do not appear in any bag of X_{i-1}, i.e., $\alpha_q, \beta_q \notin V_{i-1}$. It follows that $\alpha_q\beta_q$ is an edge of $H_{t_j} \setminus (B_{t_j} \cup V_{i-1})$, contradicting the selection of node t^* and the edge uv. Thus we have $\psi(u_j), \psi(v_j) \leq 2(w+1)$, as required by property 3. □

The following two technical lemmas will be used in the proof of Lemma B.19.

Lemma B.17. *If $\mathcal{H}$ is a multiset of at least $(1 + z \cdot r)k$ subsets of a universe U, each having size at most r, then there is a subcollection $\mathcal{H}' \subseteq \mathcal{H}$ of size k such that for every $x \in U$, either there is at most one set in $\mathcal{H}'$ containing x, or there are at least z sets in $\mathcal{H} \setminus \mathcal{H}'$ containing x.*

Proof. We prove the statement by induction on k. Let us select an arbitrary set $X \in \mathcal{H}$. For every $x \in X$, let us arbitrarily select z sets of $\mathcal{H} \setminus \{X\}$ that contain x (or all of them, if there are less than z such sets); we define $\mathcal{H}_X$ as these selected sets; we have $|\mathcal{H}_X| \leq z \cdot r$. Let us apply the induction hypothesis on the multiset $\mathcal{H}_{k-1} := \mathcal{H} \setminus (\mathcal{H}_X \cup \{X\})$ and $k-1$ (note that $\mathcal{H}_{k-1}$ has size at least $(1 + z \cdot r)(k-1)$); let $\mathcal{H}'_{k-1}$ be the resulting subcollection of $k-1$ sets. We claim that $\mathcal{H}' = \mathcal{H}'_{k-1} \cup \{X\}$ is the desired collection of k sets. Indeed, for every vertex $x \in X$, if x appears in a set of $\mathcal{H}'$, then it appears in at least z sets of $\mathcal{H}_X \subseteq \mathcal{H} \setminus (\mathcal{H}'_{k-1} \cup \{X\}))$ and the statement is true for every $x \notin X$ by the induction hypothesis. □

Lemma B.18. *Let Z be a set of vertices in a graph H of treewidth at most w. If for every $v \in Z$ there is a subdivided star S_v centered at v covering every vertex of Z, then $|Z| \leq w + 1$.*

Proof. Consider a rooted tree decomposition (T, B) of H. For every $t \in V(T)$, we denote by V_t the union of every bag $B_{t'}$ for every descendant t' of t (including t itself). For every vertex $v \in Z$, consider the node t_v closest to the root that contains v, and let us select a $v \in Z$ such that t_v has maximum distance from the root. Then $B_{t'} \cap Z \subseteq B_{t_v} \cap Z$ for every proper descendant t' of t_v, otherwise there would be a vertex $u \in Z$ such that t_u is a proper descendant of t_v. The subdivided star S_v covers Z by assumption, hence there is a path of length at most two between v and each vertex of $Z \setminus \{v\}$ such that v is the only vertex shared by these paths. We claim that each such path has to contain a vertex of $B_{t_v} \setminus \{v\}$: otherwise, the vertices of the path appear either only in bags that are proper descendants of t_v (contradicting the maximality of t_v) or only in bags where v does not appear (contradicting that a vertex of the path is a neighbor of v). Therefore, we have $|Z \setminus \{v\}| \leq |B_{t_v} \setminus \{v\}| \leq w$ and $|Z| \leq w + 1$ follows. □

We are now ready to prove the main result for bounded-treewidth graphs, which completes the proof Theorem 6.4.

Lemma B.19. *There is a function $f(k,w)$ such that if a graph H with treewidth at most w has vertex cover number greater than $f(k,w)$, then there is a k-matching gadget (H,M).*

Proof. We define the following constants (the function f_s is from Lemma B.12):

$$\begin{aligned} L &= 2(w+1) \\ r &= 2L + 2(L(w+2)+1) \\ L_1 &= 2L+2 \\ L_2 &= 4r + 2 + L_1 \\ z &= L_2 \\ k_2 &= f_s(k, L_2) \\ k_1 &= 2k_2 \\ k_0 &= (1+z\cdot r)k_1 \\ f(k,w) &= 3k_0(w+1). \end{aligned}$$

By Lemma B.16, if H has vertex cover number greater than $f(k,w)$, then H has an induced matching M_0 of size k_0 such that $\psi_H(v) \leq L$ for every $v \in V(M)$.

For every $v \in V(H) \setminus V(M_0)$, let us fix a subdivided star S_v centered at v with $\min\{\psi_H(v), L_2\}$ leaves and having the minimum number of vertices in $V(M_0)$. For every $e \in M_0$, we let $v \in V(H) \setminus V(M_0)$ be in X_e if $\psi_H(v) \geq L_1$ and S_v uses an endpoint of e.

Claim B.20. For every $e \in M_0$, $|X_e| \leq r$.

Proof. Suppose first that $v \in X_e$ and v is adjacent to e. As v has degree at least $\psi_H(v) \geq L_1 = 2L+2$, Proposition B.7 implies that each endpoint of e can have at most L such neighbors, hence there are at most $2L$ such vertices in X_e. Suppose therefore that $|X_e|$ has at least $r - 2L = 2(L(w+2)+1)$ vertices not adjacent to e. For each such vertex $v \in X_e$, the subdivided star S_v contains one or two paths vxy with y being one of the two endpoints of e. Let us fix such an x and y for each vertex $v \in X_e$ not adjacent to e; for at least $L(w+2)+1$ vertices we have the same y. If there are $L+1$ such vertices $v \in X_e$ with distinct x's, then this shows that there is a subdivided $L+1$ star centered at y, a contradiction. Therefore, there are $w+2$ vertices $v_1, \ldots, v_{w+2}$ in X_e sharing the same x. If S_{v_i} does not use vertex v_j for some $j \neq i$, then we can replace y by v_j in the subdivided star S_{v_i}, contradicting the minimality of S_{v_i} with respect to the number of vertices of $V(M_0)$ used. Thus every S_{v_i} covers the set $Z = \{v_1, \ldots, v_{w+2}\}$, contradicting Lemma B.18. ⌟

We construct a matching $M_1 \subseteq M_0$ the following way. Let $\mathcal{H}$ be the multiset containing X_e for every $e \in M_0$; we have $|\mathcal{H}| = |M| = (1+z\cdot r)k_1$. Let us invoke Lemma B.17 to obtain a subcollection $\mathcal{H}'$ of size k_1 such that for every $x \in V(H) \setminus V(M_0)$, either there is at most one set in $\mathcal{H}'$ containing x or at least z sets of $\mathcal{H} \setminus \mathcal{H}'$ contain x. Let $M_1 \subseteq M_0$ be the subset of k_1 edges corresponding to the subcollection $\mathcal{H}'$.

Claim B.21. If $v \in X_e$ for at least two different $e \in M_1$, then $\psi_{H \setminus V(M_1)}(v) \geq L_2$.

Proof. By the way the subcollection $\mathcal{H}'$ is constructed, we have that v is in X_e for at least $z = L_2$ edges $e \in M \setminus M_1$; let $F \subseteq M \setminus M_1$ be this set of edges. Let vx_1y_1, ..., $vx_\ell y_\ell$ be the paths in the subdivided star S_v. Every edge of F is intersected by one or two of these paths. Furthermore, as M_0 is an induced matching, if vx_iy_i intersects an edge of $F \subseteq M \setminus M_1$, then it cannot intersect $V(M_1)$. Therefore, at least $|F| \geq L_2$ such paths are disjoint from $V(M_1)$. These paths form a subdivided L_2-star centered at v, implying $\psi_{H\setminus V(M_1)}(v) \geq L_2$. ⌟

Let $v \in V(H) \setminus V(M_1)$ be in V_i if $\psi_H(v) = i$ and $v \in X_e$ for some $e \in M_1$. As $|X_e| \leq r$ for every $e \in M_e$ (Claim B.20), we have $\sum_{i=L_1}^{L_2-2} |V_i| \leq rk_1$. Therefore, there is an $L_1 \leq i^* \leq L_2 - 2$ such that $|V_{i^*}| + |V_{i^*+1}| \leq 2rk_1/(L_2 - 2 - L_1) = k_1/2$. Let M_2 contain every $e \in M_1$ with $X_e \cap (V_{i^*} \cup V_{i^*+1}) = \emptyset$. Note that every $v \in V_{i^*} \cup V_{i^*+1}$ is in X_e for at most one $e \in M_1$: otherwise, Claim B.21 implies that $\psi_H(v) \geq \psi_{H\setminus V(M_1)}(v) \geq L_2$, contradicting the choice $i^* \leq L_2 - 2$. Therefore, we have that the size of M_2 is at least $k_1 - (|V_{i^*}| + |V_{i^*+1}|)| \geq k_1/2 = k_2$. Furthermore, if $e \in M_2$, then X_e is disjoint from $V_{i^*} \cup V_{i^*+1}$.

Let us define Q as the set containing every $v \in V(H)$ with $\psi(v) \geq i^*$ and let $C = V(H) \setminus V(M_2)$.

Claim B.22. Set Q is a strong set of H with respect to C.

Proof. Let f be a boundary-preserving isomorphism from $H[C]$ to $H[C']$ for some $C' \subseteq V(H)$. We show that $\psi_{H[C]}(v) \geq i^*$ for every $v \in Q$. Then, as f is an isomorphism between $H[C]$ and $H[C']$, we have that $\psi_H(f(v)) \geq \psi_{H[C']}(f(v)) = \psi_{H[C]}(v) \geq i^*$, implying $f(v) \in Q$.

Consider a vertex $v \in Q$ and the subdivided S_v star. Recall that, as we have $\psi_H(v) \geq i^* \geq L_1$, the subdivided star S_v contains an endpoint of $e \in M_0$ only if $v \in X_e$. Suppose first that $v \in Q$ is not in X_e for any $e \in M_2$; in particular, this is the case for any v with $\psi_H(v) \in \{i^*, i^* + 1\}$. Then the subdivided star S_v is disjoint from $V(M_2)$, that is, S_v is fully contained in C. Then $\psi_{H[C]}(v) \geq \min\{\psi_H(v), L_2\} \geq i^*$.

Suppose now that $v \in Q$ is in X_e for exactly one edge $e \in M_2$; this is only possible if $\psi_H(v) \geq i^* + 2$. Then the subdivided star S_v intersects the endpoints of at most one edge of M_2, that is, at most two paths of S_v intersect $V(M_2)$. Therefore, a subdivided star centered at v with $\min\{\psi_H(v), L_2\} - 2 \geq i^*$ leaves appears in $H[C]$.

Finally, suppose that $v \in Q$ is in X_e for at least two edges of $M_2 \subseteq M_1$. Then Claim B.21 implies that $\psi_{H[C]}(v) \geq \psi_{H\setminus V(M_1)}(v) \geq L_2 \geq i^*$. ⌟

We have $\psi(H \setminus Q) < i^* < L_2$ and M_2 is a matching of size at least k_2 in $H \setminus Q$. Therefore, Lemma B.12 implies that there is a k-matching gadget $(H \setminus Q, M)$. As Q is a strong set, Lemma B.9 implies that (H, M) is also a k-matching gadget. □

C. Computing matchgate signatures

In several instances, we claim that specific matchgates realize certain signatures, and subsequent arguments rely crucially upon the correctness of these claims.

How the matchgates were found

Each of these matchgates was found in the following way:

1. Assume we wish to find a matchgate on n vertices, for $n \in \mathbb{N}$, that realizes a given signature $f : \{0,1\}^{[d]} \to \mathbb{Q}$. If n is not known, we simply repeat the following for all $n \in \mathbb{N}$ until the first matchgate is found.

2. Let $\mathbf{x} = (x_{ij})_{i,j \in [n]}$ be a family of indeterminates. Consider the complete graph $K = K_n$, where each edge $ij \in E(K_n)$ is weighted by $x_{i,j}$. If we wish to ensure a property such as planarity, we can choose a base graph other than K_n. For each $i \in [d]$, add a dangling edge to the i-th vertex in K.

3. Compute $\mathrm{Sig}(K) : \{0,1\}^{[d]} \to \mathbb{Z}[\mathbf{x}]$ by brute force. Note that the signature entries are multivariate polynomials over the indeterminates $\mathbf{x}$.

4. Set up a system of polynomial equations by requiring, for each $y \in \{0,1\}^{[d]}$, that

$$\mathrm{Sig}(K, y) = f(y).$$

   Solve the resulting system of 2^d equations using a licensed copy of a computer algebra system, such as MAPLE in our case. Note that the parity condition (Remark 2.19) implies that the left-hand sides of 2^{d-1} equations are zero.

Similar approaches were already carried out in the literature on Holant problems [Val08, CLX09a], but these did not result in the matchgates we presented.

How to verify the matchgates

We provide MATLAB code for computing, given a gate Γ, the signature of Γ. This code uses only a basic fragment of MATLAB and is optimized for maximum transparency. As a consequence, it is sinfully inefficient, but can still handle our inputs within reasonable time. Note that this code only allows for *verifying* Γ rather than finding it.

Let Γ be a matchgate on vertex set $[n]$ with m edges, d of which are dangling. Assume that $E(\Gamma)$ is ordered as $e_1, \ldots, e_m$ such that $D = \{e_1, \ldots, e_d\}$ are the dangling edges. Let 0 be a virtual vertex that all dangling edges are incident to.

- The input to `computeSignature` is (A, n, m, d), where A is an array of size $m \times 3$. For $i \in [m]$, its i-th row encodes edge $e_i = uv$ as a triple $(u, v, w(e_i))$. Note that dangling edges are of the format $(0, \star, 1)$. When called with the input (A, n, m, d) derived from Γ, the function outputs $\mathrm{Sig}(\Gamma)$ on the console. To save some paper, it only outputs those entries x with $\mathrm{Sig}(\Gamma, x) \neq 0$. The function uses a subroutine `computePerfMatch` that we describe in the following.

- The input to `computePerfMatch` is an array A as above, together with an external assignment $x \in \{0,1\}^D$, given as a zero-one vector of length d. When called, the function returns

$$\sum_{y \in \{0,1\}^{[m] \setminus [d]}} \texttt{weight}(xy) \cdot \texttt{degOne}(xy),$$

  where for $z \in \{0,1\}^{[m]}$, we have

$$\begin{aligned} \texttt{weight}(z) &= \prod_{i \in z} w(e_i), \\ \texttt{degOne}(z) &= [\forall 1 \leq i \leq n : \mathrm{hw}(z|_{I(i)}) = 1]. \end{aligned}$$

  This is clearly equal to $\mathrm{Sig}(\Gamma, x)$.

The code we wrote to implement these functions is listed in the following.

```
1  function computeSignature(A,n,m,d)
2     for i = 1:2^d
3
4        % compute binary representation of i-1 as vector of length d
5        x = zeros(1,d);
6        actString = dec2bin(i-1,d);
7        for t = 1:d
8           x(t) = (actString(t) == '1');
9        end
10
11       % compute and display Sig(A,x) if it is nonzero
12       signature_value = computePerfMatch(A,n,m,d,x);
13       if signature_value ~= 0
14          disp(['Sig(' actString ')␣=␣' num2str(signature_value)]);
15       end
16    end
17 end
```

```
1  function PerfMatch = computePerfMatch(A,n,m,d,x)
2
3     % initialize
4     PerfMatch = 0;
5
6     for i = 1:2^(m-d)
7
8        % compute binary representation of i-1 as vector of length m-d
```

```
9         y = zeros(1,m-d);
10        actString = dec2bin(i-1,m-d);
11        for t = 1:m-d
12           y(t) = (actString(t) == '1');
13        end
14
15        % concatenate x and y
16        z = [x y];
17
18        % compute weight(z)
19        weight = prod(A(z == 1,3));
20
21        % compute degOne, initialized to 1
22        degOne = 1;
23        for v = 1:n
24
25           % compute Hamming weight of edges incident with v
26           curHammWeight = 0;
27           for e = 1:m
28              actEdge = A(e,:);
29              if z(e) == 1 && (actEdge(1) == v || actEdge(2) == v)
30                 curHammWeight = curHammWeight + 1;
31              end
32           end
33
34           % first vertex with wrong curHammWeight sets degOne to 0
35           if curHammWeight ~= 1
36              degOne = 0;
37           end
38        end
39
40        % add the computed term to PerfMatch
41        PerfMatch = PerfMatch + weight * degOne;
42     end
43  end
```

C.1. Equality matchgate

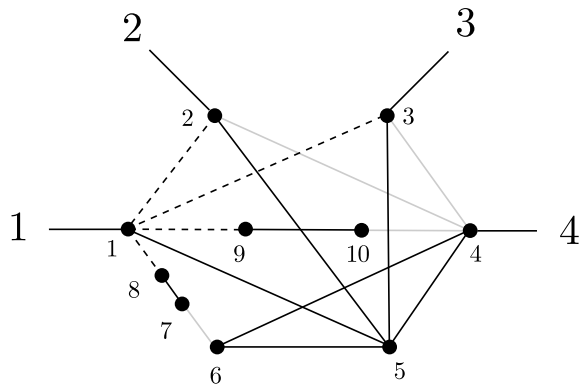


Figure C.1.: The equality matchgate $\Gamma_=$ with numbered vertices

```
>> A = [
0 1 1;
0 2 1;
0 3 1;
0 4 1;
5 6 1;
4 6 1;
3 5 1;
4 5 1;
2 5 1;
1 5 1;
3 4 -1;
2 4 -1;
1 2 0.5;
1 3 0.5;
1 9 0.5;
1 8 0.5;
6 7 -1;
4 10 -1;
9 10 1;
7 8 1
];

n = 10;
m = 20;
d = 4;

>> computeSignature(A,n,m,d);
Sig(0000) = 1
Sig(1111) = 1
```

C.2. Matchgate verifications for Section 4.1

The matchgate $\Gamma_{\texttt{PASS}}$ realizes PASS

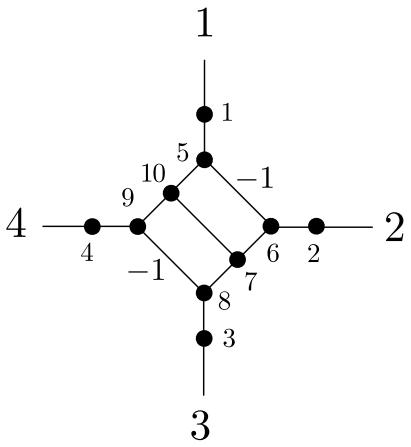


Figure C.2.: The matchgate $\Gamma_{\texttt{PASS}}$ with numbered vertices

```
>> A = [
0 1 1;
0 2 1;
0 3 1;
0 4 1;
1 5 1;
2 6 1;
6 7 1;
7 8 1;
3 8 1;
4 9 1;
9 10 1;
5 10 1;
7 10 1;
5 6 -1;
8 9 -1;
];

n = 10;
m = 15;
d = 4;

>> computeSignature(A,n,m,d);
Sig(0000) = 1
Sig(0101) = 1
Sig(1010) = 1
Sig(1111) = -1
```

The signature of the matchgate Ψ

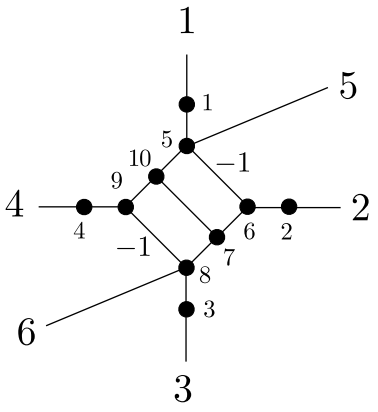


Figure C.3.: The matchgate Ψ with numbered vertices

```
>> A = [
0 1 1;
0 2 1;
0 3 1;
0 4 1;
0 5 1;
0 8 1;
1 5 1;
2 6 1;
6 7 1;
7 8 1;
3 8 1;
4 9 1;
9 10 1;
5 10 1;
7 10 1;
5 6 -1;
8 9 -1; ];

n = 10; m = 17; d = 6;

% the bitstring xy (with x of length 4) is relevant if hw(x) is even

>> computeSignature(A,n,m,d);
Sig(000000) = 1    % relevant
Sig(001001) = 1
Sig(010100) = 1    % relevant
Sig(011101) = 1
Sig(100010) = 1
Sig(101000) = 1    % relevant
Sig(101011) = 1    % relevant
Sig(110110) = 1
Sig(111100) = -1   % relevant
Sig(111111) = 1    % relevant
```