THE SIMPLE, LITTLE AND SLOW THINGS COUNT: ON PARAMETERIZED COUNTING COMPLEXITY

Radu Curticapean*

Abstract

This contribution is intended to be a self-contained and minimally technical exposition of the material in my 2015 dissertation, which was supervised by Markus Bläser. As its title suggests, the thesis investigates the complexity of combinatorial counting problems in the frameworks of parameterized (and exponential-time) complexity. More precisely, the following specific settings are explored:

- Counting perfect matchings in structurally "simple" graphs, for instance, in graphs that exclude specific fixed minors
- Counting small subgraph patterns in large host graphs
- Exponential lower bounds on the running time needed to solve counting problems, assuming popular conjectures such as the exponential-time hypothesis

1 Introduction

Many problems in theoretical computer science ask about the *existence* of solutions to a given instance of a problem. This includes, most prominently, the problems in NP, such as the NP-complete Boolean satisfiability problem SAT. However, in both practical and theoretical applications, it may be equally important to *find* a solution, to *list* all solutions, or to *count* the solutions for a given input–it is this last problem that we study in the dissertation. For instance, we can extend SAT to a counting problem #SAT, which asks, given as input a Boolean formula φ, to determine the number of assignments satisfying φ. This is obviously more difficult than merely deciding satisfiability of φ, and by Toda's theorem [41],

*Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI). Supported by ERC Starting Grant PARAMTIGHT (No. 280152). The work on this thesis was done under the supervision of Markus Bläser while the author was a student at Saarland University.

an oracle for #SAT even gives us polynomial-time algorithms for problems we assume to lie outside of NP, namely, the entire polynomial-time hierarchy.

However, counting problems also occur in scientific disciplines other than computational complexity theory: For instance, in statistical physics, various thermodynamic properties of systems can be expressed in terms of their *partition functions*, which are essentially weighted sums over the system's state space [32]. Since such spaces are typically of exponential size, brute-force summations over all states are prohibited, so more efficient algorithms are required (and can sometimes even be obtained [40, 31]). Furthermore, counting also occurs in probabilistic inference [8], since asking for the number of satisfying assigments to a formula φ is equivalent to asking for the probability that a random assignment satisfies φ. In the areas of bioinformatics and network analysis, counting problems occur when one wants to prove that a specific pattern occurs with significant frequency in a given network [27, 42].

The complexity of counting problems

For the vast majority of interesting counting problems, efficient algorithms are not known, creating a need for a complexity theory of counting problems. This was provided by Valiant in his classical paper [44], where he introduced the complexity class #P that captures the counting versions of problems in NP. For instance, #SAT is complete for #P under polynomial-time many-one reductions. More interestingly however, Valiant identified natural #P-complete counting problems whose corresponding decision version can be solved in polynomial time. For instance, his paper contains a seminal proof that counting perfect matchings in a bipartite graph is #P-complete (under polynomial-time Turing reductions), even though the existence of a perfect matching in a given graph can be tested in polynomial time [23].

Since this initial result, the complexity-theoretic study of counting problems advanced to a classical sub-area of complexity theory, and the specific problem of counting perfect matchings played an important role throughout this development. We will abbreviate this problem by #PerfMatch in the following, and we also consider a generalization to edge-weighted graphs: Here, the task is, given a graph with edge-weights $w : E(G) \to \mathbb{Q}$, to compute the value

$$\#\mathrm{PerfMatch}(G) = \sum_{M} \prod_{e \in M} w(e), \tag{1}$$

where M ranges over all perfect matchings of G [45]. This clearly generalizes the problem #PerfMatch by setting all edge-weights to 1.

For instance, it was actually already shown before Valiant's hardness result that #PerfMatch (even with edge-weights) can be solved in polynomial time on

planar input graphs [40, 31, 32]. Building upon this, Valiant recently introduced the notion of holographic algorithms [45], which allow us to reduce a variety of other counting problems to this specific positive case. Furthermore, in combinatorics and algebraic complexity theory, the number of perfect matchings in a bipartite graph with $n + n$ vertices and bi-adjacency matrix A is known as the *permanent* of A, which is defined as

$$\operatorname{perm}(A) = \sum_{\pi} \prod_{i=1}^{n} A(i, \pi(i)),$$

where π ranges over all permutations of $1, \ldots, n$. The permanent is central to algebraic complexity theory, where an algebraic variant of the "P vs. NP" question asks to distinguish the complexity of the permanent from that of the deceivingly similar looking determinant [1].

Classical strategies for coping with hardness

As it turned out that #PerfMatch and many other interesting counting problems are #P-complete, relaxed versions were introduced to cope with their computational hardness. Classical examples for such relaxations include:

- Restricted input classes: As mentioned, #PerfMatch is polynomial-time solvable on planar graphs [40, 31, 32]. However, it remains #P-complete on 3-regular graphs [20]. The related problem of counting all (not necessarily perfect) matchings is #P-complete on planar 3-regular graphs [43].

- Approximate counting: On bipartite graphs, the problem #PerfMatch admits a fully polynomial randomized approximation scheme [30]. That is, given an n-vertex bipartite graph and numbers ϵ, δ as input, we can output a multiplicative $(1 \pm \epsilon)$ approximation to #PerfMatch(G) with probability at least $1 - \delta$ in time polynomial in $n, \epsilon^{-1}, \delta^{-1}$.

- Counting modulo fixed numbers: The parity of #PerfMatch(G) is easily computed if G is a bipartite graph with bi-adjacency matrix A. Namely, since the permanent and the determinant of any matrix coincide modulo 2, the parity of #PerfMatch(G) is that of $\det(A)$. This argument can be generalized to counting modulo 2^t for fixed $t \in \mathbb{N}$ [44]. On the other hand, #PerfMatch(G) modulo q is NP-hard to compute (under randomized reductions) when q is not a power of two [44].

In my dissertation, we study two of the most recent relaxations of counting problems, namely, their parameterized complexity (introduced by Flum and Grohe

[25]) and their exponential-time complexity (introduced by Dell et al. [21]).[1]

1.1 Parameterized counting complexity

Parameterized counting complexity is dedicated to the study of *parameterized counting problems*. These are pairs $(\#\mathsf{A}, \kappa)$, where $\#\mathsf{A}$ is a counting problem with inputs $\{0,1\}^*$, together with a *parameterization* $\kappa : \{0,1\}^* \to \mathbb{N}$ such that, for "typical" instances x, the parameter value $\kappa(x)$ is much smaller than the input length $|x|$. The concrete choice of a parameterization depends on the application, but for graph problems, parameterizations can broadly be classified as follows:

- *Structural parameters* are intended to measure some notion of complexity in the input graph, and a small parameter value means that the input enjoys a simple structure that could be used algorithmically. Exemplary parameters for a graph G include its maximum degree $\Delta(G)$, its crossing number $\mathrm{cr}(G)$, or its genus $\gamma(G)$.[2]

- For some counting problems, the input itself already contains a number $k \in \mathbb{N}$ such that structures of size k are to be counted; in such cases, it can make sense to *parameterize by the solution size* k. For instance, given a graph G and $k \in \mathbb{N}$, we can ask for the number of matchings in G with exactly k edges, or the number of vertex covers with precisely k vertices. When we parameterize these problems by k, this means intuitively that we are interested in solutions that are much smaller than G. Such a perspective makes sense, e.g., when small patterns are counted in huge databases.

If a suitable parameter κ was identified for a given $\#\mathsf{P}$-hard problem $\#\mathsf{A}$, we now ask whether it can be used algorithmically. The answer to this question may take one of the following forms [24]:

1. In the worst case, $\#\mathsf{A}$ is already $\#\mathsf{P}$-complete for constant values of $\kappa(x)$. For instance, we mentioned earlier that #PerfMatch is $\#\mathsf{P}$-complete, even for input graphs G of maximum degree $\Delta(G) \leq 3$.

2. The situation is more favorable if, for every $k \in \mathbb{N}$, the problem $\#\mathsf{A}$ can be solved in polynomial time on inputs x with $\kappa(x) \leq k$, or even better, if we can find an algorithm for $\#\mathsf{A}$ that runs in time $O(|x|^{f(\kappa(x))})$, where f is

[1] For *decision* problems, parameterized complexity was already introduced by Downey and Fellows [22], and exponential-time complexity was introduced by Impagliazzo et al. [29, 28].

[2] Here, the crossing number of G is the minimum number of edge-crossings over all drawings of G in the plane. The genus of G is the minimum genus of a surface on which G can be drawn without crossings.

some computable function. In this case, we speak of an **XP algorithm**. For example, such algorithms exist for counting matchings with k edges in graphs with n vertices, since brute-force solves this problem in time $n^{O(k)}$.

3. In the ideal setting, we even obtain an algorithm for #A that admits a constant $c \in \mathbb{N}$ such that for every fixed value of $\kappa(x)$, the running time of the algorithm is bounded by $O(|x|^c)$. Compared to the previous case, we can hence even remove $\kappa(x)$ from the exponent of $|x|$. This leads to the notion of **fixed-parameter tractability (FPT)**: A problem with parameterization κ is FPT if it can be solved in time $O(f(\kappa(x)) \cdot |x|^c)$, where c is a fixed constant and f is any computable function. For instance, vertex-covers of size k can be counted in time $O(2^k \cdot n)$ on n-vertex graphs [25], and #PerfMatch can be solved[3] in time $O(4^g \cdot n^3)$ on n-vertex graphs of genus g [26].

These definitions allow us to state some of the main goals of parameterized algorithms and complexity theory: Given a parameterized problem $(\#\mathsf{A}, \kappa)$, can we find an FPT algorithm (or at least an XP algorithm)? If not, can we explain our lack of progress as a consequence of widely-believed assumptions in complexity theory? For instance, to rule out XP algorithms for parameterized counting problems, it suffices to prove #P-hardness of the problem for a fixed parameter value. This approach however fails to rule out FPT algorithms for problems that already admit XP algorithms, since these are polynomial-time solvable for every fixed parameter value.

To explain the absence of FPT algorithms, a complexity class #W[1] of parameterized counting problems (analogously to #P) was introduced, along with a suitable hardness notion [25]. This class #W[1] can be defined as the set of all parameterized problems that reduce, by means of **parameterized reductions**, to counting k-cliques in a graph, parameterized by k. Here, a parameterized (Turing) reduction from a parameterized problem $(\#\mathsf{A}, \kappa)$ to another $(\#\mathsf{B}, \tau)$ is an algorithm that solves #A on inputs x in time $f(\kappa(x)) \cdot |x|^{O(1)}$ when given oracle access to #B. However, all oracle queries y to #B must satisfy $\tau(y) \leq g(\kappa(x))$. In the above statement, both f and g are arbitrary computable functions. We observe that the class of FPT problems is closed under parameterized reductions.

Using parameterized reductions from counting k-cliques, Flum and Grohe [25] showed that the problems of counting paths (or cycles) of length k are each #W[1]-complete. Hence, under the reasonable and widely-believed assumption that counting k-cliques admits no FPT algorithm, these problems do not admit FPT algorithms. This is particularly interesting, since we can find a path (or cycle) of length k in time $2^{O(k)} \cdot n^{O(1)}$ [2].

[3] We assume here that an embedding into the surface is given along with the input.

1.2 Exponential-time complexity for counting problems

Once a parameterized problem was classified as FPT or #W[1]-hard, we know whether to expect running times of type $f(k) \cdot n^{O(1)}$ or $n^{f(k)}$. However, an even more fine-grained analysis is often possible: Using the theory of exponential-time complexity [29, 21], one can often pinpoint the optimal asymptotic growth of f under reasonable complexity-theoretic assumptions.

In our applications, we use the exponential-time hypothesis ETH [29] and its counting version #ETH [21], which postulate, roughly speaking, that SAT (and #SAT) on formulas in 3-CNF with n variables cannot be solved in time $2^{o(n)}$. Our current understanding does not allow us to prove these hypotheses, as they clearly imply the separation of P and NP (or #P). Nevertheless, a falsification of ETH or #ETH would imply an unexpected breakthrough in the theory of satisfiability algorithms. And, regardless of our belief in ETH or #ETH, it cannot be denied that, for surprisingly many problems, these hypotheses imply lower bounds that match the best known algorithms [34].

For instance, consider a counting problem #A with parameterization κ, and assume there was a polynomial-time reduction from #SAT to #A that maps formulas φ with n variables to instances x of #A with parameter $\kappa(x) = O(n)$. Then #ETH rules out algorithms with running time $2^{o(\kappa(x))} \cdot |x|^{O(1)}$ for the problem. This allows us, e.g., to rule out algorithms with running time $2^{o(g)} \cdot n^{O(1)}$ for #PerfMatch on graphs of genus g [18], thus complementing the previously mentioned upper bound of $O(4^g \cdot n^3)$. We can also use #ETH to derive lower bounds for #W[1]-hard problems: For instance, it is known that #ETH rules out algorithms with running time $f(k) \cdot n^{o(k)}$ for the problem of counting k-cliques [9]. In particular, #ETH thus implies that FPT and #W[1] do not coincide.

1.3 Organization of the dissertation

The remainder of this contribution follows the outline of my dissertation [14], which is structured into one introductory part and three main parts, each of which corresponds to an adjective in the title of the thesis.

In the introductory part of my dissertation, we first collect some basics from complexity theory, graph theory, algebra and combinatorics. We also include an introduction to the Holant framework, which will provide a clean language for many of the subsequent arguments. Building upon this, the first main part analyzes the problem #PerfMatch under structural parameters. In the second main part, we consider the problem of counting small subgraph patterns in large host graphs. In the last part, we study conditional exponential lower bounds for counting problems.

2 The Holant framework

One of the goals of the dissertation was to develop general tools for studying the complexity of counting problems. To this end, we first extend the theory of so-called *Holant problems* [45, 5, 7, 6]. These problems are defined on graphs $G = (V, E)$ whose vertices $v \in V$ are labelled with *signatures* f_v: If $I(v)$ denotes the set of edges incident with v, then f_v is a function that maps assignments $\{0,1\}^{I(v)}$ to complex numbers. Given such a graph, the problem then lies in evaluting a particular quantity Holant(G), which is a sum over all Boolean assignments $x \in \{0,1\}^E$ to the edges of G. In this sum, each assignment x is weighted by the product of evaluations $f_v(x|_{I(v)})$ over all $v \in V$. Here, $x|_{I(v)}$ denotes the restriction of the assignment x to edges in $I(v)$. Thus, we obtain

$$\text{Holant}(G) = \sum_{x \in \{0,1\}^E} \prod_{v \in V} f_v(x|_{I(v)}).$$

As an example, we can express #PerfMatch on unweighted graphs as a Holant problem: Given a graph G, assign a signature $f_v : I(v) \to \{0,1\}$ to each vertex $v \in V$ that checks whether the assignment to $I(v)$ has exactly one edge of value 1. In this case, f_v is supposed to output 1, otherwise 0. We can easily see that the non-zero terms in Holant(G) then correspond precisely to the perfect matchings in G. Using more complex signatures, we can also express other problems as Holant problems, including the problem #SAT.

In the thesis, we introduce two tools for Holant problems, which are used throughout the main parts and also found applications in other projects [19, 18]. Firstly, we construct a uniform reduction from Holant problems with arbitrary signatures to #PerfMatch. To this end, we replace the vertices of the graph G in a Holant problem by gadgets that simulate their signatures. In particular, we can show that such gadgets exist for all signatures, unless they are ruled out for trivial reasons. The previous literature focused on the simulation of signatures by *planar* gadgets [45, 4]; for many signatures however, such gadgets do not exist. In joint work with Dániel [18], we later used this technique to show that #PerfMatch cannot be solved in time $2^{o(k)} \cdot n^{O(1)}$ on graphs of crossing number k or treewidth k, unless #ETH fails.

Secondly, in joint work with Mingji Xia [19], we introduce linear combinations of signatures as a technique for parameterized reductions between Holant problems: If a graph G in a Holant problem contains a small number k of "difficult" signatures that can be expressed as linear combinations of $c \in \mathbb{N}$ "simple" signatures, then we can express Holant(G) as a linear combination of c^k values Holant(G'), where each instance G' contains only simple signatures. The resulting running time of $c^k \cdot n$ is compatible with parameterized reductions. Apart from the applications in the thesis, we used this technique to show that #PerfMatch modulo

2^k is $W[1]$-hard to compute (under randomized reductions) [19], complementing the known $n^{O(k)}$ time algorithm [44].

3 Counting perfect matchings in simple graphs

In the first main part of the dissertation, we study the problem of computing #PerfMatch(G) on graphs G that exclude fixed minors H. This can be considered as a generalization of the polynomial-time solvable case of planar graphs [40, 31, 32], since any planar graph excludes both $K_{3,3}$ and K_5 as minors.

Graph minors play a fundamental role in graph theory, where they led to the celebrated Graph Minor Theorem [38]: Every graph class C that is closed under taking minors can be expressed by a finite set $F(C)$ of forbidden minors. That is, any given graph G is contained in C if and only if G contains none of the graphs in $F(C)$ as a minor. This holds exemplarily for the class C of planar graphs, where $F(C)$ takes the form of $\{K_{3,3}, K_5\}$.

Graph minors are quite relevant to the problem of counting perfect matchings, since almost all known polynomial-time algorithms for #PerfMatch on restricted graph classes in fact apply to classes that exclude at least one fixed minor. This holds for the class of planar graphs, and for subsequent algorithms on $K_{3,3}$-free graphs [33], graphs of bounded genus [26], K_5-free graphs [39], and graphs of bounded treewidth [46].[4] In an attempt to connect these individual dots, we asked ourselves how #PerfMatch behaves on graph classes excluding an arbitrary fixed minor H, when parameterized by the size of H.

To answer this question, we use the Graph Structure Theorem [37], which describes the structure of graphs excluding fixed minors H. It asserts that, for any fixed graph H, there is a constant $k = k(H)$ such that the H-free graphs can be obtained as *clique-sums* from graphs that are *k-almost-embeddable* in a surface of genus k. Here, a clique-sum of graphs A and B is executed by choosing equal-sized cliques in A and B, then taking the disjoint union of A and B while identifying the chosen cliques, and finally deleting an arbitrary set of edges from the resulting clique in the union. Furthermore, a graph F is k-almost-embeddable in a surface S if we can delete k so-called apex vertices from F such that the resulting graph can be embedded in S without crossings, with the exception of k vortices, namely, k faces into which certain "thin" non-planar graphs may be embedded.

[4] An exceptional tractable class is, e.g., the class $\mathcal{K}$ of complete graphs. This class clearly excludes no fixed minor, but the number of perfect matchings in complete graphs can be computed by a closed formula. This fact is actually subsumed by a more general result that #PerfMatch can be computed in polynomial time on graph classes of bounded clique-width [35]. We consider this result as an exception, since it does not apply to the edge-weighted case of #PerfMatch, whereas the other algorithms in our list do.

As mentioned before, #PerfMatch is FPT on graphs that can be embedded on a surface of genus k [26]. To understand the case of #PerfMatch on general H-minor free grpahs, we hence need to understand the influence of apex vertices, vortices and clique-sums on the complexity of this problem. In joint work with Mingji Xia, we show that #PerfMatch already becomes #W[1]-hard on planar graphs with k apex vertices. This is the first application of the technique of linear combinations of signatures mentioned in Section 2.

Theorem 1 ([19])**.** *The following problem is* #W[1]*-hard: Given a graph G and a vertex set $A \subseteq V(G)$ such that $G - A$ is planar, determine* #PerfMatch(G)*, parameterized by $|A|$. Furthermore, this problem admits no $n^{o(k/\log k)}$ time algorithm unless* #ETH *fails.*

Theorem 1 implies that #PerfMatch is #W[1]-hard on graphs excluding a fixed minor H, when parameterized by the size of H, since planar graphs with a fixed number of apex vertices exclude fixed minors. By a further non-trivial reduction, we can strengthen Theorem 1 to obtain the #W[1]-hardness of the related problem of counting matchings with exactly k unmatched vertices in planar graphs [15].

Despite these negative results, we can obtain FPT algorithms for restricted classes of excluded minors. In particular, we identify one such class in the thesis, namely, the class of minors that can be drawn in the plane with at most one crossing. This class includes the graphs $K_{3,3}$ and K_5, and hence, this result generalizes some of the algorithms mentioned in the beginning of this section.

Theorem 2 ([12])**.** *If H is a graph that can be drawn in the plane with at most one crossing, then the problem* #PerfMatch *can be solved in time $O(f(H) \cdot n^4)$ on graphs that exclude H as a minor. Here, f is a computable function.*

The dissertation does not answer the question whether #PerfMatch can actually be solved in time $n^{f(H)}$ on graphs excluding arbitrary fixed minors H. Recent unpublished work by the author however suggests a negative answer.

4 Counting small subgraphs

In the second main part of the thesis, we count small subgraph patterns H on k vertices, such as paths or cycles of size k, in general host graphs G with n vertices. That is, given graphs H and G, we wish to count all subgraphs of G that are isomorphic to H, parameterized by k. This has vast applications in network analysis, see [36].

A simple brute-force approach guarantees a running time of $n^{O(k)}$ for this problem, which may however already be prohibitive for small values of k. Furthermore, as mentioned in Section 1.2, we do not expect $n^{o(k)}$ time algorithms for

k-vertex subgraphs such as cliques under #ETH. We are hence interested in identifying additional properties on H that can be exploited to render the subgraph counting problem FPT. To this end, we introduce the following problem #Sub($\mathcal{H}$) for fixed recursively enumerable graph classes $\mathcal{H}$: Given graphs G and $H \in \mathcal{H}$, count subgraphs of G that are isomorphic to H, parameterized by $|V(H)|$. Our goal is to determine, for each class $\mathcal{H}$, whether #Sub($\mathcal{H}$) is FPT or #W[1]-hard.

For instance, if $\mathcal{H}$ is the class of cycles or the class of paths, then the #W[1]-completeness of #Sub($\mathcal{H}$) was already proven by Flum and Grohe [25], despite the decision versions of these problems being FPT. The same authors also conjectured that #Sub($\mathcal{M}$) is #W[1]-complete for the class $\mathcal{M}$ of matchings. This corresponds to counting matchings with k edges in graphs, and can thus be considered as a parameterized version of #PerfMatch.

In joint work with Markus Bläser [3], we first showed that a weighted version of this problem is #W[1]-hard. The weights in this version are defined analogously as in #PerfMatch. Building upon this, the #W[1]-completeness of the unweighted version was shown later [11]. This first proof was however mostly superseded by a simplified proof that resulted from joint work with Dániel Marx [17], and which exploits linear combinations of signatures. We additionally obtain an almost-tight lower bound for this problem under #ETH:

Theorem 3 ([17])**.** *The problem* #Sub($\mathcal{M}$) *ist* #W[1]*-complete: Given a graph G and $k \in \mathbb{N}$, it is thus* #W[1]*-complete to count matchings with k edges in G. Furthermore, an $n^{o(k/\log k)}$ time algorithm for this problem would refute* #ETH.

This constitutes a useful reduction source to prove #W[1]-hardness of other counting problems. In particular, it allows us to classify the problems #Sub($\mathcal{H}$), since #Sub($\mathcal{M}$) represents the minimal hard case in this setting: It is known that #Sub($\mathcal{H}$) can even be solved in time $n^{O(1)}$ if the graphs in $\mathcal{H}$ only contain matchings of constant size [47]. This applies, e.g., to the class $\mathcal{S}$ of stars, or generally to classes with constant-sized vertex covers. On the other hand, if $\mathcal{H}$ is a graph class whose graphs contain arbitrarily large matchings, then we show in joint work with Dániel Marx that #Sub($\mathcal{M}$) can be reduced to #Sub($\mathcal{H}$). This yields:

Theorem 4 ([17])**.** *Let $\mathcal{H}$ be a recursively enumerable class of graphs. If there is a constant $c \in \mathbb{N}$ such that no graph in $\mathcal{H}$ contains a matching with c edges, then* #Sub($\mathcal{H}$) *can be solved in time $O(n^d)$, where d depends on c. On the other hand, if $\mathcal{H}$ contains arbitrarily large matchings, then* #Sub($\mathcal{H}$) *is* #W[1]*-complete.*

It should be mentioned that this theorem actually shows for each class $\mathcal{H}$ whether #Sub($\mathcal{H}$) is polynomial-time solvable or #W[1]-complete. Assuming that FPT and #W[1] do not coincide, we thus obtain an exact classification of the

problems #Sub($\mathcal{H}$) that can be solved in polynomial time. Indeed, such a classification would not be possible by merely assuming that P and #P do not coincide, since there exist #P-intermediate cases for #Sub($\mathcal{H}$) [10].

5 Conditional lower bounds for counting problems

In the last part of the thesis, we investigate whether classical counting problems can be solved in sub-exponential time, i.e., in time $2^{o(n)}$ on graphs with n vertices. Throughout this part, we assume the exponential-time hypothesis #ETH from Section 1.2. Furthermore, in this setting as well, the problem #PerfMatch does not fail to be a useful reduction source for other hardness results, so we focus on lower bounds for this particular problem.

Building upon the work from [21], proving tight lower bounds for #PerfMatch essentially boils down to finding a "resourceful" way of simulating the edge-weight -1 in the weighted version of #PerfMatch. That is, we define a problem #PerfMatch$^{-1,0,1}$ of counting weighted perfect matchings as in (1), but only on graphs with edge-weights -1 and 1. To avoid confusion, we will henceforth denote #PerfMatch on unweighted graphs by #PerfMatch^{0,1}. It was shown [21] that an algorithm with running time $2^{o(n)}$ for #PerfMatch$^{-1,0,1}$ on graphs with n vertices and $O(n)$ edges would refute #ETH.

To proceed, it is essential to remove the edge-weight -1 from #PerfMatch$^{-1,0,1}$, as it would otherwise be very unclear how to handle this negative weight in reductions to other target problems. A classical solution for such weight removal is based on polynomial interpolation [43]: Given an n-vertex graph G with edge-weights -1 and 1, we can define a graph G_x by replacing each occurrence of -1 with the indeterminate x. The quantity $p := \#\text{PerfMatch}(G_x)$ is then a polynomial of degree at most $\frac{n}{2}$ in x, and we have $p(-1) = \#\text{PerfMatch}(G)$ by definition. Hence, if we know the values $p(1), \ldots, p(\frac{n}{2} + 1)$, we can use polynomial interpolation to obtain the coefficients of p and thus evaluate $p(-1)$.

For positive integer values of t however, the evaluation of $p(t)$ can be reduced to #PerfMatch^{0,1}: An edge $uv \in E(G)$ of weight $t \in \mathbb{N}$ in a graph G is easily simulated by placing t parallel edges of weight 1 between u and v, then subdividing each of these edges twice. A graph with n vertices, m edges, and edge-weights from $\{1, \ldots, b\}$ is thus transformed to an unweighted graph on $O(n + bm)$ vertices and edges. Using this with the interpolation argument above, we obtain a polynomial-time Turing reduction from #PerfMatch$^{-1,0,1}$ to #PerfMatch^{0,1} which transforms n-vertex graphs G with edge-weights ± 1 to unweighted simple graphs with $O(n^2)$ vertices. This quadratic blowup however prevents us from proving a tight lower bound under #ETH: To obtain an algorithm with running time $2^{o(n)}$ for #PerfMatch$^{-1,0,1}$, and thus refute #ETH, we would need to solve #PerfMatch^{0,1} in

time $2^{o(\sqrt{n})}$. One could try to find more resourceful ways of simulating positive weights $t \in \mathbb{N}$, and indeed gadgets on $O(\log t)$ vertices exist [21], thus ruling out $2^{o(n/\log n)}$ time algorithms for #PerfMatch^{0,1}. This is however still not tight, and we can see that any gadgets simulating edges of weight $t \in \mathbb{N}$ *must* have size $\omega(1)$, thus apparently rendering interpolation futile for proving tight lower bounds.

In the dissertation, we solve this problem in two ways: Firstly, we introduce "block interpolation", a general approach to make polynomial interpolation compatible with tight lower bounds under #ETH. This technique relies on the insight that interpolation arguments as above, which are built around polynomials in a single indeterminate x, can often be carried out on polynomials in indeterminates $x_1, \ldots, x_t$, each of which occurs with some maximum degree d. Such polynomials can be interpolated from their evaluations on a grid $\{1, \ldots, d+1\}^t$, which requires only $d+1$ distinct values to be substituted into each individual indeterminate. By trading off t with d, we obtain reductions that run in sub-exponential time, but only create linear-sized reduction images. This general approach allows us to prove lower bounds for the following problems:

Theorem 5 ([13])**.** *Unless* #ETH *fails, the problems of counting matchings, perfect matchings, independent sets, and vertex covers cannot be solved in time* $2^{o(n)}$ *on n-vertex graphs.*

Our second solution for weight removal applies only to the specific problem #PerfMatch, but has other benefits: For any graph G with edge-weights ± 1, we show how to construct unweighted graphs G_1 and G_2 such that #PerfMatch(G) is the difference of #PerfMatch(G_1) and #PerfMatch(G_2).

Theorem 6 ([16])**.** *If G is a graph with n vertices, m edges, and edge-weights -1 and 1, then we can construct unweighted graphs G_1 and G_2 on $O(n+m)$ vertices and edges such that* #PerfMatch(G) = #PerfMatch(G_1) − #PerfMatch(G_2).

This immediately transfers the known lower bound for #PerfMatch$^{-1,0,1}$ to one for #PerfMatch^{0,1}. However, it also gives us insights into the structural complexity of counting perfect matchings, as it allows us to prove that the following "equality testing" version #PerfMatch$_=$ of #PerfMatch is complete for the class $\mathsf{C}_=\mathsf{P}$: Given two unweighted graphs, this problem asks to decide whether they have the same number of perfect matchings. Here, $\mathsf{C}_=\mathsf{P}$ can be defined as the class of problems that are polynomial-time many-one reducible to deciding whether two Boolean formulas φ_1, φ_2 have the same number of satisfying assignments. Furthermore, bridging quantitative lower bounds and structural complexity, our proof shows that ETH rules out a $2^{o(n)}$ time algorithm for #PerfMatch$_=$.

Acknowledgments

I want to thank my advisor Markus Bläser for all the guidance and support he gave me during my PhD time and afterwards, and my other co-authors Mingji Xia and Dániel Marx for great collaborations.

References

[1] Manindra Agrawal. Determinant versus permanent. In *Proceedings of the 25th International Congress of Mathematicians, ICM 2006*, volume 3, pages 985–997, 2006.

[2] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[3] Markus Bläser and Radu Curticapean. Weighted counting of k-matchings is #W[1]-hard. In *IPEC 2012*, pages 171–181, 2012.

[4] Jin-Yi Cai and Aaron Gorenstein. Matchgates revisited. *Theory of Computing*, 10:167–197, 2014.

[5] Jin-yi Cai, Pinyan Lu, and Mingji Xia. Holant problems and counting CSP. In *STOC 2009*, pages 715–724, 2009.

[6] Jin-yi Cai, Pinyan Lu, and Mingji Xia. Computational complexity of Holant problems. *SIAM J. Comput.*, 40(4):1101–1132, 2011.

[7] Jin-yi Cai, Pinyan Lu, and Mingji Xia. Dichotomy for Holant* problems of Boolean domain. In *SODA 2011*, pages 1714–1728, 2011.

[8] Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artif. Intell.*, 172(6-7):772–799, 2008.

[9] 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.

[10] Yijia Chen, Marc Thurley, and Mark Weyer. Understanding the complexity of induced subgraph isomorphisms. In *ICALP (1)*, pages 587–596, 2008.

[11] Radu Curticapean. Counting matchings of size k is #W[1]-hard. In *ICALP 2013*, pages 352–363, 2013.

[12] Radu Curticapean. Counting perfect matchings in graphs that exclude a single-crossing minor. *CoRR*, abs/1406.4056, 2014.

[13] Radu Curticapean. Block interpolation: A framework for tight exponential-time counting complexity. In *ICALP 2015*, pages 380–392, 2015.

[14] Radu Curticapean. *The simple, little and slow things count: on parameterized counting complexity*. PhD thesis, Universität des Saarlandes, 2015.

[15] Radu Curticapean. Counting matchings with k unmatched vertices in planar graphs. In *ESA 2016*, pages 33:1–33:17, 2016.

[16] Radu Curticapean. Parity separation: A scientifically proven method for permanent weight loss. In *ICALP 2016*, 2016.

[17] Radu Curticapean and Dániel Marx. Complexity of counting subgraphs: Only the boundedness of the vertex-cover number counts. In *FOCS 2014*, pages 130–139, 2014.

[18] Radu Curticapean and Dániel Marx. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. In *SODA 2016*, pages 1650–1669, 2016.

[19] Radu Curticapean and Mingji Xia. Parameterizing the permanent: Genus, apices, minors, evaluation mod 2^k. In *FOCS 2015*, pages 994–1009, 2015.

[20] Paul Dagum and Michael Luby. Approximating the permanent of graphs with large factors. *Theor. Comput. Sci.*, 102(2):283–305, 1992.

[21] 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.

[22] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

[23] Jack Edmonds. Paths, trees, and flowers. In *Classic Papers in Combinatorics*, Modern Birkhï£¡user Classics, pages 361–379. Birkhï£¡user Boston, 1987.

[24] J. Flum and M. Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[25] Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. *SIAM Journal on Computing*, (4):892–922, 2004.

[26] Anna Galluccio and Martin Loebl. On the theory of Pfaffian orientations. I. Perfect matchings and permanents. *Electronic Journal of Combinatorics*, 6, 1998.

[27] Sylvain Guillemot and Florian Sikora. Finding and counting vertex-colored subtrees. In *MFCS 2010*, pages 405–416, 2010.

[28] Russel Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[29] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Computer and Sys. Sci.*, 63(4):512–530, 2001.

[30] 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.

[31] 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.

[32] Pieter W. Kasteleyn. Graph Theory and Crystal Physics. In *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, 1967.

[33] Charles Little. An extension of Kasteleyn's method of enumerating the 1-factors of planar graphs. In *Combinatorial Mathematics*, LNCS, pages 63–72. 1974.

[34] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the EATCS*, 84:41–71, 2011.

[35] Johann A. Makowsky, Udi Rotics, Ilya Averbouch, and Benny Godlin. Computing graph polynomials on graphs of bounded clique-width. In *WG 2006*, pages 191–204, 2006.

[36] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, October 2002.

[37] Neil Robertson and Paul D. Seymour. Graph minors. XVI. Excluding a non-planar graph. *J. Comb. Theory, Ser. B*, 89(1):43 – 76, 2003.

[38] Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *J. Comb. Theory, Ser. B*, 92(2):325–357, 2004.

[39] Simon Straub, Thomas Thierauf, and Fabian Wagner. Counting the number of perfect matchings in K_5-free graphs. In *CCC 2014*, pages 66–77, 2014.

[40] H. N. V. Temperley and Michael E. Fisher. Dimer problem in statistical mechanics - an exact result. *Philosophical Magazine*, 6(68):1478–6435, 1961.

[41] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[42] Johan Ugander, Lars Backstrom, and Jon M. Kleinberg. Subgraph frequencies: mapping the empirical and extremal geography of large graph collections. In *WWW 2013*, pages 1307–1318, 2013.

[43] Salil P. Vadhan. The complexity of counting in sparse, regular, and planar graphs. *SIAM J. Comput.*, 31(2):398–427, 2001.

[44] Leslie G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8(2):189–201, 1979.

[45] Leslie G. Valiant. Holographic algorithms. *SIAM J. Comput.*, 37(5):1565–1594, 2008.

[46] Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In *ESA 2009*, pages 566–577, 2009.

[47] Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013.