

Sparse Instances of Hard Problems

DISSERTATION

zur Erlangung des akademischen Grades

DOCTOR RERUM NATURALIUM
im Fach Informatik

eingereicht an der
Mathematisch–Naturwissenschaftlichen Fakultät II
Humboldt–Universität zu Berlin

von
M.Sc. Holger Dell

Gefördert durch die Deutsche Forschungsgemeinschaft im Rahmen des Graduiertenkollegs
'Methods for Discrete Structures' (GRK 1408)

Präsident der Humboldt–Universität zu Berlin:
Prof. Dr. Jan-Hendrik Olbertz

Dekan der Mathematisch–Naturwissenschaftlichen Fakultät II:
Prof. Dr. Elmar Kulke

Gutachter:

1. Prof. Dr. Martin Grohe
2. Prof. Dr. Johannes Köbler
3. Prof. Dr. Dieter van Melkebeek

Tag der mündlichen Prüfung: 15. Juli 2011

Abstract

In this thesis, we use and refine methods of computational complexity theory to analyze the complexity of sparse instances, such as graphs with few edges or formulas with few constraints of bounded width. Two natural questions arise in this context:

- Is there an efficient algorithm that reduces arbitrary instances of an NP-hard problem to equivalent, sparse instances?
- Is there an algorithm that solves sparse instances of an NP-hard problem significantly faster than general instances can be solved?

We formalize these questions for different problems and show that positive answers for these formalizations would lead to consequences in complexity theory that are considered unlikely.

The first question is modeled by the following two-player communication process to decide a language L : The first player holds the entire input x but is polynomially bounded; the second player is computationally unbounded but does not know any part of x ; their goal is to decide cooperatively whether x belongs to L at small cost, where the cost measure is the number of bits of communication from the first player to the second player.

For any integer $d \geq 3$ and positive real ϵ we show that if satisfiability for n -variable d -CNF formulas has a protocol of cost $O(n^{d-\epsilon})$ then coNP is in NP/poly, which implies that the polynomial-time hierarchy collapses to its third level. We obtain similar results for various NP-complete covering and packing problems in graphs and hypergraphs. The results even hold when the first player is conondeterministic, and are tight as there exists a trivial protocol for $\epsilon = 0$. Under the hypothesis that coNP is not in NP/poly, our results imply surprisingly tight lower bounds for parameters of interest in several areas, namely sparsification, kernelization in parameterized complexity, lossy compression, and probabilistically checkable proofs.

We study the second question from above for counting problems in the exponential time setting. The Exponential Time Hypothesis (ETH) is the complexity assumption that the satisfiability of n -variable 3-CNF formulas cannot be decided in time $\exp(o(n))$. Assuming (variants of) ETH, we obtain asymptotically tight, exponential lower bounds for well-studied #P-hard problems:

- Computing the number of satisfying assignments of a 2-CNF formula,
- Computing the number of all independent sets in a graph,
- Computing the permanent of a matrix with entries 0 and 1,
- Evaluating the Tutte polynomial of multigraphs at fixed evaluation points.

We also obtain results for the Tutte polynomial of simple graphs, where our lower bounds are asymptotically tight up to polylogarithmic factors in the exponent of the running time.

Zusammenfassung

Diese Arbeit nutzt und verfeinert Methoden der Komplexitätstheorie, um mit diesen die Komplexität dünner Instanzen zu untersuchen. Dazu gehören etwa Graphen mit wenigen Kanten oder Formeln mit wenigen Bedingungen beschränkter Weite. Dabei ergeben sich zwei natürliche Fragestellungen:

- Gibt es einen effizienten Algorithmus für NP-schwere Probleme, der beliebige Instanzen auf äquivalente, dünne Instanzen reduziert?
- Gibt es einen Algorithmus, der dünne Instanzen NP-schwerer Probleme bedeutend schneller löst als allgemeine Instanzen gelöst werden können?

Wir formalisieren diese Fragen für verschiedene Probleme und zeigen, dass positive Antworten jeweils zu komplexitätstheoretischen Konsequenzen führen, die als unwahrscheinlich gelten.

Die erste Frage wird als Kommunikation modelliert, in der zwei Akteure kooperativ eine Sprache L entscheiden möchten: Der erste Akteur kennt hierbei die gesamte Eingabe x , ist aber zeitlich polynomiell beschränkt. Die zweite Akteurin ist ein unbeschränktes Orakel, dem aber zunächst kein Teil der Eingabe bekannt ist. Gemeinsam möchten sie nun mit möglichst wenig Aufwand entscheiden, ob x ein Wort der Sprache L ist, wobei der Aufwand einer Kommunikation die Zahl der Bits ist, die der erste Spieler an das Orakel sendet.

Wir zeigen, dass für alle natürlichen Zahlen $d \geq 3$ und alle positiven reellen Zahlen ϵ gilt: Wenn die Spieler die Erfüllbarkeit von d -KNF Formeln auf n Variablen mit einem Kommunikationsaufwand von $O(n^{d-\epsilon})$ entscheiden können, dann ist coNP eine Teilmenge von NP/poly . Letzteres impliziert, dass die Polynomialzeithierarchie auf die dritte Stufe kollabiert. Analoge Ergebnisse erhalten wir für verschiedene NP-vollständige Überdeckungs- und Packungsprobleme in Graphen und Hypergraphen. Diese Ergebnisse gelten sogar dann, wenn der erste Spieler $\text{co-nichtdeterministisch}$ ist, und sind optimal, da es jeweils ein triviales Protokoll mit $\epsilon = 0$ gibt. Unter der Hypothese, dass coNP keine Teilmenge von NP/poly ist, erhalten wir als Korollare erstaunlich scharfe untere Schranken für interessante Parameter aus verschiedenen Teilgebieten der theoretischen Informatik. Im Speziellen betrifft das die Ausdünnung von Formeln, die Kernelisierung aus der parameterisierten Komplexitätstheorie, die verlustbehaftete Kompression von Entscheidungsproblemen, und die Theorie der probabilistisch verifizierbaren Beweise.

Wir untersuchen die zweite obige Fragestellung anhand der Exponentialzeitkomplexität von Zählproblemen. Die Exponentialzeithypothese (ETH) besagt, dass das Erfüllbarkeitsproblem für 3-KNF Formeln mit n Variablen nicht in Zeit $\exp(o(n))$ gelöst werden kann. Unter (Varianten) dieser Hypothese zeigen wir asymptotisch scharfe, exponentielle untere Schranken für wichtige $\#\text{P}$ -schwere Probleme:

- Berechnen der Zahl der erfüllenden Belegungen einer 2-KNF Formel,
- Berechnen der Zahl aller unabhängigen Mengen in einem Graphen,
- Berechnen der Permanente einer Matrix mit Einträgen 0 und 1,
- Auswerten des Tuttepolynoms von Multigraphen an festen Punkten.

Außerdem zeigen wir untere Schranken für die Auswertung des Tuttepolynoms von einfachen Graphen, die bis auf polylogarithmische Faktoren im Exponenten der Laufzeit asymptotisch optimal sind.

*If you can look into the seeds of time,
And say which grain will grow, and which will not,
Speak.*
— Banquo

Acknowledgement

I am deeply grateful to my advisor Martin Grohe for his ongoing support and guidance. He and his group provided me with an excellent and stimulating environment during my years as a PhD student.

Special thanks go to my coauthors Thore Husfeldt, Dániel Marx, Nina Taslaman, Dieter van Melkebeek, and Martin Wahlén.

In addition, I would like to thank the following people for discussions, comments, and references: Matt Anderson, Albert Atserias, Christoph Berkholz, Andreas Björklund, Markus Bläser, Yijia Chen, Kord Eickmeyer, Leslie Ann Goldberg, Berit Grußien, Johan Håstad, Danny Hermelin, Bastian Laubner, Daniel Lokshtanov, Moritz Müller, Sarah Rich, Saket Saurabh, Mathias Schacht, Asaf Shapira, Siamak Tazari, Luca Trevisan, Chris Umans, Magnus Wahlström, Thomas Watson, Dalibor Zelený.

Contents

1. Introduction	1
1.1. Hardness of Sparsification	1
1.2. Sparse Instances of Counting Problems	9
2. Preliminaries	17
2.1. Review: Kernelization of Vertex Cover	19
2.2. Review: Polynomial Kernel Lower Bounds	22
3. Communicating Instances of Hard Problems	25
3.1. ORs of NP-hard problems	25
3.2. Vertex Cover	28
3.3. Satisfiability	31
3.4. Covering Problems	33
3.5. Packing Problems	38
3.6. Other Applications	46
4. Packing Edge-disjoint Cliques	49
5. Exponential Time Counting Complexity	55
5.1. Counting Independent Sets	55
5.2. The Permanent	56
5.3. The Tutte Polynomial	58
5.3.1. Hyperbolas in the Tutte plane	59
5.3.2. Individual Points for Multigraphs	63
5.3.3. Individual Points for Simple Graphs	64
6. Summary and Open Problems	75
A. Behrend's Construction	77
B. Sunflower Kernelization for Set Matching	79
C. The Sparsification Lemma	81
D. Hardness of 3-Colouring and 3-Terminal MinCut	83

1. Introduction

The P vs. NP problem lies at the heart of computational complexity theory and is arguably the most important and beautiful mathematical question of our time. Is it as easy for an agent to recognize a solution to a problem as it is to come up with a solution? Based on our every-day experience, the answer to this question should be no: Building a bike is harder than verifying that it can transport you, acting is harder than verifying that the actors' interpretation of Shakespeare makes sense, gardening is harder than harvesting edible food and thereby verifying the gardeners' success, and, for that matter, the first ascent of the Eiger north face is harder than climbing a route known to work.

For such non-technical examples it seems obvious that finding a solution is much harder than recognizing it and undeniably there is a striking amount of empirical evidence in favor of that belief. On the other hand, if it *is* possible to train agents so that they can solve problems as easily as they are already able to verify solutions, and the training method just has not been found yet, then it is of inarguable importance to find out how to do it. The beauty of the P vs. NP problem lies in the fact that it can model this question in a mathematically rigorous way for many real-world problems – albeit not the ones mentioned above.

Satisfiability of Boolean formulas constitutes one of the most central problems in computer science. It has attracted a lot of applied and theoretical research because of its immediate relevance in areas like AI and verification, and as the seminal NP-complete problem. Of particular interest is d -SAT, the satisfiability problem for d -CNF formulas, which is NP-complete for any integer $d \geq 3$ [Coo71, Lev73, Kar72].

This thesis is about *sparse* instances of d -SAT and other NP-complete problems. In the first part of this thesis, we investigate the complexity of such problems in a communication setting that captures several transformations studied in the theory of computing. In particular, our results imply that it is hard to efficiently make instances of these hard problems sparse. In contrast, the second part of this thesis is about problems whose sparse instances are hard in the sense that they are unlikely to be solvable in subexponential time. These results rely on the exponential-time hypothesis, the hypothesis that d -SAT cannot be solved in subexponential time. For decision problems, this has been studied previously and we establish the first results for many natural counting problems.

1.1. Hardness of Sparsification

We investigate the oracle communication complexity of d -SAT and other natural NP-complete problems. Assuming the polynomial-time hierarchy does not collapse, we show that a trivial communication protocol is essentially optimal for d -SAT. Under the same

1. Introduction

hypothesis the result implies tight lower bounds for parameters of interest in several areas of theoretical computer science. We first discuss those areas and then state our result for d -SAT.

Sparsification. The satisfiability of d -CNF formulas chosen by uniformly at random picking m clauses out of all possible clauses on n variables seems to exhibit a phase transition as a function of the ratio m/n . We know that the probability of satisfiability jumps from almost zero to almost one when the ratio m/n crosses a very narrow region around $2^d \ln 2$, and the existence of a single threshold point is conjectured [FB99, AM07, AP04]. Experiments also suggest that known SAT solvers have the hardest time on randomly generated instances when the ratio m/n lies around the threshold, and in some cases rigorous analyses corroborate the experiments. Nevertheless, from a complexity-theoretic perspective these results fall short of establishing sparse formulas as the hardest instances. This is because formulas that express problems like breaking random RSA instances exhibit a lot of structure and therefore have a negligible contribution to the uniform distribution.

On the other hand, it can be shown that d -SAT remains NP-complete for “sparse” instances – a simple padding argument leads to a polynomial-time mapping reduction that maps arbitrary d -CNF formulas to equisatisfiable d -CNF formulas that have $O(n)$ clauses and n variables: simply add many unused variables until the bound is holds. The original, possibly highly dense formula is still contained as a subformula, and padding does not reduce its inherent density. While the padding argument can reduce the ratio m/n to a constant, reducing this ratio does not seem to capture our intuition of what it means to make a formula sparse.

An interesting complexity-theoretic formalization to avoid the two issues above would be a reduction from arbitrary formulas to sparse formulas *on the same number of variables*. Impagliazzo et al. [IPZ01] developed such reductions but they run in subexponential time. In polynomial time we can trivially reduce a d -CNF formula to one with $m = O(n^d)$ clauses. Since there are only $2^d \cdot \binom{n}{d} = O(n^d)$ distinct d -clauses on n variables, it suffices to remove duplicate clauses. Is there a polynomial-time reduction that maps a d -CNF formula on n variables to one on $O(n)$ variables and $m = O(n^{d-\epsilon})$ clauses for some positive constant ϵ ?

Kernelization. Parameterized complexity investigates the computational difficulty of problems as a function of the input size and an additional natural parameter, k , which often only takes small values in instances of practical interest. A good example – and one we will return to soon – is deciding whether a given graph has a vertex cover of size at most k . The holy grail in parameterized complexity are algorithms with running times of the form $f(k) \cdot s^c$ on instances of size s and parameter k , where f denotes an arbitrary computable function and c a constant. Kernelization constitutes an important technique for realizing such running times: Reduce in time polynomial in s to an instance of size bounded by some computable function g of the parameter k only, and then run a brute-force algorithm on the reduced instance; the resulting algorithm has a running

time of the form $O(s^c + f(k))$. In order to obtain good parameterized algorithms the functions f and g should not behave too badly, which justifies the quest for kernels of polynomial or smaller size $g(k)$.

The number of variables n forms a natural parameter for satisfiability. In the case of d -CNF formulas, n is effectively polynomially related to the size of the input, which makes the existence of kernels of polynomial size trivial. The quest for a small kernel is a relaxation of the quest for sparsification in polynomial time. Eliminating duplicate clauses yields a kernel of bitlength $O(n^d \log n)$. Does satisfiability of n -variable d -CNF formulas have kernels of size $O(n^{d-\epsilon})$?

Lossy Compression. Harnik and Naor [HN06] introduced a notion of compression with the goal of succinctly storing instances of computational problems for resolution in the future, where there may be more time and more computational power available. The compressed version need not be an instance of the original problem, and the original instance need not be recoverable from the compressed version. The only requirement is that the solution be preserved. In the case of decision problems this simply means the yes/no answer. In analogy to image compression one can think of the Harnik-Naor notion of compression as a “lossy compression”, where the only aspect of the scenery that is guaranteed not to be lost is the solution to the problem.

Harnik and Naor applied their notion to languages in NP and showed the relevance to problems in cryptography when the compression is measured as a function of the bitlength of the underlying witnesses. In the case of satisfiability the latter coincides with the number of variables of the formula. This way lossy compression becomes a relaxation of the notion of kernelization – we now want a polynomial-time mapping reduction to *any* problem, rather than to the original problem, such that the reduced instances have small bitlength as a function of n . For d -CNF formulas bitlength $O(n^d)$ is trivially achievable – simply map to the characteristic vector that for each possible d -clause on n variables indicates whether it is present in the given formula. Can we lossily compress to instances of bitlength $O(n^{d-\epsilon})$?

Probabilistically Checkable Proofs. A somewhat different question deals with the size of probabilistically checkable proofs (PCPs). A PCP for a language L is a randomized proof system in which the verifier only needs to read a constant number of bits of the proof in order to verify that a given input x belongs to L . Completeness requires that for every input x in L there exists a proof which the verifier accepts with probability one. Soundness requires that for any input x outside of L no proof can be accepted with probability above some constant threshold less than one. For satisfiability of Boolean formulas, Dinur [Din07] constructed PCPs of bitlength $O(s \cdot \text{poly log } s)$, where s denotes the size of the formula. For d -CNF formulas on n variables, Dinur’s construction yields PCPs of bitlength $O(n^d \cdot \text{poly log } n)$. On the other hand, standard proofs only contain n bits. Do n -variable d -CNF formulas have PCPs of bitlength $O(n^{d-\epsilon})$?

Our Results for Satisfiability

We give evidence that the answer to all four of the above questions is negative: If any answer is positive then coNP is in NP/poly . The latter is considered unlikely as it means the existence of a nonuniform polynomial-time proof system for tautologies, or equivalently, that coNP has polynomial-size nondeterministic circuits, and implies that the polynomial-time hierarchy collapses to its third level [Yap83].

We obtain those statements as corollaries to a more general result, in which we consider the following communication process to decide a language L .

Definition 1.1 (Oracle Communication Protocol). *An oracle communication protocol for a language L is a communication protocol between two players. The first player is given the input x and has to run in time polynomial in the length of the input; the second player is computationally unbounded but is not given any part of x . At the end of the protocol the first player should be able to decide whether $x \in L$. The cost of the protocol is the number of bits of communication from the first player to the second player.*

We often refer to the second player as the oracle. Note that the bits sent by the oracle do not contribute towards the cost. By default the players in an oracle communication protocol are deterministic, but one can consider variants in which one or both players are randomized, nondeterministic, etc.

Satisfiability of n -variable d -CNF formulas has a trivial protocol of cost $O(n^d)$. The following result implies that there is no protocol of cost $O(n^{d-\epsilon})$ unless the polynomial-time hierarchy collapses. In fact, the result even holds when the first player is conondeterministic, i.e., when the first player can have multiple valid moves to choose from in any given step, possibly leading to different conclusions about the satisfiability of a given input formula φ , but such that (i) if φ is satisfiable then every valid execution comes to that conclusion, and (ii) if φ is not satisfiable then at least one valid execution comes to that conclusion.

Theorem 1.1. *Let $d \geq 3$ be an integer and ϵ a positive real. If $\text{coNP} \not\subseteq \text{NP}/\text{poly}$, there is no protocol of cost $O(n^{d-\epsilon})$ to decide whether an n -variable d -CNF formula is satisfiable, even when the first player is conondeterministic.*

The proof of this theorem and its corollaries is in §3.3. The corollaries about sparsification, kernelization, and lossy compression follow by considering deterministic single-round protocols in which the polynomial-time player acts as a mapping reduction, sends the reduced instance to the computationally unbounded player, and the latter answers this query as a membership oracle. The corollary about probabilistically checkable proofs follows by considering a similar single-round protocol in which the first player is conondeterministic. Note that Theorem 1.1 can handle more general reductions, in which multiple queries are made to the oracle over multiple rounds. The above corollaries can be strengthened correspondingly. In fact, Theorem 1.1 is morally even more general as it allows the oracle to play an active role that goes beyond answering queries from the polynomial-time player. We discuss this potential further in §3.6.

Our Results for Covering Problems

By reducibility the lower bounds from Theorem 1.1 carry over to other parameterized NP-complete problems, where the tightness depends on how the reduction affects the parameterization. In fact, we derive Theorem 1.1 from a similar result for the vertex cover problem on d -uniform hypergraphs.

Theorem 1.2. *Let $d \geq 2$ be an integer and ϵ a positive real. If $\text{coNP} \not\subseteq \text{NP/poly}$, there is no protocol of cost $O(n^{d-\epsilon})$ to decide whether a d -uniform hypergraph on n vertices has a vertex cover of at most k vertices, even when the first player is conondeterministic.*

We prove this theorem in §3.2. The cases of Theorem 1.2 with $d \geq 3$ are equivalent to the corresponding cases of Theorem 1.1. Note, though, that Theorem 1.2 also holds for $d = 2$, i.e., for standard graphs.

Similar to Theorem 1.1, Theorem 1.2 can be interpreted in terms of (graph) sparsification, kernelization, lossy compression, and probabilistically checkable proofs. Regarding kernelization, Theorem 1.2 has an interesting implication for the vertex cover problem parameterized by the size of the vertex cover – one of the prime examples of a parameterized problem that is NP-hard but fixed-parameter tractable. Kernelizations for this problem have received considerable attention. For standard graphs S. Buss [BG93] came up with a kernelization *avant la lettre*. He observed that any vertex of degree larger than k must be contained in any vertex cover of size k , should it exist. This gives rise to a kernelization with $O(k^2)$ vertices and $O(k^2)$ edges. Subsequently, several researchers tried to reduce the size of the kernel. Various approaches based on matching, linear programming, and crown reductions (see [GN07] for a survey) led to kernels with $O(k)$ vertices, one of which we review in §2.1, but the resulting kernels are all dense. It remains open to find kernels with $O(k^{2-\epsilon})$ edges. Since $k \leq n$, the case $d = 2$ of Theorem 1.2 implies that such kernels do not exist unless the polynomial-time hierarchy collapses.

In fact, a similar result holds for a wide class of covering-type problems known as vertex deletion problems. For a fixed graph property Π , the corresponding vertex deletion problem asks whether removing at most k vertices from a given graph G can yield a graph that satisfies Π . A host of well-studied specific problems can be cast as the vertex deletion problem corresponding to some graph property Π that is inherited by subgraphs. Examples besides the vertex cover problem include the feedback vertex set problem and the bounded-degree deletion problem (see §3.4 for the definitions of these problems and for more examples).

If only finitely many graphs satisfy Π or if all graphs satisfy Π , the vertex deletion problem is trivially decidable in polynomial time. For all other graph properties Π that are inherited by subgraphs, Lewis and Yannakakis [LY80] showed that the problem is NP-hard. They did so by constructing a mapping reduction from the vertex cover problem. By improving their reduction such that it preserves the size of the deletion set up to a constant factor, we obtain the following result.

Theorem 1.3. *Let Π be a graph property that is inherited by subgraphs, and is satisfied by infinitely many but not all graphs. Let ϵ be a positive real. If $\text{coNP} \not\subseteq \text{NP/poly}$,*

1. Introduction

there is no protocol of cost $O(k^{2-\epsilon})$ for deciding whether a graph satisfying Π can be obtained from a given graph by removing at most k vertices, even when the first player is conondeterministic.

The proof is in §3.4. Theorem 1.3 implies that problems like feedback vertex set and bounded-degree deletion do not have kernels consisting of $O(k^{2-\epsilon})$ edges unless the polynomial-time hierarchy collapses. For both problems the result is tight in the sense that kernels with $O(k^2)$ edges exist. For feedback vertex set we argue that a recent kernelization by Thomassé [Tho09] does the job; for bounded-degree deletion, kernels with $O(k^2)$ edges were known to exist [FGMN09].

Our Results for Packing Problems

The matching problem in d -uniform hypergraphs, d -SET MATCHING, is to decide whether a given hypergraph has a matching of size k , i.e., a set of k pairwise disjoint hyperedges. Correspondingly, the PERFECT d -SET MATCHING problem is to find a *perfect* matching, i.e., a matching with $k = n/d$ where n is the number of vertices. Fellows et al. [FKN⁺08] show that d -SET MATCHING has kernels with $O(k^d)$ hyperedges.

Theorem 1.4 ([FKN⁺08]). *The problem d -SET MATCHING has kernels with $O(k^d)$ hyperedges.*

In Appendix B, we sketch a straightforward but instructive proof of this fact using the sunflower lemma of Erdős and Rado [ER60]. We use our lower bound technology for oracle communication protocols to prove that the kernel size above is asymptotically optimal under the hypothesis $\text{coNP} \not\subseteq \text{NP/poly}$.

Theorem 1.5. *Let $d \geq 3$ be an integer and ϵ a positive real. If $\text{coNP} \not\subseteq \text{NP/poly}$, there is no protocol of cost $O(k^{d-\epsilon})$ for PERFECT d -SET MATCHING.*

A particularly well-studied special case of set matching is when the sets are certain fixed subgraphs (e.g., triangles, cliques, stars, etc.) of a given graph. We use the terminology of Yuster [Yus07], who surveys graph theoretical properties of such graph packing problems. Formally, an H -*matching* of size k in a graph G is a collection of k vertex-disjoint subgraphs of G that are isomorphic to H . The problem H -MATCHING is to find an H -matching of a given size in a given graph. Both problems are NP-complete whenever H contains a connected component with more than two vertices [KH78] and is in P otherwise.

The kernelization properties of graph packing problems received a lot of attention in the literature (e.g., [Mos09, FHR⁺04, PS04, FR09, WNFC10, MPS04]). H -MATCHING can be expressed as a d -SET MATCHING instance with $O(k^d)$ edges (where $d := |V(H)|$) and therefore Theorem 1.4 implies a kernel of size $O(k^d)$. In the particularly interesting special case when H is a clique K_d , we use a simple reduction to transfer the above theorem to obtain a lower bound for K_d -MATCHING.

Theorem 1.6. *Let $d \geq 4$ be an integer and ϵ a positive real. If $\text{coNP} \not\subseteq \text{NP/poly}$, there is no protocol of cost $O(k^{d-1-\epsilon})$ for K_d -MATCHING.*

An upper bound of size $O(k^d)$ follows for K_d -MATCHING from Theorem 1.4. This does not quite match our conditional lower bounds of $O(k^{d-1-\epsilon})$, and it is an interesting open problem to make the bounds tight.

The H -FACTOR problem is the restriction of H -MATCHING to the case $k = n/d$, i.e., the goal is to find an H -matching that involves all vertices. Unlike the case of matching d -sets, where we had the same bounds for PERFECT d -SET MATCHING and d -SET MATCHING, we cannot expect that the same bounds hold always for H -MATCHING and H -FACTOR. The reason is that for H -FACTOR there is a trivial $O(k^2)$ upper bound on the kernel size for *every* graph H : an n -vertex instance has size $O(n^2)$ and we have $k = \Theta(n)$ by the definition of H -FACTOR. We show that this bound is tight for every NP-hard H -FACTOR problem. Thus, we cannot reduce H -FACTOR to sparse instances.

Theorem 1.7. *Let H be a connected graph with $d \geq 3$ vertices and ϵ a positive real. If $\text{coNP} \not\subseteq \text{NP/poly}$, there is no protocol of cost $O(k^{2-\epsilon})$ for K_d -FACTOR.*

Obviously, Theorem 1.7 gives a lower bound for the more general H -MATCHING problem. In particular, it proves the missing $d = 3$ case in Theorem 1.6.

Obtaining tight bounds for H -MATCHING seems to be a challenging problem in general. As Theorem 1.6 shows in the case of cliques, the lower bound of $O(k^{2-\epsilon})$ implied by Theorem 1.7 is not always tight. We demonstrate that the upper bound of $O(k^{|V(H)|})$ is not always tight either. A simple argument shows that if H is a star of arbitrary size, then a kernel of size $O(k^2)$ is possible, which is tight by Theorem 1.7. The examples of cliques and stars show that the exact bound on the kernel size of H -MATCHING for a particular H could be very far from the weak $O(k^{|V(H)|})$ upper bound or the weak $O(k^{2-\epsilon})$ lower bound (Theorem 1.7). Full understanding of this question seems to be a very challenging, yet very natural problem. The proofs of all packing-related results are in §3.5.

Techniques and Related Work

At a high level our approach refines the framework developed by Bodlaender et al. [BDFH09] to show that certain parameterized NP-hard problems are unlikely to have kernels of polynomial size. Harnik and Naor [HN06] realized the connection between their notion of lossy compression, kernelization, and PCPs for satisfiability of general Boolean formulas, and Fortnow and Santhanam [FS08] proved the connection with the hypothesis $\text{coNP} \not\subseteq \text{NP/poly}$ in the superpolynomial setting. Several authors subsequently applied the framework to prove polynomial kernel lower bounds under this hypothesis [CFM07, BTY09, DLS09, FFL⁺09, KW10, KW09, KMW10].

We develop the first application of the framework in the polynomial setting, i.e., to problems that *do* have kernels of polynomial size, or more generally, oracle communication protocols of polynomial cost. Under the same hypothesis we show that problems like d -SAT, vertex cover, or set packing do not have protocols of polynomial cost of degree less than the best known. *All* known kernel lower bounds for fixed-parameter tractable problems use, at least implicitly, the following lemma as a proxy. The lemma follows

1. Introduction

along the lines of the proof of Fortnow and Santhanam [FS08], and we generalize it to the oracle communication model in §3.1.

Lemma 1.1 (OR Incompressibility Lemma).

For any NP-hard language L and polynomially bounded function $t : \mathbb{N} \rightarrow \mathbb{N} \setminus \{0\}$, the problem $\text{OR}^t(L)$ does not have a compression of size $O(t \log t)$ unless $\text{coNP} \subseteq \text{NP}/\text{poly}$.

Here $\text{OR}^t(L)$ is the problem whose yes-instances are t -tuples of instances of L each of the same size s such that $t = t(s)$ and at least one of the instances is a yes-instance of L . Under the assumption $\text{coNP} \not\subseteq \text{NP}/\text{poly}$, this lemma says that the problem $\text{OR}^t(L)$, seen as a parameterized problem with parameter s , does not have kernels of size $O(t(s) \log t(s))$ even if we relax our notion of kernelization to compressions, in which the target language of the reduction may differ from the source language. We review how this theorem can be used to prove conditional polynomial kernel lower bounds for the k -path problem in §2.2.

Our main result, Theorem 1.2, deals with the vertex cover problem on d -uniform hypergraphs, or equivalently, with the clique problem on such graphs, parameterized by the number of vertices. Assuming the above incompressibility lemma, the proof of this result consists of a polynomial-time mapping reduction from $\text{OR}(L)$ to the clique problem such that t -tuples of instances of size s are mapped to instances with few vertices $n = n(s, t)$. Then any assumed kernelization of size $O(n^c)$ for the clique problem can be combined with this reduction to give a compression of size $O(n^c)$ for $\text{OR}^t(L)$. As observed by Harnik and Naor [HN06], the disjoint union of the given hypergraphs provides a reduction from $\text{OR}(L)$ to L if L is the clique problem itself. However, the number of vertices is $n = s \cdot t$, so even if $c = 1$, the size of the compression for $\text{OR}(L)$ is $\omega(t \log t)$, which is too much for Lemma 1.1 to apply. As a critical piece in our proof, we present a reduction from $\text{OR}(3\text{-SAT})$ to clique that only needs $n = s \cdot t^{1/d}$ vertices. The size of the combined compression for $\text{OR}(3\text{-SAT})$ then goes down to $O(n^c) = O((s \cdot t^{1/d})^c)$, which is $O(t \log t)$ for some sufficiently large polynomial $t(s)$ as long as $c < d$.

In fact, we have two reductions for clique that yield these tight results: one is elementary and the other hinges on a graph packing that is based on high-density subsets of the integers without nontrivial arithmetic progressions of length three. For Theorem 1.7, we give a proof using the latter construction. After we developed the construction based on arithmetic progression free sets, we have learned about other applications of those sets in the theory of computing, including three-party communication protocols [CFL83], the asymptotically best known algorithm for matrix multiplication [CW90], the soundness analysis of graph tests for linearity [HW03], and lower bounds for property testing [AFKS00, Alo02, AS06, AS04, AKKR08, AS05]. The latter two applications as well as ours implicitly or explicitly rely on a connection due to Ruzsa and Szemerédi [RS78] between these subsets and dense three-partite graphs whose edges partition into triangles and that contain no other triangles. The graph packing we develop is most akin to a construction by Alon and Shapira [AS05] in the context of property testing. We refer to §4 for a more detailed discussion of the relationships.

We can generalize Lemma 1.1 to show that whenever $\text{OR}^t(L)$ has a low-cost protocol, the complement of L has short witnesses that can be verified efficiently with the help

of a polynomial-size advice string, i.e., $\overline{L} \in \text{NP/poly}$. We refer to this generalization as the *Complementary Witness Lemma* and we prove it in §3.1. It involves a refined analysis and generalization of the proof of Fortnow and Santhanam [FS08] that establishes the case where the protocol implements a mapping reduction to instances of bitlength bounded by some fixed polynomial in s . We analyze what happens for mapping reductions without the latter restriction and we observe that the argument generalizes to our oracle communication protocol setting. Our applications of Theorem 1.1 only use oracle communication protocols that implement mapping reductions or general reductions. For the results concerned with mapping reduction, Lemma 1.1 would suffice. However, the setting of oracle communication protocols is more natural and allows us to prove known results in a simpler way, as we discuss in §3.6.

1.2. Sparse Instances of Counting Problems

The permanent of a matrix and the Tutte polynomial of a graph are central topics in the study of counting algorithms. Originally defined in the combinatorics literature, they unify and abstract many enumeration problems, including immediate questions about graphs such as computing the number of perfect matchings, spanning trees, forests, colourings, certain flows and orientations, but also less obvious connections to other fields, such as link polynomials from knot theory, reliability polynomials from network theory, and (maybe most importantly) the Ising and Potts models from statistical physics.

From its definition (repeated in (1.1) below), the permanent of an $n \times n$ -matrix can be computed in $O(n!n)$ time, and the Tutte polynomial (1.2) can be evaluated in time exponential in the number of edges. Both problems are famously #P-hard, which rules out the existence of polynomial-time algorithms under standard complexity-theoretic assumptions, but that does not mean that we have to resign ourselves to brute-force evaluation of the definition. In fact, Ryser’s famous formula [Rys63] computes the permanent with only $\exp(O(n))$ arithmetic operations, and more recently, an algorithm with running time $\exp(O(n))$ for n -vertex graphs has also been found [BHKK08] for the Tutte polynomial. Curiously, both of these algorithms are based on the inclusion–exclusion principle. In this thesis, we show that these algorithms cannot be significantly improved, by providing conditional lower bounds of $\exp(\Omega(n))$ for both problems.

It is clear that #P-hardness is not the right conceptual framework for such claims, as it is unable to distinguish between different types of super-polynomial time complexities. For example, the Tutte polynomial for planar graphs remains #P-hard, but can be computed in time $\exp(O(\sqrt{n}))$ [SIT95]. Therefore, we work under the *Exponential Time Hypothesis* (ETH), viz. the complexity theoretic assumption that deciding the satisfiability of 3-CNF formulas in n variables requires time $\exp(\Omega(n))$. More specifically, we introduce #ETH, a counting analogue of ETH which models the hypothesis that *counting* the satisfying assignments requires time $\exp(\Omega(n))$.

1. Introduction

Computing the permanent

The permanent of an $n \times n$ matrix A is defined as

$$\text{per } A = \sum_{\pi \in S_n} \prod_{1 \leq i \leq n} A_{i\pi(i)}, \quad (1.1)$$

where S_n is the set of permutations of $\{1, \dots, n\}$. This is similar to the determinant from linear algebra, $\det A = \sum_{\pi} \text{sign}(\pi) \prod_i A_{i\pi(i)}$, the only difference is an easily computable sign for every summand. Both definitions involve a summation with $n!$ terms, but admit much faster algorithms that are textbook material: The determinant can be computed in polynomial time using Gaussian elimination and the permanent can be computed in $O(2^n n)$ operations using Ryser's formula.

Valiant's celebrated #P-hardness result [Val79] for the permanent shows that no polynomial-time algorithm à la "Gaussian elimination for the permanent" can exist unless $P = NP$, and indeed unless $P = P^{\#P}$. Several unconditional lower bounds for the permanent in restricted models of computation are also known. [JS82] have shown that monotone arithmetic circuits need $n(2^{n-1} - 1)$ multiplications to compute the permanent, a bound they can match with a variant of Laplace's determinant expansion. [Raz09] has shown that multi-linear arithmetic formulas for the permanent require size $\exp(\Omega(\log^2 n))$. Ryser's formula belongs to this class of formulas, but is much larger than the lower bound; no smaller construction is known. Intriguingly, the same lower bound holds for the determinant, where it is matched by a formula of size $\exp(O(\log^2 n))$ due to [Ber84]. One of the consequences of this thesis is that Ryser's formula is in some sense optimal under #ETH. In particular, no uniformly constructible, subexponential size formula such as Berkowitz's can exist for the permanent unless #ETH fails.

A related topic is the expression of $\text{per } A$ in terms of $\det f(A)$, where $f(A)$ is a matrix of constants and entries from A and is typically much larger than A . This question has fascinated many mathematicians for a long time, see Agrawal's survey [Agr06]; the best known bound on the dimension of $f(A)$ is $\exp(O(n))$ and it is conjectured that all such constructions require exponential size. In particular, it is an important open problem if a permanent of size n can be expressed as a determinant of size $\exp(O(\log^2 n))$. We show that under #ETH, if such a matrix $f(A)$ exists, computing f must take time $\exp(\Omega(n))$.

Computing the Tutte polynomial

The Tutte polynomial, a bivariate polynomial associated with a given graph $G = (V, E)$ with n vertices and m edges, is defined as

$$T(G; x, y) = \sum_{A \subseteq E} (x - 1)^{k(A) - k(E)} (y - 1)^{k(A) + |A| - |V|}, \quad (1.2)$$

where $k(A)$ denotes the number of connected components of the subgraph (V, A) .

Despite their unified definition (1.2), the various computational problems given by $T(G; x, y)$ for different points (x, y) differ widely in computational complexity, as well as in the methods used to find algorithms and lower bounds.

For example, $T(G; 1, 1)$ equals the number of spanning trees in G , which happens to admit a polynomial time algorithm, curiously again based on Gaussian elimination. On the other hand, the best known algorithm for computing $T(G; 2, 1)$, the number of forests, runs in $\exp(O(n))$ time.

Computation of the Tutte polynomial has fascinated researchers in computer science and other fields for many decades. For example, the algorithms of Onsager and Fischer from the 1940s and 1960s for computing the so-called partition function for the planar Ising model are viewed as major successes of statistical physics and theoretical chemistry; this corresponds to computing $T(G; x, y)$ along the hyperbola $(x - 1)(y - 1) = 2$ for planar G . Many serious attempts were made to extend these results to other hyperbolas or graph classes, but “after a quarter of a century and absolutely no progress”, Feynman in 1972 observed that “the exact solution for three dimensions has not yet been found”.¹

The failure of theoretical physics to “solve the Potts model” and sundry other questions implicit in the computational complexity of the Tutte polynomial were explained only with Valiant’s #P-hardness programme. After a number of papers, culminating in [JVW90], the polynomial-time complexity of exactly computing the Tutte polynomial at points (x, y) is now completely understood: it is #P-hard everywhere except at those points (x, y) where a polynomial-time algorithm is known; these points consist of the hyperbola $(x - 1)(y - 1) = 1$ as well as the four points $(1, 1), (-1, -1), (0, -1), (-1, 0)$.

In this thesis, we show an $\exp(\Omega(n))$ lower bound to match the $\exp(O(n))$ algorithm from [BHKK08], which holds under #ETH everywhere except for $|y| = 1$. In particular, this establishes a gap to the planar case, which admits an $\exp(O(\sqrt{n}))$ algorithm [SIT95]. Our hardness results apply (though not everywhere, and sometimes with a weaker bound) even if the graphs are sparse and simple. These classes are of particular interest because most of the graphs arising from applications in statistical mechanics arise from bond structures, which are sparse and simple.

It has been known since the 1970s [Law76] that graph 3-colouring can be solved in time $\exp(O(n))$, and this is matched by an $\exp(\Omega(n))$ lower bound under ETH [IPZ01]. Since graph 3-colouring corresponds to evaluating T at $(-2, 0)$, the exponential time complexity for $T(G; -2, 0)$ was thereby already understood. In particular, computing $T(G; x, y)$ for input G and (x, y) requires vertex-exponential time, an observation that is already made in [GHN06] without explicit reference to ETH.

The literature for computing the Tutte polynomial is very rich, and we make no attempt to survey it here. A recent paper of [GJ08], which shows that the Tutte polynomial is hard to even approximate for large parts of the Tutte plane, contains an overview. A list of graph classes for which subexponential time algorithms are known can be found in [BHKK08].

Results

Our hardness results are based on two versions of the exponential time hypothesis. The (randomized) exponential time hypothesis (ETH) as defined in [IPZ01] is that satisfi-

¹The Feynman quote and many other quotes describing the frustration and puzzlement of physicists around that time can be found in the copious footnotes of [Ist00].

1. Introduction

ability of 3-CNF formulas cannot be computed substantially faster than by trying all possible assignments, i.e., it requires time $\exp(\Omega(n))$. Formally, this reads as follows:

(ETH) There is a constant $c > 0$ such that no randomized algorithm can decide 3-SAT in time $\exp(c \cdot n)$ with error probability at most $\frac{1}{3}$.

The second hypothesis we are using is the (deterministic) counting exponential time hypothesis, based on the counting variant of 3-SAT.

Name #3-SAT

Input 3-CNF formula φ with n variables and m clauses.

Output The number of satisfying assignments to φ .

The best known algorithm for this problem runs in time $O(1.6423^n)$ [Kut07].

(#ETH) There is a constant $c > 0$ such that no deterministic algorithm can compute #3-SAT in time $\exp(c \cdot n)$.

ETH trivially implies #ETH whereas the other direction is not known. By introducing the sparsification lemma, [IPZ01] show that ETH is a robust notion in the sense that the clause width 3 and the parameter n in its definition can be replaced by $d \geq 3$ and m , respectively, to get an equivalent hypothesis. The constant c may change in doing so. We transfer the sparsification lemma to # d -SAT and get a similar kind of robustness for #ETH, the proof of which is spelled out in §C.

Theorem 1.8. *Let $d \geq 3$ be an integer. Then #ETH holds if and only if there is a constant $c > 0$ such that no deterministic algorithm can solve # d -SAT time $\exp(c \cdot m)$.*

For convenience, we say that # d -SAT requires time $\exp(\Omega(m))$.

Counting Independent Sets

In light of Theorem 1.8, it is natural to consider the exponential time complexity of #2-SAT. Restricted to antimonotone 2-CNF formulas, this corresponds to counting *all* independent sets in a given graph, which Hoffmann [Hof10] shows to require time $\exp(\Omega(n/\log^3 n))$ under #ETH. The loss of the poly log-factor in the exponent is due to the interpolation inherent in the hardness reduction. We avoid interpolation using the isolation lemma for d -CNF formulas [CIKP03] and get an asymptotically tight lower bound, with the drawback that our lower bound only holds under the (randomized) exponential time hypothesis ETH instead of #ETH.

Theorem 1.9. *Counting independent sets and #2-SAT require time $\exp(\Omega(m))$ under ETH, where m is the number of edges and clauses, respectively.*

We discuss the isolation technique and prove this theorem in §5.1.

The Permanent

For a set S of rationals we define the following problems:

Name PERM^S

Input Square matrix A with entries from S .

Output The value of $\text{per } A$.

We write PERM for $\text{PERM}^{\mathbb{N}}$. If B is a bipartite graph with A_{ij} edges from the i th vertex in the left half to the j th vertex in the right half ($1 \leq i, j \leq n$), then $\text{per}(A)$ equals the number of perfect matchings of B . Thus PERM and $\text{PERM}^{0,1}$ can be viewed as counting the perfect matchings in bipartite multigraphs and bipartite simple graphs, respectively.

We express our lower bounds in terms of m , the number of non-zero entries of A . Without loss of generality, $n \leq m$, so the same bounds hold for the parameter n as well. Note that these bounds imply that the hardest instances have roughly linear density.

Theorem 1.10.

- (i) $\text{PERM}^{-1,0,1}$ and PERM require time $\exp(\Omega(m))$ under $\#ETH$.
- (ii) PERM^{01} requires time $\exp(\Omega(m/\log n))$ under $\#ETH$.
- (iii) PERM^{01} requires time $\exp(\Omega(m))$ under ETH .

The proof of this theorem is in §5.2. For (i), we follow a standard reduction by Valiant [Val79, Pap94] but use a simple equality gadget derived from [BD07] instead of Valiant's XOR-gadget, and we use interpolation to get rid of the negative weights. To establish (ii) we simulate edge weights $w > 1$ by gadgets of size logarithmic in w , which increases the number of vertices and edges by a logarithmic factor. For (iii) we use the isolation lemma and the reduction from part (i), and we simulate the edge weights -1 without interpolation by replacing them with 2 and doing computation modulo 3.

The Tutte Polynomial

The computational problem $\text{TUTTE}(x, y)$ is defined for each pair (x, y) of rationals.

Name $\text{TUTTE}(x, y)$.

Input Undirected multigraph G with n vertices.

Output The value of $T(G; x, y)$.

In general, parallel edges and loops are allowed; we write $\text{TUTTE}^{01}(x, y)$ for the special case where the input graph is simple.

Our main result is that under $\#ETH$, $\text{TUTTE}(x, y)$ requires time $\exp(\Omega(n))$ for specific points (x, y) , but the size of the bound, and the graph classes for which it holds, varies. We summarise our results in the theorem below, see also Figure 1.1. Our reductions often give edge-exponential lower bounds, i.e., bounds in terms of the parameter m , which implies the same bounds in terms of n because $m \geq n$ in connected graphs.

1. Introduction

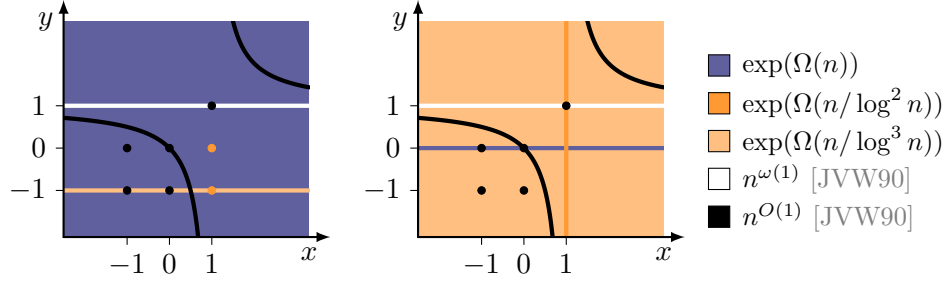


Figure 1.1.: Exponential time complexity under #ETH of the Tutte plane for multigraphs (left) and simple graphs (right) in terms of n , the number of vertices. The white line $y = 1$ on the map is uncharted territory. The black hyperbola $(x - 1)(y - 1) = 1$ and the four points close to the origin are in P. Everywhere else, in the shaded regions, we prove a lower bound exponential in n , or within a polylogarithmic factor of it.

Moreover, a lower bound of $\exp(\Omega(m/\text{poly log } m))$ together with the algorithm in time $\exp(O(n))$ from [BHK08] implies that worst-case instances are *sparse*, in the sense that $m = O(n \cdot \text{poly log } n)$.

Theorem 1.11. *Let $(x, y) \in \mathbb{Q}^2$. Under #ETH,*

- (i) ■ $\text{TUTTE}(x, y)$ requires time $\exp(\Omega(n))$ if $(x - 1)(y - 1) \neq 1$ and $y \notin \{0, \pm 1\}$,
- (ii) ■ $\text{TUTTE}^{01}(x, y)$ requires time $\exp(\Omega(n))$ if $y = 0$ and $x \notin \{0, \pm 1\}$,
- (iii) ■ $\text{TUTTE}^{01}(x, y)$ requires time $\exp(\Omega(m/\log^2 m))$ if $x = 1$ and $y \neq 1$,
- (iv) ■ $\text{TUTTE}^{01}(x, y)$ requires time $\exp(\Omega(m/\log^3 m))$ if $(x - 1)(y - 1) \notin \{0, 1\}$ and $(x, y) \notin \{(-1, -1), (-1, 0), (0, -1)\}$.

In an attempt to prove these results, we may first turn to the literature, which contains a cornucopia of constructions for proving hardness of the Tutte polynomial in various models. In these arguments, a central role is played by graph transformations called thickenings and stretches. A k -thickening replaces every edge by a bundle of k edges $\textcircled{\scriptsize k}$, and a k -stretch replaces every edge by a path of k edges $\textcircled{\scriptsize k}$. This is used to ‘move’ an evaluation from one point to another. For example, if H is the 2-stretch of G then $T(H; 2, 2) \sim T(G; 4, \frac{4}{3})$. Thus, every algorithm for $(2, 2)$ works also at $(4, \frac{4}{3})$, connecting the complexity of the two points. These reductions are very well-developed in the literature, and are used in models that are immune to polynomial-size changes in the input parameters, such as #P-hardness and approximation complexity. However, we cannot always afford such constructions in our setting, otherwise our bounds would

be of the form $\exp(\Omega(n^{1/r}))$ for some constant r depending on the blowup in the proof. In particular, the parameter n is destroyed already by a 2-stretch in a nonsparse graph.

The proofs are in §5.3. Where we can, we sample from established methods, carefully avoiding or modifying those that are not parameter-preserving. At other times we require more subtle techniques, e.g., the constructions in §5.3.3, which use graph products with graphs of polylogarithmic size instead of thickenings and stretches. Like many recent papers, we use Sokal's multivariate version of the Tutte polynomial, which vastly simplifies many of the technical details.

Consequences

The permanent and Tutte polynomial are equivalent to, or generalisations of, various other graph problems, so our lower bounds under ETH and #ETH hold for these problems as well. In particular, it takes time $\exp(\Omega(m))$ to compute the following graph polynomials (for example, as a list of their coefficients) for a given simple graph: the Ising partition function, the q -state Potts partition function ($q \neq 0, 1, 2$), the reliability polynomial, the chromatic polynomial, and the flow polynomial. Moreover, we have $\exp(\Omega(n))$ lower bounds for the following counting problems on multigraphs: # perfect matchings, # cycle covers in digraphs, # connected spanning subgraphs, all-terminal graph reliability with given edge failure probability $p > 0$, # nowhere-zero k -flows ($k \neq 0, \pm 1$), and # acyclic orientations.

The lower bound for counting the number of perfect matchings holds even in bipartite graphs, where an $O(1.414^n)$ algorithm is given by Ryser's formula. Such algorithms are also known for general graphs [BH08a], the current best bound is $O(1.619^n)$ [Koi09].

For simple graphs, we have $\exp(\Omega(m))$ lower bounds for # perfect matchings and # cycle covers in digraphs.

2. Preliminaries

Most of our notation is standard (see [AB09, Gol08] for general and [DF99, FG06, Nie06] for parameterized complexity). We suffice with a review of some particular notions and notation we use.

Problems By a problem we usually mean a decision problem, i.e., deciding membership to a language $L \subseteq \{0, 1\}^*$. Apart from their bitlength $|x|$, instances $x \in \{0, 1\}^*$ often have another natural complexity parameter $k(x)$, such as the number of vertices in the case of graph problems, or the witness length in the case of NP-problems. The function $k : \{0, 1\}^* \rightarrow \mathbb{N}$ is called *parameterization* and a *parameterized problem* is a pair (L, k) . We often write L for both the parameterized and unparameterized problem, e.g., when saying that a parameterized problem is NP-complete.

We denote the *complement* of L by \bar{L} . The *OR of a language* L is the language $\text{OR}(L)$ that consists of all tuples (x_1, \dots, x_t) with $t > 0$ for which there is an $i \in [t]$ with $x_i \in L$. An *OR-problem with arity* $t : \mathbb{N} \rightarrow \mathbb{N} \setminus \{0\}$ is a problem $\text{OR}(L)$ restricted to instances that satisfy the additional properties $s = |x_1| = \dots = |x_t|$ and $t = t(s)$ for some $s \in \mathbb{N}$. We denote this problem by $\text{OR}^t(L)$.

Satisfiability A d -CNF formula on the variables x_1, \dots, x_n is a conjunction of clauses where a clause is a disjunction of exactly d literals, i.e., the variables x_i and their negations \bar{x}_i . We denote by d -SAT the problem of deciding whether a given d -CNF formula has at least one satisfying assignment, i.e., a truth assignment to its variables that makes the formula evaluate to true.

Hypergraph Problems A hypergraph $G = (V(G), E(G))$ consists of a finite set $V(G)$ of vertices and a set $E(G)$ of subsets of $V(G)$, the (hyper)edges. A hypergraph is d -uniform if every edge has size exactly d . A *vertex cover of* G is a set $S \subseteq V(G)$ that contains at least one vertex from every edge of G , and d -VERTEX COVER is the problem of deciding whether, for a given d -uniform hypergraph G and integer k , there exists a vertex cover of G of size at most k . Similarly, a *clique of* G is a set $S \subseteq V(G)$ all of whose subsets of size d are edges of G , and d -CLIQUE is the problem of deciding whether, for given (G, k) , there exists a clique of G of size at least k . The two problems are dual to each other, in the sense that \bar{G} , the d -uniform hypergraph obtained from G by flipping the presence of all edges of size d , has a clique of size k if and only if G has a vertex cover of size $n - k$. Note that this transformation preserves the number of vertices.

Reductions Unless stated otherwise the reductions we consider are computable in time polynomial in the bitlength of the input. We indicate this by a superscript p in the

2. Preliminaries

notation \leq^p for reducibility. We consider both general reductions (also known as Turing reductions) as well as mapping reductions (also known as many-one reductions). A *mapping reduction*, or \leq_m^p -*reduction*, from L to L' is a mapping R from $\{0, 1\}^*$ to $\{0, 1\}^*$ such that $R(x) \in L'$ if and only if $x \in L$.

A *compression* of (L, k) is a \leq_m^p -reduction from L to some language L' that maps instances with parameter k to instances of bitlength at most $g(k)$ for some computable function g independent of the input size. A *kernelization* is a compression with $L = L'$. Note that any decidable parameterized problem that has a kernelization is fixed-parameter tractable, that is, it can be solved in deterministic time $f(k) \cdot \text{poly}(n)$ for some computable function f : The reduced instance has size at most $g(k)$ and can be solved using any algorithm for L .

Complexity Classes The polynomial-time hierarchy PH is the union $\cup_{i \geq 0} \Sigma_i^p$, where $\Sigma_0^p = \text{P}$, and $\Sigma_{i+1}^p = \text{NP}^{\Sigma_i^p}$ for $i \geq 0$. We say that the polynomial-time hierarchy collapses to its i th level if $\text{PH} = \Sigma_i^p$. It is widely conjectured that the polynomial-time hierarchy does not collapse to any level.

Given a class \mathcal{C} of languages, we denote by $\text{co}\mathcal{C}$ the class $\{\bar{L} \mid L \in \mathcal{C}\}$. Apart from the first few levels of the polynomial-time hierarchy and their co-classes, we make use of complexity classes with advice. Given a class \mathcal{C} of languages and a function $\ell : \mathbb{N} \rightarrow \mathbb{N}$, we denote by $\mathcal{C}/\ell(n)$ the class of languages L for which there exists a language $L' \in \mathcal{C}$ and a sequence a_0, a_1, a_2, \dots of strings with $|a_n| \leq \ell(n)$ such that for any input x , we have that $x \in L$ if and only if $\langle x, a_{|x|} \rangle \in L'$, where $\langle \cdot, \cdot \rangle$ denotes a standard pairing function. We call a_n the advice at length n . \mathcal{C}/poly is a shorthand for $\cup_{c > 0} \mathcal{C}/n^c$. P/poly consists exactly of the languages that can be decided by Boolean circuits of polynomial size. Similarly, NP/poly consists exactly of the languages that can be decided by nondeterministic Boolean circuits of polynomial size. A nondeterministic circuit has two types of inputs – the actual input x and auxiliary input y . It accepts an actual input x if and only if there exists a setting of the auxiliary input y such that the circuit outputs 1 on the combined input x and y .

Communication Protocols In general, a two-player communication protocol is described by strategies that tell each of the players when and what to communicate to the other player and how to further behave as a function of the input and the communication history. In the specific case of our oracle communication protocols of Definition 1.1, there is an asymmetry between the two players. We model the first player as a polynomial-time Turing machine M and the second player as a function f . The machine M has a special oracle query tape, oracle query symbol, and oracle answer tape. Whenever M writes the special oracle query symbol on the oracle query tape, in a single computation step the contents of the answer tape is replaced by $f(q)$, where q represents the contents of the oracle query tape at that time. Note that the function f is independent of M 's input x , which reflects the fact that the second player does not have direct access to the input. The oracle query tape is one-way and is never erased, which allows the strategy of the second player to depend on the entire communication history.

We say that the oracle communication protocol decides a parameterized problem (L, k) if M with oracle f accepts an input x if and only if $x \in L$. The cost $c(k)$ of the protocol is the maximal number of bits written on the oracle query tape over all inputs x with parameter $k(x) = k$.

By considering Turing machines other than the standard deterministic model for the first player, we obtain corresponding variants of oracle communication protocols. For example, we can let the first player be a polynomial-time conondeterministic Turing machine. The second player is always modeled as a function. Whenever there are multiple possible valid executions (as in the case of conondeterministic protocols), we define the cost as the maximum cost over all of them, i.e., we consider the worst case.

2.1. Review: Kernelization of Vertex Cover

Vertex cover on graphs is one of the most studied problems in parameterized complexity and many different kernelization algorithms for vertex cover and its variants are known. In this section, we briefly review the fact that vertex cover has kernels with $2k$ vertices. We use the Gallai–Edmonds structure theorem and the crown reduction rule.

As mentioned in the introduction, a simple high-degree argument shows that vertex cover has kernels with $O(k^2)$ vertices. Nemhauser and Trotter [NT74] use the half-integrality of the linear program relaxation of vertex cover to obtain a kernel with $2k$ vertices. Using purely combinatorial techniques, this kernel size is also achievable by applying the so-called crown reduction rule. Here the goal is to find a special structure in the graph – a crown – apply the reduction rule, and repeat this process until no crown can be found anymore. We review a simple analysis of this reduction rule that yields kernels with $3k$ vertices [FG06, Chapter 9.2]. We show how to use the Gallai–Edmonds structure theorem to find a crown which leads to kernels with $2k$ vertices after a single application of the crown reduction rule.

The following lemma constructs a maximum matching M in a graph G and an independent set I .

Lemma 2.1. *Let S be any set of vertices of G . There is a maximum matching M of G and an independent set I such that $N(I) \subseteq S$ and every vertex of $N(I)$ is matched by M to a vertex of I . Furthermore, all singleton components of $G - S$ are contained in $V(M) \cup I$, and M and I can be computed in polynomial time.*

We obtain a *crown with head* $N(I)$, and the *crown reduction* consists of picking all vertices of $N(I)$ into the vertex cover and deleting $N(I) \cup I$ from the graph to obtain the graph $G' = G - (N(I) \cup I)$. Then G' has a vertex cover of size $k - |N(I)|$ if and only if G has a vertex cover of size k . If S is the vertex set of a maximal matching in G , then $G - S$ is an independent set and each vertex of G' is either in S or in $V(M) \setminus S$. If G has a vertex cover of size k , then S has at most $2k$ elements and $V(M) \setminus S$ at most k . Thus, such choice for S yields a kernel G' with at most $3k$ vertices. We now prove Lemma 2.1 using an alternating path argument.

2. Preliminaries

Proof. Let M be a maximum matching of G that touches the maximal number of vertices of $G - S$ that have degree zero in $G - S$. Let I be the set of vertices v_ℓ for which there is a path $v_0, s_1, v_1, \dots, s_\ell, v_\ell$ in G with the following properties:

1. v_0 is in $G - S - V(M)$ and has degree zero in $G - S$, and
2. s_i is matched to v_i by M , for all $1 \leq i \leq \ell$.

We show inductively that all elements of I are in $G - S$ and have degree zero in $G - S$. This implies that the neighborhood of I in G satisfies $N(I) \subseteq S$. In the base case $\ell = 0$, this is clearly the case. For $\ell > 0$, consider any alternating path $v_0, s_1, v_1, \dots, s_\ell, v_\ell$ as above. By induction, the vertices v_i for $i < \ell$ have degree zero in $G - S$ and thus $s_i \in S$ for $i \in [\ell]$. Now M matches s_ℓ to a vertex v_ℓ in G . Assume for contradiction that v_ℓ is not a vertex of $G - S$ which has degree zero in $G - S$. Since the vertex v_0 is not touched by M , we could replace the edges $\{s_i, v_i\}$ of M with the edges $\{v_{i-1}, s_i\}$ and thereby improve M : The size of the matching does not change, but the number of vertices that have degree zero in $G - S$ and that are touched by M increases. This contradicts the fact that we chose M as to maximize that number, and thus, v_ℓ has degree zero in $G - S$. Furthermore, the choice of I guarantees that M matches the vertices of $N(I)$ to vertices in I . ■

The Gallai–Edmonds structure theorem provides us with a different set S , which we feed into Lemma 2.1 to obtain kernels with $2k$ vertices. To succinctly state the theorem, some standard notion is needed. A matching of G is *near-perfect* if it covers all but exactly one vertex. A graph G is *factor-critical* if all subgraphs obtained by removing a single vertex have a perfect matching. The structure given by the Gallai–Edmonds theorem allows us to argue about the location of maximum matchings.

Theorem 2.1 (Gallai–Edmonds).

- (i) Every graph G has a unique Gallai–Edmonds separator S , i.e., a vertex set satisfying the following two properties:
 - (a) All connected components of $G - S$ have a perfect matching or are factor-critical.
 - (b) For every $T \subseteq S$ we have $|N(T)| > |T|$.
- (ii) Every maximum matching of G contains a near-perfect or perfect matching of each component of $G - S$, and it matches all vertices of S to vertices in distinct components of $G - S$.
- (iii) There is a polynomial-time algorithm for computing S .

A short proof of that theorem uses Hall’s Marriage Theorem [Kot00]. We now analyse the reduction suggested by Lemma 2.1 when S is the Gallai–Edmonds separator.

Lemma 2.2. *Let S be the Gallai–Edmonds separator of G , and let M and I be as in Lemma 2.1. Let $G' = G - (N(I) \cup I)$. Then every vertex cover of G' uses at least $|V(G')|/2$ vertices.*

Proof. Let T be some vertex cover of G' . Let C be the set of factor-critical components of $G' - S$ that do not have a vertex matched in S by M . Since $V(M) \cup I$ contains all singleton components of $G - S$, they are not contained in $G' - S$ anymore, so each component of C has at least three vertices. In each component $B \in C$ with b vertices, T clearly uses some vertex v . The removal of that vertex leaves a component $B - v$ with $b - 1$ vertices. Since B is factor-critical, $B - v$ has a perfect matching and T contains at least $(b - 1)/2$ vertices of $B - v$. Summing over all components B of C , this implies that T contains at least $|V(C)|/2$ vertices of C . Now let $G' - C$ be the graph obtained from G' by removing its unmatched factor-critical components. We claim that M induces a perfect matching on $G' - C$. From the claim, it follows immediately that T contains at least $|V(G' - C)|/2$ vertices in $G' - C$. Thus T contains at least $|V(G')|/2$ vertices.

It remains to argue the claim. By Theorem 2.1(i), all components of $G - S$ that have an odd number of vertices are factor-critical, and all components of $G - S$ that have an even number of vertices have a perfect matching. Theorem 2.1(ii) implies that

- M contains no edges inside of S ,
- M matches all vertices of S to odd components of $G - S$, and
- M contains a near-perfect or perfect matching for every component of $G - S$.

In particular, if we remove the set C' of all odd components of G that are not matched to S by M , then M induces a perfect matching on $G - C'$. Now we need to argue that this holds for $G' - C$ as well. If we delete I and $N(I)$ from $G - C'$, we obtain $G' - C$. The only edges of M that intersect $N(I) \cup I$ are those that match a vertex of $N(I)$ to a vertex of I , and both are deleted. All other edges of M that are contained in $G - C'$ remain intact. The claim that M induces a perfect matching on $G' - C$ follows. ■

Corollary 2.1. VERTEX COVER has kernels with at most $2k$ vertices.

Proof. Since M matches vertices of $N(I)$ to vertices of I , any vertex cover must use at least $|N(I)|$ vertices to cover the edges incident to I . On the other hand, all edges incident to I are covered by $N(I)$. Let G' be the graph obtained from G by removing $I \cup N(I)$. Now G' has a vertex cover of size at most $k' = k - |N(I)|$ if and only if G has a vertex cover of size at most k . ■

It is sometimes stated in the literature that the above number of vertices in kernels for vertex cover is optimal, i.e., that its kernels cannot be guaranteed to have at most $1.99 \cdot k$ vertices unless some complexity theoretic assumptions related to the inapproximability of vertex cover fail. However, this is only known in the case that the kernelization size bound does not depend on the parameter k . More precisely, it is assumed that kernelization algorithms are given an instance (G, k) as input, and the task is to output in polynomial time an equivalent instance (G', k') such that G' has at most $c \cdot \text{vc}(G)$ vertices. Here, $\text{vc}(G)$ is the minimum size of a vertex cover of G . It is easy to see that such restricted kernelization algorithms automatically approximate $\text{vc}(G)$ up to a factor of c , so inapproximability results carry over. General kernelization algorithms only guarantee that the number of vertices in the output is $c \cdot k$, and it is an open problem to show that a general kernelization with $c < 2$ is complexity-theoretically unlikely.

2.2. Review: Polynomial Kernel Lower Bounds

We review the world of polynomial kernel lower bounds using our terminology. For an NP-hard problem L , we regard the problem $\text{OR}(L)$ as a parameterized problem with parameter s where $s = \max_i |x_i|$ is the size of a largest given instance. We argue that this problem does not have polynomial kernels unless $\text{coNP} \subseteq \text{NP/poly}$, and that *all* polynomial kernel lower bounds given in the literature are by a suitable parameter-frugal \leq_m^p -reduction from this problem (such reductions are often called “composition” or “polynomial parameter transformation” in the literature). For this to work out seamlessly, we use the more general notion of compression instead of kernelization.

Lemma 2.3 (OR Incompressibility Lemma for Polynomial Lower Bounds).

For any NP-hard language L , the problem $\text{OR}(L)$ does not have a compression of size $\text{poly}(s)$ unless $\text{coNP} \subseteq \text{NP/poly}$.

Proof. Let $t : \mathbb{N} \rightarrow \mathbb{N}_{>0}$ be polynomially bounded and assume that $\text{OR}(L)$ has a compression of size $t(s)$. In particular, this means that the problem $\text{OR}^t(L)$ has a compression of size $t(s)$. By Lemma 1.1, this implies $\text{coNP} \subseteq \text{NP/poly}$. ■

It is apparent from the proof that we can conveniently use the variant of $\text{OR}(L)$ in the above lemma in which all given instances have to have the same size s .

We showcase the simplicity of our terminology by reproving the polynomial kernel lower bound of k -PATH. In this problem, we are given a graph G and a number k and we want to determine whether G contains a simple path of length k . The problem is fixed-parameter tractable and generalizes the NP-complete Hamiltonian path problem. Bodlaender et al. [BDFH09] show that k -PATH does not have kernels of size polynomial in k unless $\text{coNP} \subseteq \text{NP/poly}$. Their proof implicitly gives rise to the following reduction.

Lemma 2.4. *Let L be the Hamiltonian path problem. There is a \leq_m^p -reduction from $\text{OR}(L)$ to k -PATH that maps tuples of instances of bitlength s each to instances of k -PATH with parameter $k \leq s$.*

Proof. In the Hamiltonian path problem, we are given a graph G on n vertices and we want to decide whether it contains a simple path of length n . We assume that L is defined in such a way that graphs are encoded using the characteristic vector of their edge relations, i.e., binary strings of length $\binom{n}{2}$. This technicality makes sure that two graphs with the same encoding length have the same number of vertices, and thus, the lengths of the paths we are looking for are equal.

For the reduction, we are given a t -tuple (x_1, \dots, x_t) of instances of L . Without loss of generality, we assume that $s = |x_1| = \dots = |x_t|$ for some $s \in \mathbb{N}$. If s is not of the form $\binom{n}{2}$, then the x_i 's do not encode graph instances and we return a trivial no instance. Otherwise, these strings encode t graphs G_1, \dots, G_t on n vertices each. Let $G = G_1 \dot{\cup} \dots \dot{\cup} G_t$ be the disjoint union of these graphs. The reduction outputs (G, n) , an instance of the k -path problem with $k = n \leq s$. For the correctness, we observe that some G_i has a Hamiltonian path if and only if G contains a simple path of length n . ■

Corollary 2.2 ([BDFH09]). *If $\text{coNP} \not\subseteq \text{NP/poly}$, then k -PATH does not have kernels of size polynomial in k .*

Proof. If k -PATH has polynomial kernels then, by the above reduction, $\text{OR}(L)$ has a compression of size $\text{poly}(s)$. By Lemma 2.3, this implies $\text{coNP} \subseteq \text{NP/poly}$. ■

3. Communicating Instances of Hard Problems

In this chapter, we investigate a variant of the P vs. NP problem: We model the situation in which a time-bounded agent wants to solve a hard problem and is allowed to communicate with another, arbitrarily powerful agent. The catch is that the second agent does not know the description of the problem yet and can learn about it only by communicating with the first agent. We then ask how much communication is required for them to cooperatively solve the problem. Surely, the first agent can send the whole problem description so that the second agent can determine the answer. We find many problems for which this simple protocol is, in a complexity-theoretically precise sense, likely to be optimal: It is unlikely that they are able to solve the problem by using significantly less communication.

3.1. ORs of NP-hard problems

All known kernel lower bounds for fixed-parameter tractable problems use – implicitly or explicitly – the fact that compressing ORs of NP-hard problems is impossible under standard complexity-theoretic assumptions. This fact is captured by Lemma 1.1, which we generalize to oracle communication protocols in this section.

Recall that $\text{OR}^t(L)$ is the problem of deciding whether at least one out of $t(s)$ inputs, each of length s , belongs to L . The following lemma shows that low-cost protocols for $\text{OR}^t(L)$ can be used to build a proof system with advice for \bar{L} .

Lemma 3.1 (Complementary Witness Lemma). *Let L be a language and $t : \mathbb{N} \rightarrow \mathbb{N} \setminus \{0\}$ be polynomially bounded. If $\text{OR}^t(L)$ has an oracle communication protocol of cost $O(t(s) \log t(s))$, where the first player can be nondeterministic, then $\bar{L} \in \text{NP/poly}$.*

Lemma 1.1 – the incompressibility of ORs of NP-hard problems under $\text{coNP} \not\subseteq \text{NP/poly}$ – is a corollary of this lemma. To see this, we observe that compressions give rise to low-cost protocols since the first player can just compress its input and then send it to the oracle. We now sketch the proof of Lemma 3.1 in the special case where the language L is P-selective. The simpler argument for that case provides a good starting point for the proof of the general case.

A P-selector for a language L is a polynomial-time algorithm that takes two instances x and y as input and outputs one of them, with the guarantee that if at least one of the inputs belongs to L then so does the output. Note that a P-selector for L immediately yields a low-cost oracle communication protocol for deciding $\text{OR}(L)$ on inputs consisting of t instances of size s each – the first player uses the selector $t - 1$ times to determine

3. Communicating Instances of Hard Problems

which of the instances is “most likely” to be in L , sends that instance to the oracle, who responds with the membership of that instance to L . Since the cost of this protocol is s , any P-selective language satisfies the premise of Lemma 3.1 whenever $t \geq s$.

Ko [Ko83] showed that the existence of a P-selector for L implies that \bar{L} (and thus L) can be decided by circuits of polynomial size. The key insight is the following way to prove that an instance x belongs to \bar{L} : Exhibit an instance y that is known to be in \bar{L} and which the selector S outputs when given x and y as input. We call such a y a complementary witness. By viewing S on all pairs of a given subset $F \subseteq \bar{L}$ as a tournament, there always exists a $y \in F$ that beats at least half of the $x \in F$ and therefore can be used as a proof of membership of x to \bar{L} . Starting from the set of all instances of size s in \bar{L} , we repeatedly apply this procedure to the remaining set F of instances that have not yet been beaten by some of the y 's we picked, until the set becomes empty. This way, we obtain a collection A_s of at most s elements y such that $x \in \bar{L}$ if and only if there exists a $y \in A_s$ such that $S(x, y) = y$. Using the set A_s as advice, this shows that $\bar{L} \in \text{P/poly}$. If we allow the selector S to be nondeterministic (even multivalued), we similarly obtain that $\bar{L} \in \text{NP/poly}$ [HHN⁺95].

Fortnow and Santhanam [FS08] established the case of Lemma 3.1 where the protocol implements a \leq_m^p -reduction from $\text{OR}(L)$ to some language L' such that t -tuples consisting of instances of bitlength s are mapped to an instance of bitlength bounded by some fixed polynomial in s , independent of t . Their proof can be viewed as an extension of the above argument. The witnesses y are now elements from \bar{L}' , and the requirement on the bitlength of the reduced instances guarantees that sufficiently popular y 's exist, so we do not need too many of them. The statement of Lemma 3.1 results from a more careful analysis of that argument for size bounds that can grow slowly with t , and from the extension to the general setting of our oracle communication protocols.

Proof (of Lemma 3.1). Let us first consider the case of a deterministic oracle communication protocol P modeled by a deterministic polynomial-time Turing machine M and a function f (see §2 for the notation). In this proof we make use of the notion of a communication transcript on a given input x . Such a transcript consists of the sequence of all queries P makes on input x (i.e., the contents of M 's oracle query tape at the end of the protocol) as well as the answers $f(q)$ to each of the oracle queries q .

The key ingredient of the proof is the following equivalence: An instance x of length s is in \bar{L} if and only if there exists a sequence $x_2, \dots, x_{t(s)}$ of instances of length s such that $P(x, x_2, \dots, x_{t(s)})$ rejects. Here we can view the sequence $(x, x_2, \dots, x_{t(s)})$ as an unordered sequence since we can assume w.l.o.g. that P sorts the t instances lexicographically before its actual computation starts. By including a large enough set A_s of communication transcripts and the value of $t(s)$ as advice, this leads to the following proof system with advice for \bar{L} . On input an instance x of length s :

1. Guess a sequence $x_2, \dots, x_{t(s)}$ where each x_i has length s .
2. Check whether there is a communication transcript τ in A_s that is consistent with P on input $(x, x_2, \dots, x_{t(s)})$ and that $P(x, x_2, \dots, x_{t(s)})$ rejects. If so, accept; otherwise, reject.

The check for a given transcript τ involves simulating the first player on the input $(x, x_2, \dots, x_{t(s)})$. Whenever the first player sends a bit to the second player (by writing on the oracle query tape), verify that it agrees with the corresponding bit in τ . Whenever the first player expects a bit from the second player (by reading from the oracle answer tape), use the corresponding bit in τ . This process continues until a discrepancy is detected or the first player halts.

This proof system is sound as long as all communication transcripts in A_s are consistent with the protocol P . All that remains to show is the existence of a small subset A_s of such transcripts that guarantees completeness.

We construct the advice set A_s for a fixed s in the following greedy way. Consider instances $x_1, \dots, x_{t(s)}$ of L of length s , and let $T(x_1, \dots, x_{t(s)})$ denote the communication transcript of P on input $(x_1, \dots, x_{t(s)})$. Since the second player is not given the input $(x_1, \dots, x_{t(s)})$, the transcript $T(x_1, \dots, x_{t(s)})$ is determined solely by the bits sent from the first player to the second player. Therefore, the number of distinct such transcripts is less than $2^{c(s)+1}$, where $c(s)$ denotes the cost of the protocol on inputs consisting of $t(s)$ instances of length s each. We say that a rejecting transcript τ covers an instance $x \in \bar{L}$ of length s if there exists a sequence $x_2, \dots, x_{t(s)}$ of instances of length s each such that $T(x, x_2, \dots, x_{t(s)}) = \tau$. We start with A_s empty and successively pick a rejecting communication transcript τ that covers the largest number of instances $x \in \bar{L}$ of length s that are not covered thus far, and add τ to A_s . We keep doing so until there are no more instances $x \in \bar{L}$ of length s left to cover.

Consider one step in the construction of A_s and let F denote the set of uncovered instances $x \in \bar{L}$ of length s at the beginning of the step. Since every tuple in $F^{t(s)}$ is mapped by T to one of the rejecting transcripts above and there are less than $2^{c(s)+1}$ distinct such transcripts, there exists a rejecting transcript τ^* such that at least a fraction $1/2^{c(s)+1}$ of the tuples in $F^{t(s)}$ are mapped by T to this particular τ^* , i.e., $|T^{-1}(\tau^*) \cap F^{t(s)}| \geq |F|^{t(s)}/2^{c(s)+1}$. Now, each component of each tuple in $T^{-1}(\tau^*) \cap F^{t(s)}$ is covered by τ^* since we can regard the tuples as unordered sequences. Thus, if we let G denote the subset of F that is covered by τ^* , we have that $T^{-1}(\tau^*) \cap F^{t(s)} \subseteq G^{t(s)}$. We conclude that

$$|G|^{t(s)} \geq |T^{-1}(\tau^*) \cap F^{t(s)}| \geq |F|^{t(s)}/2^{c(s)+1},$$

whence $|G| \geq \varphi(s) \cdot |F|$ where $\varphi(s) = 1/2^{(c(s)+1)/t(s)}$.

Thus, every step covers a fraction at least $\varphi(s)$ of the remaining instances to be covered. Since there are at most 2^s instances of length s to begin with, after ℓ steps there are no more than $(1 - \varphi(s))^\ell \cdot 2^s \leq \exp(-\varphi(s)\ell) \cdot 2^s$ instances left to cover, so the process ends after $O(s/\varphi(s))$ steps. Now, $1/\varphi(s) = 2^{(c(s)+1)/t(s)}$ is polynomially bounded in $t(s)$ as long as $c(s) = O(t(s) \log t(s))$. Since each transcript as well as the running time of the proof system are polynomially bounded in s and $t(s)$, for polynomially bounded $t(s)$ the resulting algorithm for \bar{L} runs in NP/poly.

This finishes the proof for the case of deterministic protocols P . For conondeterministic protocols we can define $T(x_1, \dots, x_{t(s)})$ to be an arbitrary transcript of an execution on which P produces the correct output. The check in step 2 now involves nondeter-

3. Communicating Instances of Hard Problems

minim. The fact that P has no valid rejecting executions for inputs $(x_1, \dots, x_{t(s)})$ in $\text{OR}(L)$ guarantees the soundness of the proof system, and the existence of at least one valid rejecting execution of P on an input $(x_1, \dots, x_{t(s)})$ outside of $\text{OR}(L)$ guarantees completeness. The counting argument carries over verbatim. ■

3.2. Vertex Cover

In this section we establish Theorem 1.2 – that d -VERTEX COVER has no oracle communication protocol of cost $O(n^{d-\epsilon})$ for any positive constant ϵ unless $\text{coNP} \subseteq \text{NP/poly}$, where n represents the number of vertices of the d -uniform hypergraph. For ease of exposition we actually develop the equivalent result for d -CLIQUE rather than for d -VERTEX COVER. Theorem 1.2 then follows by hypergraph complementation.

We follow the approach outlined in the introduction: We assume that d -CLIQUE has a protocol of low cost and we come up with a suitable reduction from the OR of an NP-hard language L to d -CLIQUE to devise a low-cost protocol for the OR-problem. We then invoke the complementary witness lemma and obtain that $\text{coNP} \subseteq \text{NP/poly}$ if the cost of the assumed protocol for d -CLIQUE was too small. The choice of L does not matter, and we pick it to be 3-SAT for convenience. The following lemma provides us with a reduction that is sufficient to obtain our tight results.

Lemma 3.2. *Let $d \geq 2$ be an integer. There is a \leq_m^p -reduction from $\text{OR}(3\text{-SAT})$ to d -CLIQUE that maps t -tuples of instances of bitlength at most s each to instances with at most $t^{1/d+o(1)} \cdot \text{poly}(s)$ vertices.*

Before we delve into the proof of this lemma, let us first formally derive Theorem 1.2.

Theorem 1.2 (restated). *Let $d \geq 2$ be an integer and ϵ a positive real. If $\text{coNP} \not\subseteq \text{NP/poly}$, there is no protocol of cost $O(n^{d-\epsilon})$ to decide whether a d -uniform hypergraph on n vertices has a vertex cover of at most k vertices, even when the first player is conondeterministic.*

Proof. Suppose d -VERTEX COVER on n -vertex graphs has a protocol of cost $O(n^c)$ for some constant $c < d$. Let L denote 3-SAT. By combining the reduction from Lemma 3.2 with the standard reduction from d -CLIQUE to d -VERTEX COVER (mentioned in the preliminaries) and running the above protocol for d -VERTEX COVER on the result of the combined reduction, we obtain a protocol for $\text{OR}(L)$ of cost $O(n^c) = O((s \cdot \max(s, t^{1/d+o(1)}))^c)$. Since $c < d$ the latter expression is $O(t(s))$ for sufficiently large polynomials $t(s)$. Lemma 3.1 then shows that 3-SAT is in coNP/poly , which is equivalent to $\text{coNP} \subseteq \text{NP/poly}$. ■

For the reduction in Lemma 3.2, we are given t 3-CNF formulas $\varphi_1, \dots, \varphi_t$, and we need to construct a d -uniform hypergraph G on few vertices n and an integer k such that at least one of the φ_i 's is satisfiable if and only if G has a clique of size at least k . We have two independent constructions that achieve an asymptotically optimal bound on the number of vertices n .

Elementary Proof

We present an elementary reduction from $\text{OR}(3\text{-SAT})$ to $d\text{-CLIQUE}$ whose basic structure reappears in our lower bounds for packing problems in §3.5. In this reduction we actually achieve a bound of $t^{1/d} \cdot \text{poly}(s)$ on the number of vertices, so we do not need to use the additional $o(1)$ -slack in the exponent of t .

Proof (of Lemma 3.2). Let $\varphi_1, \dots, \varphi_t$ be the t instances of 3-SAT. Without loss of generality, assume that each formula has exactly $p = s$ clauses, each consisting of a sequence of 3 literals. Let P and K_1, \dots, K_t be the hypergraphs provided by Lemma 3.3. Along the lines of the standard reduction from 3-SAT to 2-CLIQUE by Karp [Kar72], we first translate the 3-CNF formulas φ_i into d -uniform hypergraphs G_i on the vertex sets $V(K_i) \times [3]$. For each i , we identify the elements of $V(K_i) \times [3]$ with (positions of) literals of φ_i : The first component selects a clause from φ_i and the second component selects a literal from the clause. We let G_i be the d -uniform hypergraph with as edges all subsets $e \subseteq V(K_i) \times [3]$ of size d such that no two elements of e correspond to the same clause φ_i or represent complementary literals. Note that each such e induces a satisfying assignment of the conjunction of the d clauses touched by e , and that G_i has a clique of size s if and only if φ_i is satisfiable.

Let G be the union of the G_i 's, that is, the graph with $V(G) = \bigcup_{i \in [t]} V(G_i) \subseteq V(P) \times [3]$ and $E(G) = \bigcup_{i \in [t]} E(G_i)$. If φ_i has a satisfying assignment, then G_i has a clique of size s and so has G . For the other direction, let K be a clique of size s in G . The projection K' of K onto the first component is a clique of size s in P . By property (ii) of Lemma 3.3, $K' = K_i$ for some $i \in [t]$. Moreover, by property (i) of Lemma 3.3, the projections of $E(G_i)$ and $E(G_j)$ for $j \neq i$ are disjoint. It follows that K is a clique of size s in G_i , and therefore φ_i is satisfiable.

Thus, $(G, s) \in d\text{-CLIQUE}$ if and only if $(\varphi_1, \dots, \varphi_t) \in \text{OR}(3\text{-SAT})$. Since G and s are computable in time polynomial in the bitlength of $(\varphi_1, \dots, \varphi_t)$ and $|V(G)| \leq 3|V(P)| \leq O(s \cdot \max(s, t^{1/d+o(1)}))$, we have established the \leq_m^p -reductions claimed in Lemma 3.2. ■

Proof based on AP-free sets

This section contains the original proof of Lemma 3.2 that appeared in [DvM10]. It is based on high-density subsets of the integers that do not contain arithmetic progressions of length three.

The reduction works by first applying a standard translation of the t individual 3-SAT-instances $\varphi_1, \dots, \varphi_t$, say of size s , into equivalent $d\text{-CLIQUE}$ -instances consisting of d -uniform hypergraphs G_1, \dots, G_t on $3s$ vertices each, such that G_i has a clique of size s if and only if φ_i is satisfiable. All that is left then is to turn these t instances into a single instance of $d\text{-CLIQUE}$ which is positive if and only if at least one of the t instances is. If we take G as the disjoint union of the G_i 's, then G is a d -uniform hypergraph that has a clique of size s if and only if at least one of the G_i 's has a clique of size s . However, this G contains $n = 3s \cdot t$ vertices, which is too many for our purposes. In order to do better, we need to pack the graphs G_i more tightly while maintaining the properties required of the reduction. The following almost-optimal packing of cliques is

3. Communicating Instances of Hard Problems

the critical ingredient in the construction and allows us to achieve the almost-optimal lower bounds given in Theorem 1.2.

Lemma 3.3 (Packing Lemma). *For any integers $p \geq d \geq 2$ and $t > 0$ there exists an p -partite d -uniform hypergraph P on $O(p \cdot \max(p, t^{1/d+o(1)}))$ vertices such that*

- (i) *the hyperedges of P partition into t cliques K_1, \dots, K_t on p vertices each, and*
- (ii) *P contains no cliques on p vertices other than the K_i 's.*

Furthermore, for any fixed d , the hypergraph P and the K_i 's can be constructed in time polynomial in p and t .

Condition (i) in Lemma 3.3 formalizes the notion of a packing. The part that P contains the t cliques K_i ensures the completeness of the reduction, i.e., that G has a clique of size $p := s$ if at least one of the G_i 's does. The part that the K_i 's are edge-disjoint and condition (ii) guarantee the soundness of the reduction, i.e., that G has a clique of size s only if at least one of the G_i 's does.

We defer the proof of Lemma 3.3 to §4. Using it as sketched above we obtain an alternative reduction for Lemma 3.2.

Proof (of Lemma 3.2). Let $\varphi_1, \dots, \varphi_t$ be the t instances of 3-SAT. Without loss of generality, assume that each formula has exactly $p = s$ clauses, each consisting of a sequence of 3 literals. Let P and K_1, \dots, K_t be the hypergraphs provided by Lemma 3.3. Along the lines of the standard reduction from 3-SAT to 2-CLIQUE by Karp [Kar72], we first translate the 3-CNF formulas φ_i into d -uniform hypergraphs G_i on the vertex sets $V(K_i) \times [3]$. For each i , we identify the elements of $V(K_i) \times [3]$ with (positions of) literals of φ_i : The first component selects a clause from φ_i and the second component selects a literal from the clause. We let G_i be the d -uniform hypergraph with as edges all subsets $e \subseteq V(K_i) \times [3]$ of size d such that no two elements of e correspond to the same clause φ_i or represent complementary literals. Note that each such e induces a satisfying assignment of the conjunction of the d clauses touched by e , and that G_i has a clique of size s if and only if φ_i is satisfiable.

Let G be the union of the G_i 's, that is, the graph with $V(G) = \bigcup_{i \in [t]} V(G_i) \subseteq V(P) \times [3]$ and $E(G) = \bigcup_{i \in [t]} E(G_i)$. If φ_i has a satisfying assignment, then G_i has a clique of size s and so has G . For the other direction, let K be a clique of size s in G . The projection K' of K onto the first component is a clique of size s in P . By property (ii) of Lemma 3.3, $K' = K_i$ for some $i \in [t]$. Moreover, by property (i) of Lemma 3.3, the projections of $E(G_i)$ and $E(G_j)$ for $j \neq i$ are disjoint. It follows that K is a clique of size s in G_i , and therefore φ_i is satisfiable.

Thus, $(G, s) \in d$ -CLIQUE if and only if $(\varphi_1, \dots, \varphi_t) \in \text{OR}(3\text{-SAT})$. Since G and s are computable in time polynomial in the bitlength of $(\varphi_1, \dots, \varphi_t)$ and $|V(G)| \leq 3|V(P)| \leq O(s \cdot \max(s, t^{1/d+o(1)}))$, we have established the \leq_m^p -reductions claimed in Lemma 3.2. ■

3.3. Satisfiability

Theorem 1.1, our tight oracle communication lower bound for d -SAT parameterized by the number of variables of the formula, immediately follows from Theorem 1.2 and the next lemma.

Lemma 3.4. *For every $d \geq 3$, there is a \leq_m^p -reduction from d -VERTEX COVER to d -SAT that maps d -uniform hypergraphs on n vertices to d -CNF formulas on $O(n)$ variables.*

Proof. Let (G, k) be an n -vertex instance of d -VERTEX COVER. The following d -CNF formula on variables x_v for $v \in V(G)$ has as satisfying assignments precisely the characteristic vectors of vertex covers of G :

$$\varphi := \bigwedge_{e \in E(G)} \bigvee_{v \in e} x_v.$$

Using at most $O(n)$ new variables, we construct a 3-CNF formula ψ that is satisfied by all assignments in which at most k distinct x_v are set to true. Then $\varphi \wedge \psi$ is satisfiable if and only if G has a vertex cover of size at most k .

For the construction of ψ , we use a Boolean circuit of constant fan-in that has at most $O(n)$ gates and checks whether at most k of the n input variables are set to true. Such circuits can be constructed for any symmetric function in time polynomial in n when given oracle access to the function [Weg87, Theorem 4.1]. Once we have that circuit, we construct ψ in a standard way by introducing a new variable for each gate, and letting ψ be the conjunction of clauses that express the correct behavior of each of the gates, and the clause stipulating that the output gate is set. ■

Proof (of Theorem 1.1). Suppose there exists an oracle communication protocol of cost $O(n^{d-\epsilon})$ for n -variable instances of d -SAT. We combine the reduction from Lemma 3.4 with the former and obtain an protocol of cost $O(n^{d-\epsilon})$ for n -vertex instances of d -VERTEX COVER. By Theorem 1.2, this implies that $\text{coNP} \subseteq \text{NP/poly}$. ■

The following corollary to Theorem 1.1 embodies the consequences for sparsification, kernelization, and lossy compression.

Corollary 3.1. *Let $d \geq 3$ be an integer. If $\text{coNP} \not\subseteq \text{NP/poly}$, there is no polynomial-time reduction from d -SAT to any problem that makes at most $O(n^b)$ queries and only queries strings of bitlength $O(n^c)$, where b and c are any nonnegative reals with $b+c < d$.*

In particular, under the hypothesis that $\text{coNP} \not\subseteq \text{NP/poly}$, Corollary 3.1 implies that \leq_m^p -reductions cannot reduce the density of n -variable d -SAT instances to $O(n^c)$ clauses for any constant c below the trivial $c = d$. This is in contrast to the subexponential-time level: The sparsification lemma of [IPZ01] gives a reduction which, on input an n -variable d -CNF formula and a rational $\epsilon > 0$, runs in time $2^{\epsilon n} \cdot \text{poly}(n)$ and makes $2^{\epsilon n}$ nonadaptive queries, each of which are d -CNF formulas with at most $f(d, \epsilon) \cdot n$ clauses. The best known bound on the sparsification constant $f(d, \epsilon)$ is $(d/\epsilon)^{3d}$ [CIP06]. The sparsification

3. Communicating Instances of Hard Problems

lemma implies that sparse instances of d -SAT are hard under subexponential-time reductions while Corollary 3.1 suggests that such a result is impossible under \leq_m^p -reductions. Interpretations of Corollary 3.1 in terms of kernelization and lossy compression follow along the same lines.

Another consequence of Theorem 1.1 deals with the size of probabilistically checkable proofs for satisfiability. Recall that Dinur [Din07] constructed such PCPs of size $O(s \cdot \text{poly log } s)$, where s denotes the bitlength of the formula. Based on a connection due to Harnik and Naor [HN06] between PCPs and lossy compression, Fortnow and Santhanam [FS08] showed that satisfiability of Boolean formulas does not have PCPs of size bounded by a polynomial in the number of variables only, unless $\text{coNP} \subseteq \text{NP/poly}$. Plugging in our lower bound for d -SAT into their argument shows that d -SAT does not have q -query PCPs of size $O(n^{d/q-\epsilon})$ unless $\text{coNP} \subseteq \text{NP/poly}$. Since $q \geq 3$ this bound is not tight. Using a different argument and exploiting the fact that Theorem 1.1 also holds for conondeterministic protocols, we can close the gap between the upper and lower bound.

Corollary 3.2. *Let $d \geq 3$ be an integer and ϵ a positive real. If $\text{coNP} \not\subseteq \text{NP/poly}$, then d -SAT does not have probabilistically checkable proofs of bitlength $O(n^{d-\epsilon})$ where n denotes the number of variables of the input formula.*

Proof. Suppose that d -SAT has PCPs of size $s = O(n^c)$ that make q nonadaptive queries, where c and q are constants. We claim that this implies a conondeterministic multi-valued mapping reduction from d -SAT to q -SAT that maps formulas on n variables to instances of bitlength $O(n^c \log n)$ in the following sense: There exists a nondeterministic polynomial-time Turing machine M which outputs a q -CNF formula on each computation path (where the formula may depend on the input and the computation path) such that (i) if the input is in d -SAT then every output is in q -SAT, and (ii) otherwise at least one output is not in q -SAT. For $c < d$, Theorem 1.1 then shows that $\text{coNP} \subseteq \text{NP/poly}$.

All that remains is to argue the claim. For a given formula φ on n variables, introduce s new variables y , namely one for each bit position in a candidate PCP of size s . If the PCP system reads at most q bits of the proof, each condition the PCP system checks can be expressed efficiently as a q -CNF. By picking a condition according to the distribution of the PCP system and a clause of the corresponding q -CNF formula uniformly at random, we obtain a polynomial-time randomized procedure that produces a q -clause on the variables y with the property that if φ is satisfiable, then all q -clauses produced are simultaneously satisfiable, and otherwise less than a constant fraction $\rho < 1$ is. By averaging, the latter implies that for every collection of candidate PCPs of size s for an unsatisfiable input φ , there exists a produced q -clause that is violated by more than a fraction $1 - \rho$ of the collection. Since there are 2^s candidate PCPs of size s in total, this means that there is a set of $s/\log(1/\rho)$ produced q -clauses that cannot be satisfied by any PCP of size s . The reduction nondeterministically guesses $s/\log(1/\rho)$ many q -clauses that are produced by the PCP system on input φ , and outputs their conjunction. The conjunction has bitlength $O(n^c \log n)$, is always satisfiable if φ is, and is not satisfiable on at least one computation path otherwise. ■

It is straightforward to obtain applications similar to Corollaries 3.1 and 3.2 for d -VERTEX COVER and other problems.

3.4. Covering Problems

Combinatorial covering problems are problems in which we are given a graph and a pattern, and we want to find few vertices that cover each occurrence of the pattern in the graph. Put differently, we want to know whether it is possible to delete k vertices from the graph such that no copies of the respective pattern remain. Clearly d -VERTEX COVER is of that kind, since we want to find k vertices whose removal leaves a graph without edges. A natural parameter for d -VERTEX COVER and covering problems in general is the size k of the deletion set. We investigate the consequences of Theorem 1.2 for this parameterization of covering problems, first for the case $d = 2$, i.e., for standard graphs, and then for d -uniform hypergraphs for general d .

Result for Standard Graphs

We consider the following generalization of the vertex cover problem. Recall that a graph property is a predicate on graphs that is invariant under graph isomorphism.

Definition 3.1 (Vertex Deletion). *Fix a graph property Π . The Π -VERTEX DELETION problem is to decide, for a given graph G and integer k , whether there exists a subset S of at most k vertices such that $G \setminus S$ satisfies Π .*

We say that a graph property Π is inherited by subgraphs if, whenever a graph G satisfies Π , every subgraph of G also satisfies Π . The following natural graph problems are special cases of Π -VERTEX DELETION for a graph property Π that is inherited by subgraphs.

- VERTEX COVER: Can we delete k vertices to destroy all edges?
- FEEDBACK VERTEX SET: Can we delete k vertices to destroy all cycles?
- BOUNDED-DEGREE DELETION: Can we delete k vertices to get a maximum degree of d ?
- NON-PLANAR DELETION: Can we delete k vertices to make the graph planar?
- Can we delete k vertices to make the graph embeddable into some surface?
- Can we delete k vertices to make the graph exclude any fixed set of minors?

As mentioned in the introduction, if only finitely many graphs satisfy Π or if all graphs satisfy Π , Π -VERTEX DELETION is trivially decidable in polynomial time. For all other graph properties Π that are inherited by subgraphs, Theorem 1.3 implies that Π -VERTEX DELETION does not have kernels with $O(k^{2-\epsilon})$ edges unless $\text{coNP} \subseteq \text{NP/poly}$.

3. Communicating Instances of Hard Problems

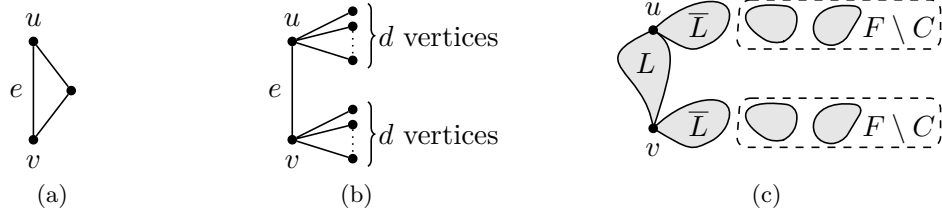


Figure 3.1.: Replacement of an edge $e = \{u, v\}$ in the transformation from G to G' in the proof of Lemma 3.5. (a) FEEDBACK VERTEX SET. (b) BOUNDED-DEGREE DELETION. (c) The general case.

We now prove Theorem 1.3 by constructing a \leq_m^p -reduction from VERTEX COVER to Π -VERTEX DELETION that blows up the size of the deletion set by no more than a constant factor. In order to develop some intuition, we first consider the standard reduction from VERTEX COVER to FEEDBACK VERTEX SET [Kar72]. The reduction replaces every edge e of a VERTEX COVER-instance G by a cycle of length three using an additional new vertex, as depicted in Figure 3.1a. Let us denote the resulting graph by G' . Since every cycle in G' contains two vertices that are adjacent in G , every vertex cover of G hits every cycle of G' and therefore is a feedback vertex set of G' . Conversely, every feedback vertex set of G' contains a vertex of every triangle we created, and can therefore be turned into a vertex cover of G of at most the same size. Thus, G has a vertex cover of size k if and only if G' has a feedback vertex set of size k .

As another example, consider the case of BOUNDED-DEGREE DELETION. In the known reduction from VERTEX COVER to this problem [KD79], d new edges are attached to every vertex of G (see Figure 3.1b). Removing any vertex cover of G from G' reduces the maximum degree to d . Vice versa, any set that reduces the maximum degree in G' to d can be transformed into a vertex cover of G of at most the same size.

Next consider the more general case in which the minimal graphs that violate Π are connected. Generalizing the above two examples we obtain G' by replacing every edge of the VERTEX COVER-instance G by a copy of a fixed connected graph F violating Π . We refer to F as a “forbidden” graph since no graph satisfying Π can contain F as a subgraph. Thus, any deletion set in G' has to pick at least one vertex from every copy of F . Projecting the deletion set back onto the graph G yields a vertex cover of size no more than the deletion set. This way we can guarantee the soundness of the reduction – if G' has a deletion set of size at most k then G has a vertex cover of size at most k .

For the completeness of the reduction, we would like to ensure that removing a vertex cover S of G from G' leaves a graph $G' \setminus S$ satisfying Π . This is not automatically the case because $G' \setminus S$ may contain components of the form depicted in Figure 3.2a, where the bullets are vertices of G and the hashed vertices are part of the vertex cover S (and are therefore not part of $G' \setminus S$) but the center vertex is not. Such a component could contain a copy of F , in which case $G' \setminus S$ would not satisfy Π . However, by attaching the copies of F in an appropriate way we can make sure that the connected components

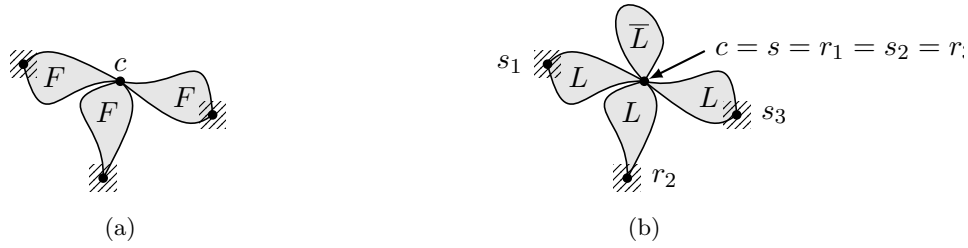


Figure 3.2.: Connected component C' that might remain after removing a vertex cover S of G from G' , centered around a vertex c that has degree 3 in G and does not belong to S . (a) Naïve construction. (b) Final construction.

of $G' \setminus S$ are all “simpler” than F . Picking F to be a “simplest” connected graph that violates Π then does the job as long as all minimal graphs violating Π are connected.

More generally, consider a graph F violating Π whose most complex connected component C is as simple as possible among all graphs violating Π . If F has no other connected component of the same complexity as C , then the above construction still works, using a copy of C to replace every edge in G and including a copy of $F \setminus C$ for every vertex of G .

In the most general case, where minimal graphs violating Π can have multiple components of the same complexity, we use a slightly different construction that involves multiple copies of G . The graph F now becomes a “simplest” graph for which the number of disjoint copies of F that satisfies Π is bounded. The reduction is no longer parameter preserving in general, but the parameter k' for G' is still linearly bounded by the parameter k for G . The latter ensures that the lower bound for Π -VERTEX DELETION is as strong as for VERTEX COVER modulo a constant factor.

The simplicity measure we use is the same as the one of Lewis and Yannakakis [LY80] but the construction is a bit different: their construction blows up the parameter k' to $\Theta(nk)$, but a straightforward modification reduces k' to $\Theta(k^2)$. We further reduce k' to $\Theta(k)$ using a matching argument.

Lemma 3.5. *Let Π be a graph property that is inherited by subgraphs, and is satisfied by infinitely many but not all graphs. There is a \leq_m^p -reduction from VERTEX COVER to Π -VERTEX DELETION that maps instances with parameter k to instances with parameter $O(k)$.*

Proof. We start by spelling out the simplicity measure for graphs. We first consider a connected graph C . For any vertex s of C , we define the character of C relative to s as the sequence $\chi = (\chi_i)_{i \in \mathbb{N}}$ where χ_i denotes the number of connected components of $C \setminus \{s\}$ that have exactly i vertices. We compare two characters χ and η using the colexicographical order, i.e., $\chi < \eta$ if there exists a positive integer i such that $\chi_j = \eta_j$ for all integers $j > i$ and $\chi_i < \eta_i$. The corresponding relation \leq defines a well-order on the set of characters, that is, a total order in which every nonempty subset has a

3. Communicating Instances of Hard Problems

smallest element. We define the *character* of C as a smallest character of C relative to s over all vertices s of C .

For an arbitrary graph G we define its *signature* as a mapping σ from the set of all characters to \mathbb{N} , where $\sigma(\chi)$ equals the number of connected components of G with character χ . We compare two signatures σ and τ using the colexicographical order induced by the order on characters, i.e., $\sigma < \tau$ if there exists a character χ such that $\sigma(\eta) = \tau(\eta)$ for all characters $\eta > \chi$ and $\sigma(\chi) < \tau(\chi)$. The corresponding relation \leq defines a well-order on the set of signatures.

Our simplicity measure on graphs is induced by the \leq -relation on their signatures. We choose a graph F with the smallest signature among all graphs for which the number of disjoint copies that satisfy Π is bounded. Note that F exists because not all graphs satisfy Π . Let t be the positive integer such that the disjoint union of $t - 1$ copies of F satisfies Π but t disjoint copies do not. Let C denote a connected component of F with largest character and let $s \in V(C)$ be a witness for that character. Let L be the subgraph of C spanned by s and the vertices of a largest connected component of $C \setminus \{s\}$, and let \bar{L} be the subgraph of C spanned by s and the vertices of $C \setminus L$. Note that L contains at least one other vertex than s . Otherwise, F would consist of isolated vertices only and only finitely many graphs would satisfy Π . Let r be an arbitrary vertex of $L \setminus \{s\}$.

We are now in position to describe the reduction transforming an instance (G, k) of VERTEX COVER into an instance (G', k') of Π -VERTEX DELETION such that G has a vertex cover of size k if and only if k' vertices can be deleted from G' to make the residual graph satisfy Π . For the construction of G' we start with $2t - 1$ disjoint copies G_1, \dots, G_{2t-1} of G . We replace every edge e of G_i by a copy L_e of the component L such that the endpoints of e are identified with s and r in an arbitrary way; the vertices of L_e outside of e are new. Furthermore, we attach to every vertex $v \in V(G)$ a graph R_v that consists of a copy of \bar{L} and disjoint copies of $F \setminus C$; here we identify v with the vertex s of \bar{L} and create all other vertices of R_v new. See Figure 3.1c. In the remainder, we show that the reduction works when we set $k' = (2t - 1)k$.

For the soundness of the reduction, let S' be a set of k' vertices in G' such that $G' \setminus S'$ satisfies Π . Let S denote the projection of S' onto $V(G_1) \cup \dots \cup V(G_{2t-1})$, where the projection of a vertex $u \in V(G')$ is one of the vertices of e (chosen arbitrarily) in case $u \in V(L_e) \setminus e$ and the vertex v in case $u \in V(R_v)$. We claim that S is at most $2t - 2$ vertices away from being a vertex cover of $G_1 \cup \dots \cup G_{2t-1}$. Let M be a maximal matching in $(G_1 \cup \dots \cup G_{2t-1}) \setminus S$. If M contains at least t edges, then S' avoids at least t disjoint subgraphs $L_e \cup R_u \cup R_v$ for $e = (u, v)$. In particular, $G' \setminus S'$ contains t copies of F as subgraphs, which contradicts the fact that $G' \setminus S'$ satisfies Π . Thus, M contains at most $t - 1$ edges. Adding $V(M)$ to S , we thus get a vertex cover of $G_1 \cup \dots \cup G_{2t-1}$ of size at most $(2t - 1)k + 2t - 2$. By averaging, there is an i with $|S \cap V(G_i)| \leq \lfloor k + 1 - \frac{1}{2t-1} \rfloor = k$. Hence G has a vertex cover of size at most k .

For the completeness of the reduction, let S be a vertex cover of G of size at most k . Let S' consist of the $2t - 1$ copies of S in the graphs G_1, \dots, G_{2t-1} . Clearly, $|S'| \leq (2t - 1)k$. Let H be obtained from $G' \setminus S'$ by removing duplicate isomorphic copies of connected components. Note that $G' \setminus S'$ is a subgraph of finitely many disjoint copies

of H . Thus, if we can show that H has a strictly smaller signature than F , then any number of disjoint copies of H satisfies Π and by inheritance the subgraph $G' \setminus S'$ also satisfies Π . Therefore, S' is a set of at most $k' = (2t - 1)k$ vertices such that $G' \setminus S'$ satisfies Π .

It remains to argue that H has a strictly smaller signature than F . In order to do so we consider the connected components of H , and we distinguish four types: (1) components isomorphic to components of $F \setminus C$, (2) components isomorphic to components of $L \setminus \{s, r\}$, (3) components isomorphic to components of $\bar{L} \setminus \{s\}$, and (4) components as in Figure 3.2b consisting of a single copy of \bar{L} and one or more copies of $L \setminus \{s\}$ and $L \setminus \{r\}$ in which all remaining copies of s and r have been identified with the vertex c . We show that for each of the connected components of types (2), (3), and (4), the character is strictly less than for C . Since C is the connected component of F with the largest character and H has no duplicate isomorphic connected components, this implies that no connected component of H has a character larger than C , and that the number of connected components of H with the same character as C is strictly less than in F . Therefore, the signature of H is strictly less than the one of F .

Let us first consider a connected component C' of H of type (4). Consider removing the vertex c in Figure 3.2b. Since $L \setminus \{s\}$ is a largest connected component of $C \setminus \{s\}$, no connected component of $C' \setminus \{c\}$ can have more vertices than $L \setminus \{s\}$. Moreover, the only components in $C' \setminus \{c\}$ that can have $|V(L \setminus \{s\})|$ vertices must come from the part $\bar{L} \setminus \{s\}$. Since $C = L \cup \bar{L}$, this means that $C \setminus \{s\}$ has one more connected component with $|V(L \setminus \{s\})|$ vertices than $C' \setminus \{c\}$. Thus, the character of C' relative to c , and a fortiori the character of C' , is strictly less than the character of C .

The claim that connected components of types (2) and (3) have characters strictly less than C follows from the corresponding claim for type (4) since (2) and (3) are subgraphs of a graph of type (4) and taking subgraphs cannot result in larger characters. ■

We point out that the proof of Lewis and Yannakakis [LY80] only needs inheritance by *induced* subgraphs. The only step in the proof of Lemma 3.5 that requires the stronger property of inheritance by subgraphs is the matching argument. That step is vacuous when $t = 1$, e.g., when all minimal graphs violating Π are connected. The stronger property is also not necessary when the vertex s is not adjacent to all vertices of L (and we choose r as one of the non-adjacent vertices). In such cases our proof can do with inheritance by induced subgraphs.

Proof (of Theorem 1.3). Suppose that Π -VERTEX DELETION parameterized by the size of the deletion set has a cost $O(k^{2-\epsilon})$ protocol. By combining the \leq_m^p -reduction from Lemma 3.5 with that protocol, we obtain a cost $O(k^{2-\epsilon})$ protocol for VERTEX COVER parameterized by the size of the vertex cover. Since $k \leq n$, the case $d = 2$ of Theorem 1.2 then implies that $\text{coNP} \subseteq \text{NP/poly}$. ■

Theorem 1.3 applies, among others, to FEEDBACK VERTEX SET, another problem whose kernelization has received considerable attention in parameterized complexity. Theorem 1.3 implies that FEEDBACK VERTEX SET does not have kernels consisting of $O(k^{2-\epsilon})$ edges unless $\text{coNP} \subseteq \text{NP/poly}$. This result is tight – a kernel with $O(k^2)$ edges

3. Communicating Instances of Hard Problems

follows from recent work by Thomassé [Tho09]. He constructs a kernel with at most $4k^2$ vertices and maximum degree at most $4k$. For such an instance to be positive, the number of edges can be no larger than $8k^2$. Indeed, suppose that S is a feedback vertex set of G of size at most k . Then the graph induced by $V(G) \setminus S$ is a forest and has at most $4k^2$ edges. All other edges of G are incident to a vertex of S . As the maximum degree is no larger than $4k$, at most $4k^2$ edges are incident to S . Summing up, G has at most $8k^2$ edges. Thus, if G has more than $8k^2$ edges, we can reduce to a trivial negative instance; otherwise, we reduce to G . This results in a kernel with $O(k^2)$ edges.

Extension to Hypergraphs

We now turn to vertex cover and related problems on d -uniform hypergraphs. Since $k \leq n$, Theorem 1.2 implies that d -VERTEX COVER does not have kernels with $O(k^{d-\epsilon})$ edges unless $\text{coNP} \subseteq \text{NP/poly}$. We point out that kernels with $O(k^d)$ edges exist for d -VERTEX COVER. This follows from a generalization of Buss' high-degree rule (see the introduction) and a folklore application of the sunflower lemma (see [FG06, chapter 9.1], for example). Recall that for a hypergraph G , a sunflower with heart $h \subseteq V(G)$ and p petals is a set of distinct edges whose pairwise intersection is exactly h . The kernelization proceeds by repeatedly picking a sunflower with at least $k + 1$ petals, removing the involved edges, and adding the heart as a new edge to the graph. Note that in this process, edges of size less than d may be added to G . To get back a d -uniform graph, one can complete those edges with fresh vertices, which doesn't affect the number of edges nor the minimum size of a vertex cover. The process continues until no sunflower with $k + 1$ petals exists, which is bound to happen as the number of edges decreases in every step. The sunflower lemma of Erdős and Rado [ER60] states that any d -uniform hypergraph with more than $d! \cdot k^d$ edges has a sunflower with $k + 1$ petals. Thus, the hypergraph that remains at the end has at most $d \cdot d! \cdot k^d = O(k^d)$ edges, and has a vertex cover of size at most k if and only if the original hypergraph does.

Regarding extensions of Theorem 1.3 to d -uniform hypergraphs for $d > 2$, we cannot expect to rule out protocols of cost $O(k^{d-\epsilon})$ for all hypergraph properties Π that are inherited by subgraphs and for which the deletion problem is nontrivial. This is because the property Π could only depend on the primal graph underlying the hypergraph, for which protocols of cost $O(k^2)$ are known in some cases.

3.5. Packing Problems

Kernelization of the Set Matching Problem

The d -SET MATCHING problem is to find a maximum collection of hyperedges in a d -uniform hypergraph such that any two hyperedges are disjoint. For $d = 2$, this is the maximum matching problem and polynomial-time solvable. The restriction of this problem to d -partite hypergraphs is the d -dimensional matching problem and NP-hard [Kar72] for $d \geq 3$.

We use Lemma 1.1 to prove that the kernel size of $O(k^d)$ in Theorem 1.4 is asymptotically optimal under the hypothesis $\text{coNP} \not\subseteq \text{NP/poly}$. For the reduction, we use gadgets with few vertices that coordinate the availability of groups of vertices. For example, we may have two sets U_1, U_2 of vertices and our gadget makes sure that in every perfect packing of the graph one set is fully covered by the gadget while the other group has to be covered by hyperedges of the graph external to the gadget. Ultimately, this enables us to choose between different instances in the OR-problem. The precise formulation of the gadget is as follows.

Lemma 3.6. *Let $d \geq 3$, $m \geq 1$, and $s \geq 1$ be integers. In time polynomial in d, m, s , we can compute a d -uniform hypergraph S with $O(dsm)$ vertices and pairwise disjoint sets $U_1(S), \dots, U_m(S) \subset V(S)$ of size s each, such that the following conditions hold.*

- (i) (Completeness) *For each i , $S - U_i$ has a perfect matching.*
- (ii) (Soundness) *If S is a subgraph of some G and the vertices of $S - (U_1 \cup \dots \cup U_m)$ are only contained in edges of S , then every perfect matching of G contains a perfect matching of $S - U_i$ for some i .*
- (iii) *The underlying graph of S (the graph obtained by replacing the d -hyperedges of S by d -cliques) does not contain a clique of size $d + 1$ and it contains $\bigcup_i U_i$ as an independent set.*

In addition to the completeness and the soundness properties that make the gadget work the way we want, we also have a structural property (iii), which we need later when we transfer our results to K_d -MATCHING. We defer the proof of Lemma 3.6 to the end of this section and use it now to prove the following.

Lemma 3.7. *Let $d \geq 3$ be an integer.*

There is a \leq_m^p -reduction from OR(d -SET MATCHING) to d -SET MATCHING that maps t -tuples of instances of bitlength s each to instances on $t^{1/d} \cdot \text{poly}(s)$ vertices whose underlying graph does not contain a clique of size $d + 1$.

Proof. Let G_1, \dots, G_t be instances of d -SET MATCHING, i.e., d -uniform hypergraphs of size s each. Finding perfect matchings in d -partite d -uniform hypergraphs is NP-hard for $d \geq 3$, so we can assume w.l.o.g. that the G_i 's are d -partite and each part of the partition contains exactly s/d vertices. The goal is to find out whether some G_i contains a perfect matching. We reduce this question to an instance G on few vertices.

The vertex set of G consists of $d \cdot t^{1/d}$ groups of n/d vertices each, i.e., $V(G) = \bigcup_{a,b} V_{a,b}$ for $a \in [d]$ and $b \in [t^{1/d}]$. Then we can write the input graphs as G_b using an index vector $b = (b_1, \dots, b_d) \in [t^{1/d}]^d$. For each graph G_b we add edges to G in the following way: We identify the vertex set of G_b with $V_{1,b_1} \dot{\cup} \dots \dot{\cup} V_{d,b_d}$, and we let G contain all the edges of G_b . Since each G_b is d -partite, the same is true for G at this stage of the construction. Now we modify G such that each perfect matching of G only ever uses edges originating from at most one graph G_b . For this it suffices to add a gadget for every $a \in [d]$ that blocks all but exactly one group $V_{a,b}$ in every perfect matching. For each

3. Communicating Instances of Hard Problems

$a \in [d]$, we add a copy S_a of $S(V_{a,1}, \dots, V_{a,m})$ from Lemma 3.6 to G , where $m = t^{1/d}$. Clearly, $|V(G)| \leq O(st^{1/d})$. Furthermore, the underlying graph of G does not contain a clique of size $d + 1$ as the graph restricted to $\bigcup_{a,b} V_{a,b}$ is d -partite and the gadgets do not contain cliques of size $d + 1$ in their underlying graph.

Now we verify the correctness of the reduction. If some G_b has a perfect matching then the completeness property of S_a ensures that $S_a - V_{a,b_a}$ has a perfect matching for all $a \in [d]$. Together with the perfect matching of G_b this gives a perfect matching of G . For the soundness, assume M is a perfect matching of G . Then each S_a is guaranteed to have a b_a such that M contains a perfect matching of $S_a - V_{a,b_a}$. Since V_{a,b_a} is an independent set in S_a , M uses only edges of G_b to cover the V_{a,b_a} . In particular, G_b has a perfect matching. ■

Our kernel lower bound for d -SET MATCHING, now follows immediately by combining the above with Lemma 1.1.

Theorem 1.5 (restated). *Let $d \geq 3$ be an integer and ϵ a positive real. If $\text{coNP} \not\subseteq \text{NP/poly}$, there is no protocol of cost $O(k^{d-\epsilon})$ for PERFECT d -SET MATCHING.*

Proof of Lemma 3.6

We use cycles as building blocks in the gadget constructions. A *loose cycle of length ℓ* in a d -uniform hypergraph is a sequence $C = v_1, e_1, v_2, e_2, \dots, v_\ell, e_\ell$ with the property that $e_i \cap e_{i+1} = \{v_{i+1}\}$ and $e_i \cap e_j = \emptyset$ if $i \notin \{j-1, j, j+1\}$. The indices are always understood modulo ℓ . The vertices v_1, \dots, v_ℓ are the *connection* vertices, whereas all other vertices are *free* vertices of the cycle. Our first lemma, which allows us to coordinate two sets of vertices.

Lemma 3.8. *Let $d \geq 3$ and $s \geq 1$ be integers. Let $C = v_1, e_1, v_2, e_2, \dots, v_{2s}, e_{2s}$ be a loose cycle of d -hyperedges as depicted in Figure 3.3. We define $U_1(C) = \bigcup_{i \text{ even}} e_i \setminus \{v_i, v_{i+1}\}$ and $U_2(C) = \bigcup_{i \text{ odd}} e_i \setminus \{v_i, v_{i+1}\}$. Then*

- (i) (Completeness) $C - U_1$ and $C - U_2$ have a perfect matching.
- (ii) (Soundness) If C is a subgraph of some G and the vertices of $C - (U_1 \cup U_2)$ are only contained in edges of C , then every perfect matching of G contains a perfect matching of $C - U_i$ for some i .

Proof. For the completeness, $\{e_{2i+1}\}$ forms a perfect matching of $C - U_1$ and $\{e_{2i}\}$ forms a perfect matching of $C - U_2$. For the soundness, the only way to cover a vertex v_i of C is to pick one of its two incident hyperedges. Since C is an even cycle, the two ways of doing this for all such vertices in a consistent way are as in the completeness step. ■

We use the above gadget with two choices to construct the gadget in Lemma 3.6, which forces perfect matchings to choose properly between m sets of vertices.

Proof (of Lemma 3.6). We construct a *coordination gadget* S as depicted in Figure 3.4 as follows:

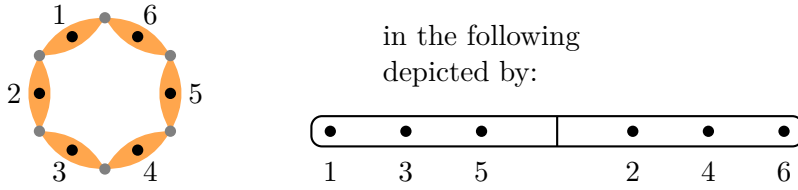


Figure 3.3.: *Left*: An even cycle gadget with $d = 3$, $s = 3$, $U_1 = \{1, 3, 5\}$, and $U_2 = \{2, 4, 6\}$. Black vertices are free vertices, and gray vertices are connection vertices that are not supposed to be adjacent to any other vertex of the outside graph. *Right*: Pictorial abbreviation of the graph on the left. By Lemma 3.8, any perfect matching blocks exactly the vertices in one of the halves using edges of the gadget.

1. We start with s disjoint odd cycles: loose cycles C_1, \dots, C_s of length $2m + 1$ each. We denote the vertices in these cycles with $c_{i,j}$ and the edges with $C_{i,j}$, i.e.,

$$C_i = c_{i,1}, C_{i,1}, c_{i,2}, C_{i,2}, \dots, c_{i,2m+1}, C_{i,2m+1}.$$

Let $C = \bigcup_{i,j} C_{i,j} \setminus \{c_{i,j}, c_{i,j+1}\}$ be the set of all free vertices in these cycles.

2. We define $U_j(S) = \{c_{1,2j}, \dots, c_{s,2j}\}$ for all $j \in [m]$.
3. We add $2m + 1$ disjoint even cycles: loose cycles F_1, \dots, F_{2m+1} of length $2s$ as in Lemma 3.8. We denote the vertices in these cycles with $f_{j,i}$ and the edges with $F_{j,i}$, i.e.,

$$F_j = f_{j,1}, F_{j,1}, f_{j,2}, F_{j,2}, \dots, f_{j,2s}, F_{j,2s}.$$

We identify $\bigcup_j U_1(F_j)$ and C in such a way that

$$F_{j,2i} \setminus \{f_{j,2i}, f_{j,2i+1}\} = C_{i,j} \setminus \{c_{i,j}, c_{i,j+1}\}. \quad (3.1)$$

Let $F = \bigcup_{j,i} F_{j,i} \setminus \{f_{j,i}, f_{j,i+1}\}$ be the set of all free vertices in the even cycles.

4. For $j \in [2m+1]$, enumerate the vertices $v_{j,1}, \dots, v_{j,(d-2)s}$ of $U_2(F_j) = V(F_j) \cap F \setminus C$ arbitrarily. For each $k \in [(d-2)s]$, add a set A_k containing $m+1$ sets $\{u_1, \dots, u_{d-1}\}$ of $(d-1)$ fresh vertices, and for all j we add the edges $\{u_1, \dots, u_{d-1}, v_{j,k}\}$ to S .

This finishes the construction of S . First we show (iii). For this we consider the underlying graph and assume for contradiction that T is a clique of size $d + 1$. By the way S was constructed, and in particular by (3.1), each hyperedge of S intersects at most one set of free vertices that belongs to some cycle edge, so any two vertices from distinct sets of free vertices must be non-adjacent in the underlying hypergraph. To reach a contradiction, we distinguish two cases.

Case 1: T contains a vertex $v \in A_k$ for some k . Since v 's only neighbors are $d - 2$

3. Communicating Instances of Hard Problems

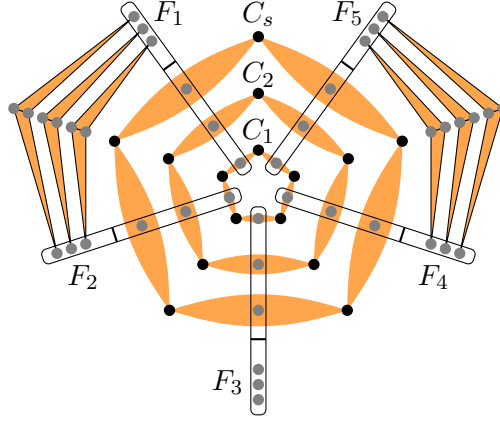


Figure 3.4.: A coordination gadget as in Lemma 3.6 for $d = 3$, $s = 3$ and $m = 2$. Only two of the $m + 1 = 3$ groups in each A_k of S and not all of their incident edges are shown.

other vertices of A_k and the vertices $v_{j,k}$, T contains $v_{j,k}$ and $v_{j',k}$ for $j \neq j'$. However, these vertices are not adjacent since they belong to different even cycles.

Case 2: T contains only vertices of the cycles. Then T must contain a connection vertex v of one of the cycles since any free vertex is adjacent to at most $d - 3$ other free vertices. The vertex v is adjacent to exactly $2d - 2$ vertices, and so T contains a free vertex w in the edge before v and w' in the edge after v in the respective cycle. By the above, w and w' are not adjacent.

This shows that the underlying graph does not contain a $(d+1)$ -clique. For the second part, we observe that $\bigcup_{i \in [m]} U_i$ is the set of connection vertices at even positions of the odd cycles, so they are pairwise non-adjacent.

For the completeness, we construct a perfect matching of $S - U_{j_0}(S)$ for each $j_0 \in [m]$. We define the set of indices

$$J = \{2j_0 + 2j \mid j = 0, \dots, m\}. \quad (3.2)$$

We use the completeness of the even cycle gadgets and take a perfect matching of F_j that covers $U_1(F_j)$ for all $j \in J$, and one that covers $U_2(F_j)$ for the m other choices $j \in [2m + 1] \setminus J$. This is consistent since the even cycles are disjoint. In each odd cycle C_i , we pick the edges $C_{i,j}$ into the matching for $j \in [2m + 1] \setminus J$. This is consistent because these edges do not contain a vertex of $U_{j_0}(S)$ or of $U_1(F_j)$ for $j \in J$, and we never take two consecutive edges. Furthermore, we have covered all vertices of $C - U_{j_0}(S)$. Indeed, the only vertices not yet covered are the $U_2(F_j) = \{v_{j,1}, \dots, v_{j,(d-2)s}\}$ for $j \in J$ and the vertices of the A_k . For each $k \in [(d - 2)s]$ and $j \in J$, we cover the vertex $v_{j,k}$ using a saturation edge with some $(d - 1)$ -group of A_k . This is possible since each A_k contains exactly $|J| = m + 1$ groups. Now all vertices of $S - U_{j_0}$ are covered by a perfect matching.

For the soundness, the claim is that any perfect matching of G has some j_0 such

that U_{j_0} is not covered in the matching by edges of S , whereas all other vertices of S are. Let M be a perfect matching of G . The soundness of the even cycle gadgets guarantees that exactly one of $U_1(F_j)$ and $U_2(F_j)$ are covered with edges of F_j . Let J be the set of indices j for which $U_1(F_j)$ and not $U_2(F_j)$ is covered by the edges of F_j . The only way that M can cover the vertices $U_2(F_j)$ for $j \in J$ is by using $|U_2(F_j)| = (d-2)s$ edges with the A_k 's. Since there are only $|A_k| = m+1$ such edges available for any given k , we have $|J| = |A_k| = m+1$. The only way that M can cover the free vertices of $C_{i,j}$ for $j \in [2m+1] \setminus J$ is by picking $C_{i,j}$ into M . Since M does not contain consecutive edges of C_i and J contains $m+1$ elements of $[2m+1]$, this means that J must be of the form (3.2) for some j_0 . Hence $U_j(S)$ for $j \neq j_0$ is covered in M by edges of the odd cycles and no vertex of U_{j_0} is covered in M by edges of S . ■

Kernel Lower Bounds for Graph Matching Problems

For a graph H , the H -matching problem is to find a maximal number of vertex-disjoint copies of H in a given graph G . This problem is NP-complete whenever H contains a connected component with more than two vertices [KH78] and is in P otherwise.

Clique Packing

We prove Theorem 1.6, that K_d -MATCHING for $d \geq 4$ does not have kernels of size $O(k^{d-1-\epsilon})$ unless $\text{coNP} \subseteq \text{NP/poly}$. For this, we devise a parameter-preserving reduction from the problem of finding a perfect matching in a $(d-1)$ -uniform hypergraph whose underlying graph does not contain a d -clique.

Lemma 3.9. *Let $d \geq 4$ be an integer. There is a \leq_m^p -reduction from $(d-1)$ -SET MATCHING in $(d-1)$ -uniform hypergraphs whose underlying graph does not contain a clique of size d to K_d -MATCHING that does not change the parameter k .*

Proof. Let G be a $(d-1)$ -uniform hypergraph on n vertices without d -clique in its underlying graph. For each edge e of G , we add a new vertex v_e and transform $e \cup \{v_e\}$ into a d -clique in G' . We claim that G has a matching of size $k := n/(d-1)$ if and only if G' has a K_d -matching of size k . The completeness is clear since any given matching of G can be turned into a K_d -matching of G' by taking the respective d -clique for every $(d-1)$ -hyperedge. For the soundness, let G' contain a K_d -matching of size k . Note that any d -clique of G' uses exactly one vertex v_e since the underlying graph of G does not contain any d -cliques and since no two v_e 's are adjacent. Thus every d -clique of G' is of the form $e \cup \{v_e\}$, which gives rise to a matching of G of size k . ■

This combined with Lemma 1.1 and Lemma 3.7 implies Theorem 1.6

General Graph Matching Problems

We prove Theorem 1.7, that H -FACTOR does not have kernels of size $O(k^{2-\epsilon})$ unless $\text{coNP} \subseteq \text{NP/poly}$, whenever H is a connected graph with at least three vertices. In particular, this implies the missing case $d = 3$ of K_d -MATCHING.

3. Communicating Instances of Hard Problems

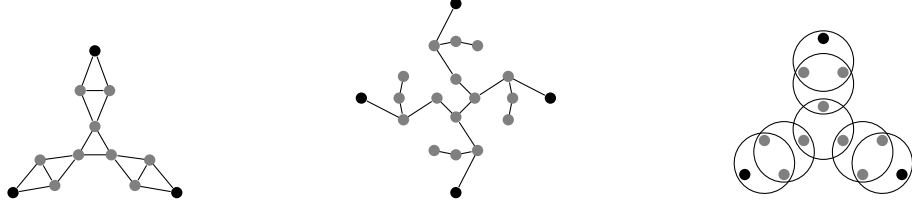


Figure 3.5.: Hyperedge gadgets for different H -matching problems. The outermost, black vertices are the vertices of the simulated hyperedge and the gray vertices are not supposed to be adjacent to any other vertex of the graph. *Left:* Triangle matching. *Middle:* 3-Path matching. *Right:* The general case; each circle represents a copy of H .

We use the coordination gadget of Lemma 3.6 in a reduction from a suitable OR-problem to H -MATCHING. To do so, we translate the coordination gadget for PERFECT d -SET MATCHING to H -FACTOR, which we achieve by replacing hyperedges with the following hyperedge-gadgets of [KH78].

Lemma 3.10. *Let H be a connected graph on $d \geq 3$ vertices. There is a graph $e = e(v_1, \dots, v_d)$ that contains $\{v_1, \dots, v_d\}$ as an independent set such that, for all $S \subseteq \{v_1, \dots, v_d\}$, the graph $e - S$ has an H -factor if and only if $|S| = 0$ or $|S| = d$.*

Proof. Let v be a vertex of H . We construct e as in Figure 3.5. We start with one central copy of H . For each vertex $u \in [d] = V(H)$, we create a new copy H_u of H and denote its copy of v by v_u . Finally, we add an edge between $u \in H$ and $w \in (H_u - v_u)$ if $v_u w$ is an edge of H_u .

For the claim, assume that $0 < |S| < d$. Then $|V(e - S)|$ is not an integer multiple of $d = |V(H)|$ and there can be no H -factor in $e - S$. For the other direction, assume that $|S| = 0$. Then the subgraphs H_u for $u \in [d]$ and H are $d + 1$ pairwise disjoint copies of H in e and form an H -factor of e . In the case $|S| = d$, we observe that the d subgraphs $(H_u - v_u) \cup \{u\}$ form an H -factor of $e - S = e - \{v_1, \dots, v_d\}$. ■

For the proof of the H -PACKING kernel lower bounds, we need the Packing Lemma, Lemma 3.3. The chromatic number $\chi(H)$ is the minimum number of colors required in a proper vertex-coloring of H . The proof of [KH78] shows that H -FACTOR is NP-complete even in case we are looking for an H -factor in $\chi(H)$ -partite graphs. We are going to make use of that in the following reduction.

Lemma 3.11. *There is a \leq_m^p -reduction from OR(H -FACTOR) to H -FACTOR that maps t -tuples of instances of size s each to instances that have at most $\sqrt{t}^{1+o(1)} \cdot \text{poly}(s)$ vertices.*

Proof. Let $p = \chi(H)$ be the chromatic number of H . For an instance G_1, \dots, G_t of OR(H -FACTOR), we can assume w.l.o.g. that the G_i are p -partite graphs with n vertices

in each part. We construct a graph G that has an H -factor if and only if some G_i has an H -factor. For this, we invoke the Packing Lemma, Lemma 3.3, with $d = 2$, and we obtain a p -partite graph P that contains t cliques K_1, \dots, K_t on p vertices each. We identify the vertex set of G_i with $V(K_i) \times [n]$ injectively in such a way that vertices in the same colour class have the same first coordinate. We define an intermediate p -partite graph G' on the vertex set $V(P) \times [n]$ as $G' = G_1 \cup \dots \cup G_t$. To obtain G from G' , we add p coordination gadgets of Lemma 3.6 with $m = \sqrt{t}^{1+o(1)}$ and $d = p$. For each colour class $C \subset V(G')$, we add a coordination gadget where the $U_i \subset C$ are those vertices that project to the same vertex in P . Finally, we replace each p -hyperedge by the gadget in Lemma 3.10, which finishes the construction of G .

For the completeness of the reduction, assume G_i has an H -factor M . To construct an H -factor of G , we start by using M to cover the vertices $V(G_i)$ in G . The completeness of the coordination gadgets guarantees that we find a perfect matching in the d -uniform hypergraph $G' - V(G_i)$ that uses only hyperedges of the coordination gadgets. By Lemma 3.10, this gives rise to an H -factor of G .

For the soundness, assume we have an H -factor M of G . Lemma 3.10 guarantees that the edge gadgets can be seen as p -hyperedges in the intermediate graph G' . Soundness of the coordination gadgets guarantees that M leaves exactly one group free per part. Now let H' be a copy of H that is contained in G but not in any of the gadgets. Since H' has chromatic number p , H' intersects all p parts and has an edge between any two distinct parts. By construction of G , this implies that the projection of H onto P is a clique. By the packing lemma, this clique is one of the K_i 's. Therefore, each H' of the H -factor M that is not in one of the gadgets is contained in G_i , which implies that G_i has an H -factor.

The claim follows since G is a graph on $\sqrt{t}^{1+o(1)}$ poly(s) vertices that has an H -factor if and only if some G_i has an H -factor. ■

Together with Lemma 1.1, the above implies Theorem 1.7, our kernel lower bounds for H -FACTOR.

Kernels for Graph Matching Problems

The sunflower kernelization in Theorem 1.4 immediately transfers to H -MATCHING for any fixed graph H and yields kernels with $O(k^d)$ edges. For graphs H , Moser [Mos09] shows that H -MATCHING has kernels with $O(k^{d-1})$ vertices where $d = |V(H)|$, but this gives only the weaker bound $O(k^{2d-2})$ on the number of edges. Here we show that for some specific H , we can obtain kernels that are better than the $O(k^d)$ bound implied by Theorem 1.4. As a very simple example, we show this first for $K_{1,d}$ -MATCHING, the problem of packing vertex-disjoint stars with d leaves.

Observation 3.1. $K_{1,d}$ -MATCHING has kernels with $O(k^2)$ edges.

Proof. Let (G, k) be an instance of $K_{1,d}$ -MATCHING. If G has a vertex v of degree at least $dk + 1$, let e be an edge incident to v . We claim that we can safely remove e . If $G - e$ has a $K_{1,d}$ -matching of size k , then this also holds for G . For the other direction,

3. Communicating Instances of Hard Problems

let M be a $K_{1,d}$ -matching of size k in G . If M does not contain e , it is also a matching of $G - e$. Otherwise M contains e . Let M' be obtained from M by removing the star that contains e . Now v is not contained in M' . Since M' induces at most $d(k - 1)$ vertices, at least $d + 1$ neighbors of v are not contained in M' . Even if we remove e , we can therefore augment M' with a vertex-disjoint star that is centered at v and has d leaves. This yields a star matching of size k in $G - e$.

For the kernelization, we repeatedly delete edges incident to high-degree vertices. Then every vertex has degree at most dk . Now we greedily compute a maximal star matching M and answer 'yes' if M has size k . Otherwise, we claim that the graph has most $O(k^2)$ edges: Since M induces at most dk vertices, the degree bound implies that at most $(dk)^2$ edges are incident to M . The vertices of G outside of M have at most $d - 1$ neighbors outside of M because they would otherwise have been added to M . Thus there are at most $(d - 1) \cdot (dk)^2$ edges not incident to M . Thus G has at most $d^3 \cdot k^2$ edges. ■

By Theorem 1.7, it is unlikely that star matching problems have kernels with $O(k^{2-\epsilon})$ edges, so the above kernels are likely to be asymptotically optimal.

3.6. Other Applications

To illustrate the use of our oracle communication model we describe two applications in the original framework of Bodlaender et al. [BDFH09].

For several NP-hard parameterized problems L there exists a \leq_m^p -reduction from $\text{OR}(L)$ to L that maps t instances of size s each to a single instance of L of size $\text{poly}(s) \cdot t^{1+o(1)}$ and parameter $k = \text{poly}(s)$. For example, for problems like SAT and CLIQUE, such reductions follow from the disjoint union construction mentioned in the introduction. For certain other problems such reductions are more involved but still exist (see [CFM07, BTY09, DLS09, FFL⁺09, KW10, KW09, KMW10] for examples). Whenever such reductions exist, Lemma 3.1 implies that L does not have an oracle communication protocol of cost $\text{poly}(k)$ unless $\text{coNP} \subseteq \text{NP}/\text{poly}$. In particular, such problems do not have kernels of polynomial size unless $\text{coNP} \subseteq \text{NP}/\text{poly}$.

Turing kernelizations

Fernau et al. [FFL⁺09] exhibit a parameterized problem that has no standard kernel of polynomial size unless $\text{coNP} \subseteq \text{NP}/\text{poly}$, but does have a ‘‘Turing kernelization’’ of size $O(k^3)$ in the following sense: The problem has a self-reduction which, on inputs of size s and parameter k , makes at most s queries, all of which are of size $O(k^3)$. Using oracle communication protocols that implement general reductions rather than mapping reductions, Lemma 3.1 allows us to rule out the following for that problem, assuming $\text{coNP} \not\subseteq \text{NP}/\text{poly}$: Reductions that, on inputs of size s and parameter k , make at most $s^{1-\epsilon}$ queries for some positive real ϵ and only query instances of bitlength bounded by a polynomial in k . In particular, this shows that the number of queries in the Turing kernel of Fernau et al. [FFL⁺09] is likely to be tight – reducing it from s to $s^{1-\epsilon}$ for some positive real ϵ would collapse the polynomial-time hierarchy.

Density of NP-hard languages

Buhrman and Hitchcock [BH08b] showed that a language S that contains no more than $2^{n^{o(1)}}$ strings of any length n cannot be hard for NP under reductions that make $n^{1-\epsilon}$ queries for some positive real ϵ unless $\text{coNP} \subseteq \text{NP/poly}$. The proof in [BH08b] is a modification of the proof in [FS08]. As an illustration of the power of our oracle communication protocols, we show that this result immediately follows from Lemma 3.1 using an oracle that actively tries to extract enough information from the first player to decide the membership to S of any query that the first player wants to make.

Suppose such an NP-hard language S does exist and consider the reduction from SAT to S that makes $n^{1-\epsilon}$ queries. Since the reduction runs in polynomial time, the size of the queries is bounded by $m = \text{poly}(n)$. Consider the lexicographic ordering of all strings of length up to m . The set S breaks up this ordering into at most $2 \cdot |S \cap \{0, 1\}^{\leq m}| + 1$ intervals on which the membership to S is constant. In order for the oracle to decide the membership to S of a query, it suffices for the oracle to figure out which interval the query falls in. It can do so by running a binary search with the help of the first player, who knows the exact query. The binary search only takes $\log(2 \cdot |S \cap \{0, 1\}^{\leq m}| + 1)$ bits of communication from the first player to the oracle. Overall, this leads to a communication protocol for SAT of cost $O(n^{1-\epsilon} \cdot \log(2 \cdot |S \cap \{0, 1\}^{\leq m}| + 1)) = O(n^{1-\epsilon+o(1)})$. Combining this protocol with the \leq_m^p -reduction from OR(SAT) to SAT mentioned above, we obtain an oracle communication protocol for OR(SAT) of cost $O(\text{poly}(s) \cdot t^{1-\epsilon+o(1)})$ on inputs consisting of t instances of size s each. As the latter quantity is $O(t \log t)$ whenever t is a sufficiently large polynomial in s , Lemma 3.1 implies that $\text{coNP} \subseteq \text{NP/poly}$.

Notes

The complementary witness lemma in §3.1, the AP-free set based proof in §3.2, the corollaries for satisfiability in §3.3 and for covering problems §3.4, and the discussion in §3.6 are joint work with Dieter van Melkebeek and appeared in [DvM10]. The elementary proof in §3.2 and the results about packing problems in §3.5 is unpublished joint work with Dániel Marx.

4. Packing Edge-disjoint Cliques

In this chapter we establish the Packing Lemma, Lemma 3.3, a graph-theoretical result that is useful for proving the hardness of oracle communication. It is a critical ingredient in the original proof of Theorem 1.2, the hardness of vertex cover, that appeared in [DvM10] and is spelled out in §3.2. Even though we later found a proof of Theorem 1.2 that does not use the Packing Lemma, it remains an important tool in the area. For example, we used it to prove Theorem 1.7, the hardness of cost $O(n^{2-\epsilon})$ protocols for H -packing problems.

Our Construction

We first develop the construction for the case $d = 2$, i.e., for standard graphs, and then show how to generalize it to d -uniform hypergraphs for arbitrary $d \geq 2$. We also discuss the relationship of our construction to earlier ones. We need to construct a graph P on few vertices such that

- (i) the edges of P partition into t cliques K_1, \dots, K_t on p vertices each, and
- (ii) P contains no other cliques on p vertices.

We first focus on realizing condition (i) and then see how to modify the construction to also realize (ii).

We construct P as an p -partite graph and think of the p partitions as the columns of a two-dimensional array of vertices, say of size q by p . Each of the K_i 's then contains exactly one vertex from each of the p columns. Condition (i) expresses that P is a packing of the K_i 's. The trivial packing consists of the disjoint union and requires $q = t$ rows, resulting in $p \cdot t$ vertices in total. The trivial packing is wasteful because it leaves many of the potential edges unused. In an ideal packing each of the q^2 potential edges between two columns of the array are assigned to some K_i . This would only require a number of rows $q = \sqrt{t}$ and therefore $p \cdot \sqrt{t}$ vertices. We can realize such a tight packing by picking the vertex of K_i in column j as the value of j under a hash function h_i from a minimum 2-universal family. If q is a prime at least p , we can identify the rows as well as the columns with elements of \mathbb{F}_q and use the family of linear functions over \mathbb{F}_q . More precisely, we construct P on the vertex set $V(P) = [p] \times \mathbb{F}_q$ as the union of the t cliques K_i on the vertex sets $V(K_i) = \{(j, h_i(j)) \mid j \in [p]\}$, where h_i is a linear function over \mathbb{F}_q uniquely associated with K_i . See Figure 4.1a. Note that there are q^2 distinct linear functions h_i over \mathbb{F}_q , so we can accommodate that many cliques K_i . Moreover, since two points define a line, every edge of P is contained in exactly one of the K_i 's. For

4. Packing Edge-disjoint Cliques

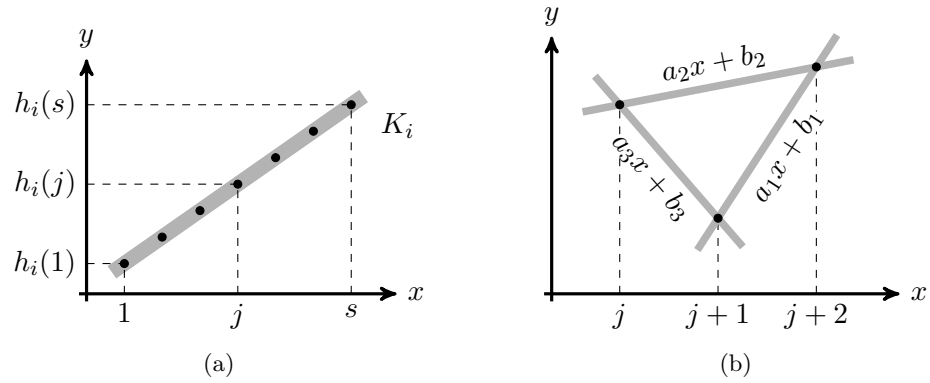


Figure 4.1.: (a) The placement of one of the K_i 's. (b) Triangle on three consecutive abscissae.

arbitrary values of p and t , we can pick q to be the first prime $q \geq \max(p, \sqrt{t})$, resulting in a packing with $O(p \cdot \max(p, \sqrt{t}))$ vertices.

Note that this P is in fact a complete p -partite graph and therefore fails to satisfy condition (ii) miserably – *every* clique of size p that has one vertex from each column is present in P , which is many more than just the K_i 's. In order to remedy that problem, let us analyze the cliques of size p in P more closely.

Let K denote a clique of size p in P . Each of the p columns of P has to contain exactly one vertex of K , i.e., there exists a function $h : [p] \rightarrow \mathbb{F}_q$ such that $V(K) = \{(j, h(j)) \mid j \in [p]\}$. We would like to ensure that K coincides with one of the cliques K_i , or equivalently, that the function h coincides with one of the linear functions h_i .

Consider three consecutive columns, j , $j+1$, and $j+2$, and the triangle that K induces between them – see Figure 4.1b, where each edge is labeled by the linear function h_i defining the clique K_i to which the edge belongs. We claim that the highest-order coefficients of those linear functions have to form an arithmetic progression. This follows by considering the two paths in Figure 4.1b that go from the vertex in column j to the one in column $j+2$. The direct path on top involves an increase in y -value of $2a_2$, whereas the indirect path on the bottom involves an increase in y of a_3 followed by an increase of a_1 . Since both paths end up at the same point, we have that

$$2a_2 = a_1 + a_3, \tag{4.1}$$

or equivalently, that $a_3 - a_2 = a_2 - a_1$, or yet equivalently, that the sequence a_1, a_2, a_3 forms an arithmetic progression. If we restrict the highest-order coefficients of the linear functions to come from a subset $A \subseteq \mathbb{F}_q$ that contains no nontrivial arithmetic progressions of length three, the arithmetic progression a_1, a_2, a_3 has to be trivial, i.e., $a_1 = a_2 = a_3$. The latter implies that the three lines in Figure 4.1b coincide. As this implication holds for all choices of three consecutive columns, we conclude that all vertices of K lie on a single line defined by one of the h_i 's, as we wanted.

Of course, the additional restriction on the highest-order coefficients means that we need to choose q larger. However, we only need to increase q slightly thanks to the existence of efficiently constructible subsets $A \subseteq \mathbb{F}_q$ of high density that contain no nontrivial arithmetic progressions of length three. For our purposes the following classical result from additive combinatorics suffices.

Lemma 4.1 (AP₃-Free Sets [SS42]). *For every positive integer q there exists a subset $A \subseteq \mathbb{Z}_q$ of size at least $q^{1-o(1)}$ which contains no nontrivial arithmetic progressions of length three. Furthermore, such a set A can be determined in time polynomial in q .*

For completeness we provide a proof of Lemma 4.1 in the Appendix. The resulting graph P has $p \cdot q$ vertices where $q = O(\max(p, \sqrt{t}^{1+o(1)}))$.

This finishes the construction of the packing lemma for the case of standard graphs. The generalization to d -uniform hypergraphs follows by using polynomials of degree $d-1$ instead of linear functions over \mathbb{F}_q . Their use guarantees requirement (i) in Lemma 3.3. Regarding requirement (ii), the following proof shows that the case $d > 2$ reduces to the case $d = 2$. For arbitrary $d \geq 2$, we fulfill requirement (ii) by restricting the coefficient of degree $d-1$ to a set that contains no nontrivial arithmetic progressions of length three, namely the set $A \subseteq \mathbb{F}_q$ determined in Lemma 4.1.

Lemma 3.3 (restated). *For any integers $p \geq d \geq 2$ and $t > 0$ there exists an p -partite d -uniform hypergraph P on $O(p \cdot \max(p, t^{1/d+o(1)}))$ vertices such that*

- (i) *the hyperedges of P partition into t cliques K_1, \dots, K_t on p vertices each, and*
- (ii) *P contains no cliques on p vertices other than the K_i 's.*

Furthermore, for any fixed d , the hypergraph P and the K_i 's can be constructed in time polynomial in p and t .

Proof (of Lemma 3.3). Let q be the smallest prime such that $q \geq p$ and $|A| \cdot q^{d-1} \geq t$, where A denotes the set given by Lemma 4.1. We have that $q = O(\max(p, t^{1/d+o(1)}))$ and can compute q and the set A in time polynomial in p and t .

Let $V(P) = [p] \times \mathbb{F}_q$. We consider polynomials of degree at most $d-1$ over \mathbb{F}_q whose coefficient of x^{d-1} belongs to A . Note that there are $|A| \cdot q^{d-1} \geq t$ such polynomials. For $i \in [t]$, let h_i denote the i th such polynomial in lexicographic order, and let K_i be the complete d -uniform hypergraph on vertex set $V(K_i) = \{(j, h_i(j)) \mid j \in [p]\}$. We define the d -uniform hypergraph P as the union of the t cliques K_i . The hypergraphs P and K_i can be constructed in time polynomial in p and t .

In order to argue property (i), it suffices to observe that every hyperedge of P is contained in at most one of the K_i 's. This follows because the requirement that a given hyperedge of P belongs to K_i is equivalent to stipulating the value of h_i on d distinct values $j \in [p]$, which uniquely determines h_i as a polynomial of degree at most $d-1$ over \mathbb{F}_q , and therefore determines i .

In order to argue property (ii), we need to establish the following for any function $h : [p] \rightarrow \mathbb{F}_q$: If for every subset $D \subseteq [p]$ of size d there exists an $i \in [t]$ such that h

4. Packing Edge-disjoint Cliques

and h_i agree on D , then there exists an $i \in [t]$ such that h and h_i agree on all of $[p]$. The property follows by applying the next claim to successive values of $j \in [p-d]$, where q_k denotes the polynomial h_i which the hypothesis gives for the subset $D = [j, j+d] \setminus \{k\}$.

Claim. For each $k \in [j, j+d]$, let q_k be a polynomial of degree at most $d-1$ such that the set of coefficients of degree $d-1$ of the q_k 's contains no nontrivial arithmetic progression of length three. If for all $k, \ell \in [j, j+d]$, the polynomials q_k and q_ℓ agree on $[j, j+d] \setminus \{k, \ell\}$, then the polynomials q_k are all the same.

We prove the claim by induction on d . We already argued the base case $d=2$, captured by Figure 4.1b, earlier in Section 4. For the inductive step, assume the claim holds for $d-1$ and let us prove it for d . Let q_j, \dots, q_{j+d} be polynomials as in the claim. For each $k \in [j, j+d-1]$, define q'_k as the difference quotient $\Delta_{j+d}(q_k)$, i.e., $q'_k : [j, j+d-1] \rightarrow \mathbb{F}_q$ such that $q'_k(x) = (q_k(x) - q_k(j+d))/(x - j - d)$ for $x \in [j, j+d-1]$. Note that q'_k is a polynomial of degree at most $d-2$ whose coefficient of degree $d-2$ equals the coefficient of q_k of degree x^{d-1} . Moreover, for $k, \ell \in [j, j+d-1]$, the polynomials q'_k and q'_ℓ agree on each $x \in [j, j+d-1] \setminus \{k, \ell\}$ because the polynomials q_k and q_ℓ agree on both x and $j+d$. Thus, by the induction hypothesis, all polynomials q'_k are the same. By the definition of $q'_k = \Delta_{j+d}(q_k)$ and the fact that the polynomials q_k for $k \in [j, j+d-1]$ agree on $j+d$, this implies that the polynomials q_k for $k \in [j, j+d-1]$ are all the same, say q . All that remains to show is that the polynomial q_{j+d} also coincides with q . The latter follows because q_{j+d} is a polynomial of degree at most $d-1$ which agrees with the polynomial q of degree at most $d-1$ on all d points in $[j, j+d-1]$. \blacksquare

Related Constructions

After we developed our construction we learned about similar applications of high-density subsets of the integers without nontrivial arithmetic progressions of length three.

Back in 1976, Ruzsa and Szemerédi [RS78] constructed dense three-partite graphs whose edges partition into triangles and that contain no other triangles. Their construction corresponds to the case $(d, s) = (2, 3)$ of our Packing Lemma, and appears between any three consecutive columns of our construction for $d=2$ and general p . Our geometric derivation of the arithmetic progression condition (4.1), as captured in Figure 4.1b, may be new; all the derivations we have found in the literature work by manipulating equations in a – to us – less intuitive way.

Different aspects of the Ruzsa-Szemerédi construction matter for the various applications we know of in the theory of computing. For their soundness analysis of graph tests for linearity, Håstad and Wigderson [HW03] use the interpretation that for each of the q points in the first column, the triangles involving that point span an induced matching of $q^{1-o(1)}$ edges between the other columns.

Another application area is the lower bounds for testing the graph property of being F -free, where F is some fixed graph. An ϵ -tester for this property accepts all graphs that are F -free, and rejects all graphs that are at least ϵ away from being F -free, i.e., from which at least ϵn^2 edges need to be removed to make it F -free [GGR98]. A strategy

for proving lower bounds on the number of queries of such a tester is to construct high-density graphs G with the following properties: (i) the edges of G partition into copies of F , and (ii) G contains few other copies of F so the total number of copies of F in G is significantly less than expected in a random graphs of the same density as G [Alo02]. Qualitatively, (i) implies that G is far from being F -free, and (ii) implies that testers with few queries have a small probability of detecting a violation of F -freeness on input G . Alon and coauthors [Alo02, AS06, AS04, AKKR08, AS05] constructed such graphs G for various F based on the work of Ruzsa and Szemerédi [RS78].

The requirements for our application are similar but not identical to the ones for property testing. On the one hand we only need to consider the cases where F is a clique; on the other hand the graphs G cannot contain *any* copy of F other than those in which the edges partition. Our actual construction is very similar to the one Alon and Shapira [AS05] develop. Their construction would also work for our purposes. Our proof differs from theirs and makes the arithmetic progression condition more transparent. Our construction slightly improves¹ on theirs as we only restrict the highest-order coefficient to the set A , whereas they restrict all coefficients to that set.

Notes

The content of this chapter is joint work with Dieter van Melkebeek and appeared in [DvM10].

¹This allows us to relax the condition $q(\epsilon) = \max\{m : (f(m))^k \geq \epsilon\}$ in Lemma 4.1 of [AS05] to $q(\epsilon) = \max\{m : f(m) \geq \epsilon\}$.

5. Exponential Time Counting Complexity

5.1. Counting Independent Sets

In this section, we establish Theorem 1.9, the hardness of counting independent sets and of #2-SAT. For the proof, we make use of the ETH-hardness of the following problem.

Name UNIQUE 3-SAT.

Input 3-CNF formula φ with m clauses and at most one satisfying assignment.

Decide Is φ satisfiable?

Calabro et al. [CIKP03] use an isolation lemma for d -CNF formulas to show that solving this problem in subexponential time implies that the (randomized) exponential time hypothesis fails.

Theorem 5.1 (Corollary 2 of [CIKP03]).

ETH holds if and only if UNIQUE 3-SAT *requires time* $\exp(\Omega(m))$.

We are now in the position to prove Theorem 1.9.

Theorem 1.9 (restated). *Counting independent sets and #2-SAT both require time* $\exp(\Omega(m))$ *under ETH, where* m *is the number of edges and clauses, respectively.*

Proof. Let φ be an instance of UNIQUE 3-SAT with m clauses. We construct a graph G with $O(m)$ edges that has an odd number of independent sets if and only if φ is satisfiable. For each variable x , we introduce vertices x and \bar{x} , and the edge $(x\bar{x})$. This makes sure that any independent set of G chooses at most one of $\{x, \bar{x}\}$, so we can interpret the independent set as a partial assignment to the variables of φ . For each clause $c = (\ell_1 \vee \ell_2 \vee \ell_3)$ of φ , we introduce a clique in G that consists of seven vertices c_1, \dots, c_7 . These vertices correspond to the seven partial assignments that assign truth values to the literals ℓ_1, ℓ_2 , and ℓ_3 in such a way that c is satisfied. Any independent set of G contains at most one c_i for each clause c . To ensure that the independent set chooses the variables and partial assignments of the clauses consistently, we add an edge for every c_i and every variable x occurring in the clause c : If the partial assignment that corresponds to c_i sets x to true, we add $(c_i\bar{x})$ to G ; otherwise, we add (c_ix) to G . To finalize the construction, we introduce guard vertices g_x and g_c for every variable x and every clause c , along with the edges $(g_x x)$, $(g_x \bar{x})$, and $(g_c c_i)$ for $i = 1, \dots, 7$.

We now prove that G has the required properties. First, any independent set contains at most n literal vertices and at most m clause vertices. *Good* independent sets are those that contain exactly n literal and m clause vertices (and no guard vertex). *Good* independent sets correspond to the satisfying assignments of φ in a natural way. We

5. Exponential Time Counting Complexity

now show that the number of bad independent sets is even. For this, let S be a bad independent set, that is, S is disjoint from $\{x, \bar{x}\}$ for some x or it is disjoint from $\{c_1, \dots, c_7\}$ for some clause c . By construction, the neighborhood of either g_x or g_c is disjoint from S . Let g be the lexicographically first guard vertex whose neighborhood is disjoint from S . Both the sets $S \setminus \{g\}$ and $S \cup \{g\}$ are bad independent sets and S is one of these sets. Formally, we can therefore define a function that maps these sets onto each other. This function is a well-defined involution on the set of bad independent sets, and it does not have any fixed points. Therefore, the number of bad independent sets is even, and the parity of the number of independent sets of G is equal to the parity of the number of satisfying assignments of φ .

The above reduction shows that an $\exp(o(m))$ -time algorithm for counting independent sets modulo 2 implies an $\exp(o(m))$ -time algorithm for UNIQUE 3-SAT. By Theorem 5.1, this implies that ETH fails.

To establish the hardness of #2-SAT, we reduce from counting independent sets. Let G be a graph. For each vertex v , we introduce a variable v , and each edge (uv) becomes a clause $(\bar{u} \vee \bar{v})$. The satisfying assignments of the so constructed 2-CNF formula are in one-to-one correspondence with the independent sets of G . ■

5.2. The Permanent

This section contains the proof of Theorem 1.10. With $[0, n] = \{0, 1, \dots, n\}$ we establish the reduction chain $\#3\text{-SAT} \preceq \text{PERM}^{-1,0,1} \preceq \text{PERM}^{[0,n]} \preceq \text{PERM}^{01}$ while taking care of the instance sizes.

Theorem 1.10 (restated).

- (i) $\text{PERM}^{-1,0,1}$ and $\text{PERM}^{[0,n]}$ require time $\exp(\Omega(m))$ under #ETH.
- (ii) PERM^{01} requires time $\exp(\Omega(m/\log n))$ under #ETH.
- (iii) PERM^{01} requires time $\exp(\Omega(m))$ under ETH.

Proof. To establish (i), we reduce #3-SAT in polynomial time to $\text{PERM}^{-1,0,1}$ such that 3-CNF formulas φ with m clauses are mapped to graphs G with $O(m)$ edges. For technical reasons, we preprocess φ such that every variable x occurs equally often as a positive literal and as a negative literal \bar{x} (e.g., by adding trivial clauses of the form $(x \vee \bar{x} \vee \bar{x})$ to φ). We construct G with $O(m)$ edges and weights $w : E \rightarrow \{\pm 1\}$ such that $\#\text{SAT}(\varphi)$ can be derived from $\text{per } G$ in polynomial time. For weighted graphs, the permanent is

$$\text{per } G = \sum_{C \subseteq E} w(C), \quad \text{where } w(C) = \prod_{e \in C} w(e).$$

The sum above is over all cycle covers C of G , that is, subgraphs (V, C) with an in- and outdegree of 1 at every vertex.

In Figure 5.1, the gadgets of the construction are depicted. For every variable x that occurs in φ , we add a *selector gadget* to G . For every clause $c = \ell_1 \vee \ell_2 \vee \ell_3$ of φ , we

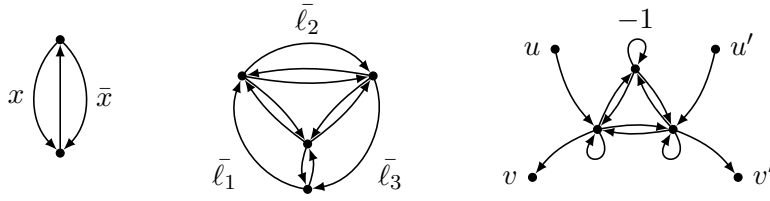


Figure 5.1.: Left: A selector gadget for variable x . Depending on which of the two cycles is chosen, we assume x to be set to true or false. Middle: A clause gadget for the clause $\ell_1 \vee \ell_2 \vee \ell_3$. The gadget allows all possible configurations for the outer edges, except for the case that all three are chosen (which would correspond to $\ell_1 = \ell_2 = \ell_3 = 0$). Right: An equality gadget that replaces two edges uv and $u'v'$. The top loop carries a weight of -1 . It can be checked that the gadget contributes a weight of -1 if all four outer edges are taken, $+2$ if none of them is taken, and 0 otherwise.

add a *clause gadget* to G . Finally, we connect the edge labelled by a literal ℓ in the selector gadget with all occurrences of ℓ in the clause gadgets, using *equality gadgets*. This concludes the construction of G .

The number of edges of the resulting graph G is linear in the number of clauses. The correctness of the reduction follows along the lines of [Pap94] and [BD07]. The satisfying assignments stand in bijection to cycle covers of weight $(-1)^i 2^j$ where i (resp. j) is the number of occurrences of literals set to false (resp. true) by the assignment, and all other cycle covers sum up to 0 . Since we preprocessed φ such that $i = j$ holds and i is constant over all assignments, we obtain $\text{per } G = (-2)^i \cdot \#\text{SAT}(\varphi)$.

For the second part of (i), we reduce $\text{PERM}^{-1,0,1}$ in polynomial time to $\text{PERM}^{[0,n]}$ by interpolation: On input G , we conceptually replace all occurrences of the weight -1 by a variable x and call this new graph G_x . We can assume that only loops have weight x in G_x because the output graph G from the previous reduction has weight -1 only on loops. Then $p(x) = \text{per } G_x$ is a polynomial of degree $d \leq n$.

If we replace x by a value $a \in [0, n]$, then G_a is a weighted graph with as many edges as G . As a consequence, we can use the oracle to compute $\text{per } G_a$ for $a = 0, \dots, d$ and then interpolate, to get the coefficients of the polynomial $p(x)$. At last, we return the value $p(-1) = \text{per } G$. This completes the reduction, which queries the oracle $d+1$ graphs that have at most m edges each.

For (ii), we have to get rid of weights larger than 1 . Let G_a be one query of the last reduction. Again we assume that $a \leq n$ and that weights $\neq 1$ are only allowed at loop edges. We replace every edge of weight a by the gadget that is drawn in Figure 5.2, and call this new unweighted graph G' . It can be checked easily that the gadget indeed simulates a weight of a (parallel paths correspond to addition, serial edges to multiplication), i.e., $\text{per } G' = \text{per } G_a$. Unfortunately, the reduction increases the number of edges by a superconstant factor: The number of edges of G' is $m(G') \leq (m+n \log a) \leq O(m+n \log n)$.

5. Exponential Time Counting Complexity

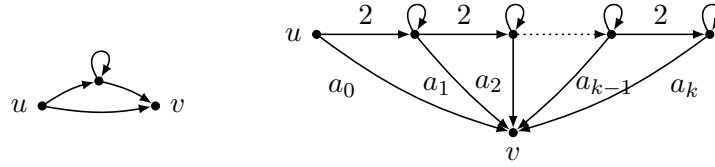


Figure 5.2.: Left: This gadget simulates in unweighted graphs edges uv of weight 2. Right: This gadget simulates edges uv of weight $a = \sum_{i=0}^k a_i 2^i$ with $a_i \in \{0, 1\}$.

But since $m(G')/\log m(G') \leq O(m)$, the reduction implies that (ii).

For (iii), we assume that ETH holds. Theorem 5.1 gives that UNIQUE 3-SAT cannot be computed in time $\exp(o(m))$. Now we apply the first reduction of (i) to a formula φ which is promised to have at most one satisfying assignment. Then the number per $G = (-2)^i \cdot \#\text{SAT}(\varphi)$ is either 0 or $(-2)^i$. In G , we replace each edge of weight -1 by a gadget of weight $2 \equiv -1 \pmod{3}$ and similarly get that $(\text{per } G \pmod{3})$ is $(0 \pmod{3})$ or $(4^i \pmod{3})$. Since $(4^i \pmod{3}) \neq 0$, we can distinguish the case in which φ is unsatisfiable from the case in which φ has exactly one satisfying assignment. ■

5.3. The Tutte Polynomial

In this section, we prove Theorem 1.11, our hardness results for evaluating the Tutte polynomial. For quick reference, we state the propositions in which the individual results are proved and the techniques used in each case.

Theorem 1.11 (restated). *Let $(x, y) \in \mathbb{Q}^2$. Under #ETH,*

- (i) ■ $\text{TUTTE}(x, y)$ requires time $\exp(\Omega(n))$
if $(x-1)(y-1) \neq 1$ and $y \notin \{0, \pm 1\}$,
(Proposition 5.3 in §5.3.2; stretching and thickening)
- (ii) ■ $\text{TUTTE}^{01}(x, y)$ requires time $\exp(\Omega(n))$
if $y = 0$ and $x \notin \{0, \pm 1\}$,
(Proposition D.1 in Appendix D; Linial's reduction)
- (iii) ■ $\text{TUTTE}^{01}(x, y)$ requires time $\exp(\Omega(m/\log^2 m))$
if $x = 1$ and $y \neq 1$,
(Proposition 5.5 in §5.3.3; inflation with Bounce graphs)
- (iv) ■ $\text{TUTTE}^{01}(x, y)$ requires time $\exp(\Omega(m/\log^3 m))$
if $(x-1)(y-1) \notin \{0, 1\}$ and $(x, y) \notin \{(-1, -1), (-1, 0), (0, -1)\}$.
(Proposition 5.4 in §5.3.3; inflation with Theta graphs)

5.3.1. Hyperbolas in the Tutte plane

Our first goal is to show that computing the coefficients of the Tutte polynomial is hard along any single hyperbola. It is useful to view the Tutte polynomial in the Fortuin–Kasteleyn formulation:

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

where $k(A)$ is the number of connected components in the subgraph (V, A) . The connection to the Tutte polynomial is given by

$$\begin{aligned} T(G; x, y) &= (x-1)^{-k(E)} (y-1)^{-|V|} Z(G; q, w), \\ &\text{where } q = (x-1)(y-1) \text{ and } w = y-1, \end{aligned} \tag{5.1}$$

see [Sok05, eq. (2.26)].

The Ising Hyperbola

We begin with the case $q = 2$.

Proposition 5.1. *Computing the coefficients of the polynomial $w \mapsto Z(G; 2, w)$ for given simple graph G requires time $\exp(\Omega(m))$ under #ETH.*

Proof. The reduction is from #MAXCUT and well-known, see, e.g., [JS93, Theorem 15].

Name #MAXCUT

Input Simple undirected graph G .

Output The number of maximum cuts.

A maximum cut is a set $C \subseteq V(G)$ that maximizes the number $|E(C, \overline{C})|$ of edges of G that cross the cut. By the Fortuin–Kasteleyn identity [Sok05, Theorem 2.3], one can express $Z(G; 2, w)$ for $G = (V, E)$ as

$$\sum_{\sigma: V \mapsto \pm 1} \prod_{uv \in E} (1 + w \cdot [\sigma(u) = \sigma(v)]).$$

Here the Iverson bracket $[P]$ is 1 if P is true and is 0 if P is false. The sets $\sigma^{-1}(1)$ and $\sigma^{-1}(-1)$ define a cut in G , so we can write the above expression as

$$\sum_{U \subseteq V} \prod_{\substack{uv \in E \\ [u \in U] = [v \in U]}} (1 + w) = \sum_{C \subseteq V(G)} (1 + w)^{m - |E(C, \overline{C})|},$$

Now, the coefficient of $(1 + w)^{m-c}$ in $Z(G; 2, w)$ is the number of cuts in G of size c . In particular, after some interpolation, we can compute the number of maximum cuts in G from the coefficients of $w \mapsto Z(G; 2, w)$. But as we observe in Appendix D, #MAXCUT requires time $\exp(\Omega(m))$ under #ETH. ■

5. Exponential Time Counting Complexity

The Multivariate Tutte Polynomial

For other q , in particular nonintegers, it is simpler to work with a *multivariate* formulation of the Tutte polynomial due to Fortuin and Kasteleyn [FK72]. We use Sokal's definition [Sok05]: Let $G = (V, E)$ be an undirected graph whose edge weights are given by a function $\mathbf{w}: E \rightarrow \mathbb{Q}$. Then

$$Z(G; q, \mathbf{w}) = \sum_{A \subseteq E} q^{k(A)} \prod_{e \in A} \mathbf{w}(e). \quad (5.2)$$

If \mathbf{w} is single-valued, in the sense that $\mathbf{w}(e) = w$ for all $e \in E$, we recover $Z(G; q, w)$.

The conceptual strength of the multivariate perspective is that it turns the Tutte polynomial's second variable y , suitably transformed, into an edge weight of the graph. In particular, the multivariate formulation allows the graph to have different weights on different edges, which turns out to be a dramatic technical simplification even when, as in the present work, we are ultimately interested in the single-valued case.

Sokal's polynomial vanishes at $q = 0$, so we sometimes use the polynomial

$$Z_0(G; q, \mathbf{w}) = \sum_{A \subseteq E} q^{k(A) - k(E)} \prod_{e \in A} \mathbf{w}(e),$$

which gives something non-trivial for $q = 0$ and is otherwise a proxy for Z :

$$Z(G; q, \mathbf{w}) = q^{k(E)} Z_0(G; q, \mathbf{w}). \quad (5.3)$$

Three-terminal minimum cut

For $q \notin \{1, 2\}$, we first establish that with two different edge weights, one of them negative, the multivariate Tutte polynomial computes the size of a 3-terminal minimum cut, for which we observe hardness under #ETH in Appendix D. This connection has been used already in [GJ07, GJ08], with different reductions, to prove hardness of approximation.

The graphs we consider here are connected and have rather simple weight functions. The edges are partitioned into two sets $E \dot{\cup} T$, and for fixed rational w the weight function is given by

$$\mathbf{w}(e) = \begin{cases} -1, & \text{if } e \in T, \\ w, & \text{if } e \in E. \end{cases} \quad (5.4)$$

For such a graph, we have

$$Z_0(G; q, \mathbf{w}) = \sum_{A \subseteq E \dot{\cup} T} q^{k(A) - 1} w^{|A \cap E|} (-1)^{|A \cap T|}. \quad (5.5)$$

For fixed G and q , this is a polynomial in w of degree at most m .

Lemma 5.1. *Let q be a rational number with $q \notin \{1, 2\}$. Computing the coefficients of the polynomial $w \mapsto Z_0(G; q, \mathbf{w})$, with \mathbf{w} as in (5.4), for a given simple graph G requires time $\exp(\Omega(m))$ under #ETH. Moreover, this is true even if $|T| = 3$.*

Proof. In Appendix D, we argue that a standard reduction from #MAXCUT already implies that the problem #3-TERMINAL MINCUT requires time $\exp(\Omega(m))$ under #ETH.

Name #3-TERMINAL MINCUT

Input Simple undirected graph $G = (V, E)$ with three distinguished vertices (“terminals”) $t_1, t_2, t_3 \in V$.

Output The number of edge subsets $A \subseteq E$ of minimal size that separate t_1 from t_2 , t_2 from t_3 , and t_3 from t_1 .

We reduce this problem to the problem of evaluating the coefficients of Z_0 at $q \notin \{1, 2\}$. Suppose $G' = (V, E, t_1, t_2, t_3)$ is an instance of #3-TERMINAL MINCUT with $n = |V|$ and $m = |E|$. We can assume that G' is simple and connected. We modify G' by adding a triangle between the terminals, obtaining the graph $G = (V, E \cup T)$ where $T = \{t_1t_2, t_2t_3, t_1t_3\}$; note that $n(G) = n$, $m(G) = m + 3$, and $|T| = 3$.

We focus our attention on the family \mathcal{A} of edge subsets $A \subseteq E$ for which t_1, t_2 , and t_3 each belong to a distinct component in the graph (V, A) . In other words, A belongs to \mathcal{A} if and only if $E - A$ is a 3-terminal cut in G' . Then we can split the sum in (5.5) into

$$Z_0(G; q, \mathbf{w}) = \sum_{B \subseteq T} \left(\sum_{A \in \mathcal{A}} q^{k(A \cup B) - 1} w^{|A|} (-1)^{|B|} + \sum_{A \notin \mathcal{A}} q^{k(A \cup B) - 1} w^{|A|} (-1)^{|B|} \right). \quad (5.6)$$

We first show that the second term of (5.6) vanishes. Consider an edge subset $A \notin \mathcal{A}$ and assume without loss of generality that it connects the terminals t_1 and t_2 . Consider $B \subseteq T$, and let $B' = B \oplus \{t_1t_2\}$, so that B' is the same as B except for t_1t_2 . Then the contributions of $A \cup B$ and $A \cup B'$ cancel: First, $k(A \cup B)$ equals $k(A \cup B')$ because t_1 and t_2 are connected through A already, so the presence or absence of the edge t_1t_2 makes no difference. Second, $(-1)^{|B|}$ equals $-(-1)^{|B'|}$.

We proceed to simplify the first term of (5.6). The edges in B only ever connect vertices in T , and for $A \in \mathcal{A}$, each of these lies in a separate component of (V, A) , so

$$k(A \cup B) = \begin{cases} k(A) - |B|, & \text{if } |B| = 0, 1, 2, \\ k(A) - 2, & \text{if } |B| = 3. \end{cases}$$

Calculating the contribution of B for each size $|B|$, we arrive at

$$\sum_{B \subseteq T} \sum_{A \in \mathcal{A}} q^{k(A \cup B) - 1} w^{|A|} (-1)^{|B|} = \sum_{A \in \mathcal{A}} q^{k(A) - 1} (q^0 - 3q^{-1} + 3q^{-2} - q^{-2}) w^{|A|},$$

and after some simplification we can write (5.6) as

$$Z_0(G; q, \mathbf{w}) = Q \cdot \sum_{A \in \mathcal{A}} q^{k(A) - 3} w^{|A|}, \quad \text{where } Q = (q - 1)(q - 2). \quad (5.7)$$

Note that, by assumption on q , we have $Q \neq 0$.

Let us write $\sum_{i=0}^m d_i w^i = Q^{-1} Z_0(G; q, \mathbf{w})$, i.e., d_i is the coefficient of the monomial w^i

5. Exponential Time Counting Complexity

in the sum above. More specifically,

$$Q \cdot d_i = \sum_{A \in \mathcal{A} : |A|=i} q^{k(A)-3}.$$

The edge subsets $A \in \mathcal{A}$ are exactly the complements of the 3-terminal cuts in G' . Now consider the family \mathcal{C} of *minimal* 3-terminal cuts, all of size c . The sets $E - A$ in \mathcal{C} are exactly the sets A of size $m - c$ in \mathcal{A} , and by minimality, $k(A) = 3$. Thus,

$$Q \cdot d_{m-c} = \sum_{A \in \mathcal{A} : |A|=m-c} q^{3-3} = |\mathcal{C}|.$$

Thus, if we could compute the coefficients d_0, \dots, d_m of $w \mapsto Q^{-1}Z_0(G; q, \mathbf{w})$, then we could determine the smallest c so that $d_{m-c} \neq 0$ and return $d_{m-c} = |\mathcal{C}|/Q$, the number of 3-terminal mincuts. \blacksquare

General Hyperbolas

We want to use Lemma 5.1 to show that computing the coefficients of the univariate Tutte polynomial at any fixed $q \notin \{1, 2\}$ is hard. For this, we need to get rid of negative weights and reduce to a single-valued weight function. In [GJ08], this is done by thickening and stretches, which we have to avoid. Since the number of edges with a negative weight is small (in fact, 3), we can use another tool: deletion-contraction.

A *deletion-contraction* identity expresses a function of the graph G in terms of two graphs $G - e$ and G/e , where $G - e$ arises from G by *deleting* the edge e ($\triangleleft \mapsto \triangleleft$) and G/e arises from G by *contracting* the edge e ($\triangleleft \mapsto \infty$) that is, deleting it and identifying its endpoints (so any remaining edges between these two endpoints become loops).

It is known [Sok05, eq. (4.6)] that

$$Z(G; q, \mathbf{w}) = Z(G - e; q, \mathbf{w}) + \mathbf{w}(e)Z(G/e; q, \mathbf{w}).$$

An edge e is a *bridge* of G if deleting e from G increases the number of connected components. The above gives a deletion-contraction identity for Z_0 as well:

$$Z_0(G; q, \mathbf{w}) = \begin{cases} qZ_0(G - e; q, \mathbf{w}) + \mathbf{w}(e)Z_0(G/e; q, \mathbf{w}) & \text{if } e \text{ is a bridge,} \\ Z_0(G - e; q, \mathbf{w}) + \mathbf{w}(e)Z_0(G/e; q, \mathbf{w}) & \text{otherwise.} \end{cases} \quad (5.8)$$

Proposition 5.2. *Let q be a rational number with $q \notin \{1, 2\}$. Computing the coefficients of the polynomial $v \mapsto Z_0(G; q, v)$ for a given simple graph G requires time $\exp(\Omega(m))$ under #ETH.*

By (5.3), this proposition also holds for Z instead of Z_0 when $q \notin \{0, 1, 2\}$.

Proof. Let $G = (V, E)$ be a graph as in the previous lemma, with three edges $T = \{e_1, e_2, e_3\}$ of weight -1 . The given reduction actually uses the restriction that $G' =$

$(V, E \setminus T)$ is connected, so we can assume that this is the case. Thus, none of the T -edges is a bridge, so three applications of (5.8) to delete and contract these edges, gives

$$Z_0(G; q, \mathbf{w}) = \sum_{C \subseteq \{1,2,3\}} (-1)^{|C|} Z_0(G_C; q, \mathbf{w}), \quad (5.9)$$

where for each $C \subseteq \{1, 2, 3\}$, the graph G_C is constructed from G by removing e_1, e_2, e_3 as follows: If $i \in C$ then e_i is contracted, otherwise it is deleted. In any case, the edges of T have disappeared and remaining edges of G_C are in one-to-one correspondence with the edges in E ; especially, they all have the same weight w , so $Z_0(G_C; q, \mathbf{w}) = Z_0(G_C; q, w)$.

The resulting G_C are not necessarily simple, because the contracted edges from T may have been part of a triangle and may have produced a loop. (In fact, investigating the details of the previous lemma, we can see that this is indeed the case.) Thus we construct the simple graph G'_C from G_C by subdividing every edge into a 3-path. This operation, known as a 3-stretch, is known to largely preserve the value of Z and Z_0 (see [Sok04] for the former and [GJ08] for the latter). In particular,

$$Z_0(G_C; q, w) = f(q, w')^m \cdot Z_0(G'_C; q, w'),$$

where for $q \neq 0$

$$1 + \frac{q}{w} = \left(1 + \frac{q}{w'}\right)^3 \quad \text{and} \quad f(q, w') = q^{-1} \cdot ((q + w')^3 - w'^3),$$

and for $q = 0$

$$w = w'/3 \quad \text{and} \quad f(q, w') = 1/(3w'^2).$$

In summary, to compute the coefficients of the polynomial $w \mapsto Z_0(G; q, \mathbf{w})$, we need to compute the 8 polynomials $v \mapsto Z_0(G_C; q, v)$, one for each G_C . We use the above equation and the assumed oracle for simple graphs to do this. We note that every G'_C is simple and has at most $n + m$ vertices and at most $2m$ edges. ■

5.3.2. Individual Points for Multigraphs

If we allow graphs to have multiple edges, we can use thickening and interpolation, one of the original strategies of [JVW90], for relocating the hardness result for hyperbolas from Proposition 5.1 and Proposition 5.2 to individual points in the Tutte plane. For most points, this gives us tight bounds in terms of n , the number of vertices, but not for points with $y \in \{0, \pm 1\}$, where thickening fails completely.

We recall the thickening identities for the Tutte polynomial. The k -thickening of G is the graph G_k in which all edges have been replaced by k parallel edges. One can show [Sok05, (4.21)] that, with $w_k = (1 + w)^k - 1$,

$$Z(G; q, w_k) = Z(G_k; q, w). \quad (5.10)$$

It is easy to transfer this result to the Tutte polynomial T using (5.1), yielding special cases of Brylawski's well-known graph transformation rules.

5. Exponential Time Counting Complexity

We use interpolation and obtain Theorem 1.11 (i) for $y \neq 0$ from the following.

Proposition 5.3. *Let $(q, w) \in \mathbb{Q}^2$ with $w \notin \{0, -1, -2\}$ and $q \neq 1$. Computing $Z(G; q, w)$ for a given graph G (not necessarily simple) requires time $\exp(\Omega(n))$ under #ETH.*

Proof. We observe that the values $w_k = (1 + w)^k - 1$ are all distinct for $k = 0, 1, \dots, m$. Thus, the k -thickenings G_k of G give rise to $m + 1$ different weight shifts, the evaluations of which, $Z(G; q, w_k)$, can be obtained from $Z(G_k; q, w)$ using (5.10). Thus, with oracle access to $G' \mapsto Z(G'; q, w)$, we can compute the coefficients of the polynomial $v \mapsto Z(G; q, v)$ in polynomial time for any given G . By Proposition 5.1 and Proposition 5.2, doing so requires time $\exp(\Omega(n))$ under #ETH. Since the number of vertices is n in each G_k , computing $G' \mapsto Z(G'; q, w)$ requires time $\exp(\Omega(n))$ under #ETH. ■

The proof of Theorem 1.11 (ii) uses Linial's well-known reduction for the chromatic polynomial [Lin86], and is deferred to Proposition D.1 in Appendix D.

5.3.3. Individual Points for Simple Graphs

In this section we show that most points (x, y) of the Tutte plane are as hard as the entire hyperbola on which they lie, even for sparse, simple graphs. The drawback of our method is that we lose a polylogarithmic factor in the exponent of the lower bound. The results are particularly interesting for the points on the line $y = -1$, for which we know no other good exponential lower bounds under #ETH, even in more general graph classes. We remark that the points $(-1, -1)$, $(0, -1)$, and $(\frac{1}{2}, -1)$ on this line are known to admit a polynomial-time algorithm, and indeed our hardness result does not apply here.

Graph inflations

We use the graph theoretic version of Brylawski's tensor product for matroids [Bry82]. We found the following terminology more intuitive in our setting.

Definition 5.1 (Graph inflation). *Let H be a 2-terminal undirected graph. For any undirected graph $G = (V, E)$, an H -inflation of G , denoted $G \otimes H$, is obtained by replacing every edge $xy \in E$ by (a fresh copy of) H , identifying x with one of the terminals of H and y with the other.*

If H is not symmetric with respect to its two terminals, then the graph $G \otimes H$ need not be unique since there are in general two non-isomorphic ways to replace an edge xy by H . For us this difference does not matter since the resulting Tutte polynomials turn out to be the same; in fact, in any graph one can remove a maximal biconnected component and reinsert it in the other direction without changing the Tutte polynomial, an operation that is called the *Whitney twist*. Thus we choose $G \otimes H$ arbitrarily among the graphs that satisfy the condition in the definition above. Graph inflation is not commutative and Sokal uses the notation \vec{G}^H .

5. Exponential Time Counting Complexity

(ii) $w_{S_i} \neq w_{S_j}$ for all $i \neq j$.

Furthermore, the sets S_i can be computed in time polynomial in m .

Proof. Let $b = |1 + q/w|$ and $f(s) = 1 + q/(b^s - 1)$ for $s > 0$. Our choice of parameters ensures that $b > 0$ and $b \neq 1$, so f is a well-defined, continuous, and strictly monotone function from $\mathbb{R}^+ \rightarrow \mathbb{R}$. Furthermore, $w_S = -1 + \prod_{s \in S} f(s)$ for all finite sets S of positive even integers. Now let $s_0 \geq 2$ be an even integer such that $f(s)$ is nonzero and has the same sign as $f(s_0)$ for all $s \geq s_0$. For $i = 0, \dots, m$, let $b_\ell \cdots b_0$ denote the binary expansion of i where $\ell = \lfloor \log m \rfloor$. Let $\Delta > 6$ be a gap parameter that is an even integer and large, but only depends on q and w and is chosen later. We define

$$S_i = \{ s_0 + \Delta \lfloor \log m \rfloor \cdot (2j + b_j) : 0 \leq j \leq \ell \}.$$

The salient feature of this construction is that all sets S_i are different, of equal small cardinality, contain only positive even integers, and are from a range where f does not change sign. Most important for our analysis is that the elements of the S_i are spaced apart significantly, i.e.,

$$\text{for } i, j \text{ and any } s \in S_i \text{ and } t \in S_j, \text{ either } s = t \text{ or } |s - t| \geq \Delta \log m. \quad (\text{P})$$

From $|S_i| = \lfloor \log m \rfloor + 1$ and the fact that all numbers in the sets are bounded by $O(\log^2 m)$, we immediately get (i).

To establish (ii), let $0 \leq i < j \leq m$. We want to show that $w_{S_i} \neq w_{S_j}$. Let us define $S = S_i \setminus S_j$ and $T = S_j \setminus S_i$. From (5.12), we see by multiplying with $(w_{S_i \cap S_j} + 1)$ on both sides that $w_S + 1 = w_T + 1$ is equivalent to $w_{S_i} = w_{S_j}$ since $w_{S_i \cap S_j} \neq -1$.

It remains to show that $\prod_{s \in S} f(s) \neq \prod_{t \in T} f(t)$. Equivalently,

$$\prod_{s \in S} (b^s + q - 1) \prod_{t \in T} (b^t - 1) - \prod_{t \in T} (b^t + q - 1) \prod_{s \in S} (b^s - 1) \neq 0 \quad (5.13)$$

We will factor out the products in (5.13). Using the notation $\|X\| = \sum_{x \in X} x$, we rewrite

$$\prod_{s \in S} (b^s + q - 1) \prod_{t \in T} (b^t - 1) = \sum_{X \subseteq S \cup T} (-1)^{|T \setminus X|} (q - 1)^{|S \setminus X|} b^{\|X\|}.$$

Here we use the convention that for $X \subseteq S \cup T$, the term b^s is taken in the first factor if $s \in X \cap S$, and b^t is taken in the second factor if $t \in X \cap T$. Doing this for both terms of (5.13) and collecting terms we arrive at the equivalent claim

$$\sum_{X \subseteq S \cup T} g(X) \neq 0, \quad (5.14)$$

where

$$g(X) = \left((-1)^{|T \setminus X|} (q - 1)^{|S \setminus X|} - (-1)^{|S \setminus X|} (q - 1)^{|T \setminus X|} \right) \cdot b^{\|X\|}. \quad (5.15)$$

Let s_1 be the smallest element of $S \cup T$ and without loss of generality assume that $s_1 \in S$

(otherwise exchange S and T). Now from (5.15) and $|S| = |T|$, it follows that

$$\begin{aligned} g(S \cup T) &= g(\emptyset) = 0 \\ g((S \cup T) \setminus \{s_1\}) &= q \cdot b^{\|S \cup T\| - s_1} \\ g(\{s_1\}) &= (-q) \cdot (1 - q)^{|S| - 1} \cdot b^{s_1}. \end{aligned}$$

The largest exponent of b with nonzero coefficient in (5.15) is $\|S \cup T\| - s_1$ and all other exponents are at least $\Delta \log m$ smaller than that. Similarly, the smallest exponent of b with nonzero coefficient is s_1 and all other exponents are at least $\Delta \log m$ larger. We will let X_0 denote the term with the largest contribution in (5.14); so we set $X_0 = S \cup T \setminus \{s_1\}$ for $b > 1$ and $X_0 = \{s_1\}$ for $b < 1$.

The total contribution of the remaining terms is $h = \sum_{X \neq X_0} g(X)$. We prove (5.14) by showing $|h| < |g(X_0)|$. From the triangle inequality and the fact that $S \cup T$ has at most $4m^2$ subsets X , we get

$$|h| \leq 4m^2 \cdot \max_{X \neq X_0} |g(X)| \leq 4m^2 \cdot 2|q - 1|^{1 + \log m} \cdot b^{\|X_0\| \pm \Delta \log m}$$

where the sign in $\pm \Delta \log m$ depends on whether b is larger or smaller than 1. If $b > 1$, the sign is negative. In this case, notice that $\Delta = \Delta(q, w)$ can be chosen so that $4m^2 \cdot 2|q - 1|^{1 + \log m} < |q| \cdot b^{\Delta \log m}$ for all $m \geq 2$. If $b < 1$, we can similarly choose Δ as to satisfy $4m^2 \cdot 2|q - 1|^{1 + \log m} < |q| \cdot |1 - q|^{|S| - 1} \cdot b^{-\Delta \log m}$. Thus, in both cases we have $|h| < |g(X_0)|$, establishing (ii). ■

Points on the Hyperbolas

We show Theorem 1.11 (iv), that evaluating Z at most points (q, w) with $q \notin \{0, 1\}$ is hard.

Proposition 5.4. *Let $(q, w) \in \mathbb{Q}^2 \setminus \{(4, -2), (2, -1), (2, -2)\}$ with $q \notin \{0, 1\}$ and $w \neq 0$. Computing $Z(G; q, w)$ for a given simple graph G requires time $\exp(\Omega(m / \log^3 m))$ under #ETH.*

By (5.1), the points $(4, -2)$, $(2, -1)$, and $(2, -2)$ in the (q, w) -plane correspond to the polynomial-time computable points $(-1, -1)$, $(-1, 0)$, and $(0, -1)$ in the (x, y) -plane.

Proof. We reduce from the problem of computing the coefficients of the polynomial $v \mapsto Z(G; q, v)$, which requires time $\exp(\Omega(m))$ for $q \notin \{0, 1\}$ by Proposition 5.1 and Proposition 5.2. We interpolate as in the proof of Proposition 5.3, but instead of thickenings we use Theta inflations to keep the number of edges relatively small.

First we consider the degenerate case in which $q = -w$ or $q = -2w$. For a positive integer constant k , let G' be the k -thickening of G . This transformation shifts the weight to w' with

$$w' = (1 + w)^k - 1,$$

which allows us to compute $Z(G; q, w')$ from $Z(G'; q, w)$ using (5.10). In the case $q = -w$, we have $1 + w = 1 - q$, which cannot be 1 or 0, but which can also not be -1 since then

5. Exponential Time Counting Complexity

$(q, w) = (2, -2)$. Similarly, in the case $q = -2w$, we have $1 + w = 1 - q/2$, which cannot be 1. It can also not be 0 since then $(q, w) = (2, -1)$, neither can it be -1 since then $(q, w) = (4, -2)$. Thus, in any case, $(1 + w) \notin \{0, \pm 1\}$. This means that we can choose k large enough so that $q \notin \{-w', -2w'\}$. This remains true if we let G'' be the 2-stretch to G' , which shifts the weight to w'' with

$$1 + \frac{q}{w''} = \left(1 + \frac{q}{w'}\right)^2,$$

so that $Z(G; q, w'')$ can be computed from $Z(G''; q, w)$ (see [Sok04]). We choose k so that $q \notin \{-w'', -2w''\}$. The graph G'' after this transformation is simple and the number of edges is only increased by a constant factor of $2k$.

By the above, we can assume w.l.o.g. that $q \notin \{-w, -2w\}$. We observe that the conditions $w \neq 0$ and $q \notin \{0, -w, -2w\}$ of Lemma 5.3 hold, and thus we can compute $m+1$ sets S_0, S_1, \dots, S_m with all distinct weight shifts w_0, \dots, w_m under Theta inflations.

For a given graph G , let $G_i = G \otimes_{\Theta_{S_i}}$. Using Lemma 5.2, we can compute the values $Z(G; q; w_i)$ from $Z(G_i; q, w)$. Moreover, as is clear from (5.2), the function $v \mapsto Z(G; q, v)$ is a polynomial of degree at most m , so we can use interpolation to recover its coefficients. We remark that the G_i are simple graphs with at most $O(m \log^3 m)$ edges, so the claim follows. \blacksquare

Bounce Graphs

The *reliability line* of the Tutte plane, i.e., the line $x = 1$, is not covered by the above since here $q = 0$ holds. On this line, the Tutte polynomial specializes to the *reliability polynomial* $R(G; p)$ (with $p = 1/y$), an object studied in algebraic graph theory [GR01, Section 15.8]. Given a connected graph G and a probability p , $R(G; p)$ is the probability that G stays connected if every edge independently fails with probability p . For example $R(\text{triangle}; \frac{1}{3}) = Pr(\text{triangle}) + 5Pr(\text{path}) = (\frac{2}{3})^5 + 5 \cdot \frac{1}{3} \cdot (\frac{2}{3})^4 = \frac{112}{243}$. Note that $R(G; 1) = 0$ for all connected graphs, so $p = 1$ is easy to evaluate – as it should be since it corresponds to the polynomial-time solvable point $(1, 1)$ in the Tutte plane.

Along the reliability line, weight shift identities take a different form. Using deletion–contraction identities we obtain the following rules, which are simple multi-weighted generalizations of [GJ08, Section 4.3].

Lemma 5.4. *Let G be a graph with edge weights given by $\mathbf{w} : E(G) \rightarrow \mathbb{Q}$.*

If $\varphi(G)$ is obtained from G by replacing a single edge $e \in E$ with a simple path of k edges $P = \{e_1, \dots, e_k\}$ with $\mathbf{w}(e_i) = w_i$, then

$$Z_0(\varphi(G); 0, \mathbf{w}) = C_P \cdot Z_0(G; 0, \mathbf{w}[e \mapsto w']),$$

where

$$\frac{1}{w'} = \frac{1}{w_1} + \dots + \frac{1}{w_k} \quad \text{and} \quad C_P = \frac{1}{w'} \prod_{i=1}^k w_i.$$

Here $\mathbf{w}[e \mapsto w']$ denotes the function $\mathbf{w}' : E(G) \rightarrow \mathbb{Q}$ that is identical to \mathbf{w} except at the point e where it is $\mathbf{w}'(e) = w'$.

Lemma 5.5. *If $\varphi(G)$ is obtained from G by replacing a single edge $e \in E$ with a bundle of parallel edges $B = \{e_1, \dots, e_k\}$ with $\mathbf{w}(e_i) = w_i$, then*

$$Z_0(\varphi(G); 0, \mathbf{w}) = Z_0(G; 0, \mathbf{w}[e \mapsto w']),$$

where

$$w' = -1 + \prod_{i=1}^k (1 + w_i).$$

Corollary 5.1. *If $\varphi(G)$ is obtained from G by replacing a single edge $e \in E$ with a simple path of k edges of constant weight w , then*

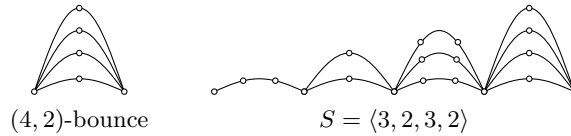
$$Z_0(\varphi(G); 0, \mathbf{w}) = kw^{k-1} \cdot Z_0(G; 0, \mathbf{w}[e \mapsto w/k]), \quad (5.16)$$

and if it is obtained from G by replacing $e \in E$ with a bundle of k parallel edges of constant weight w , then

$$Z_0(\varphi(G); 0, \mathbf{w}) = Z_0(G; 0, \mathbf{w}[e \mapsto (1 + w)^k - 1]). \quad (5.17)$$

These rules are transitive [GJ08, Lemma 1], and so can be freely combined for more intricate weight shifts. We define a class of graph inflations, *Bounce inflations*, and use the above to show that they give rise to distinct weight shifts along the reliability line of the Tutte polynomial. Bounce inflations are mildly inspired by l -byte numbers, in the sense that each has associated to it a sequence of length l , such that the lexicographic order of these sequences determines the size of the corresponding (shifted) weights.

Definition 5.2 (Bounce graph). *For positive integers i (height) and s (width), an (i, s) -bounce is the graph obtained by identifying all the left and all the right endpoints of i simple paths of length s each. Given a sequence $S = \langle s_1, s_2, \dots, s_l \rangle$ of l positive integers, the Bounce graph B_S is the graph obtained by concatenating l bounces at their endpoints, where the i -th bounce is an (i, s_i) -bounce, i.e., its height is i and its width is s_i .*



The number l is the length of the Bounce graph B_S .

Inflating a graph by a Bounce graph shifts the weights on the reliability line as follows.

5. Exponential Time Counting Complexity

Lemma 5.6. *For any graph G with m edges, any sequence $S = \langle s_1, s_2, \dots, s_l \rangle$ of positive integers, and any non-zero rational number w , we have*

$$Z_0(G \otimes B_S; 0, w) = C_S^m \cdot Z_0(G; 0, w_S),$$

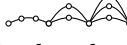

where

$$\frac{1}{w_S} = \sum_{i=1}^l \frac{1}{(1 + w/s_i)^i - 1} \quad \text{and} \quad C_S = \frac{1}{w_S} \cdot \prod_{i=1}^l w^{(s_i-1)i} \left((w + s_i)^i - s_i^i \right).$$

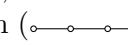
Proof. We start with $G \otimes B_S$ and consider the effect that replacing one of the m canonical copies of B_S with a single edge e has. We show that, with φ denoting this operation,

$$Z_0(G \otimes B_S; 0, w) = C_S \cdot Z_0(\varphi(G \otimes B_S); 0, \mathbf{w}[e \mapsto w_S]), \quad (5.18)$$

where w_S has the above form, and \mathbf{w} has the old value w on all unaffected edges. The lemma then follows by successively applying φ to each canonical copy of B_S in $G \otimes B_S$.

The first step towards transforming a Bounce graph (say, ) into a single edge, consists of contracting the paths of the bounces to a single edge each. For the i -th bounce, this is just the inverse of an s_i -stretching applied to each of the i paths. By (5.16) of Corollary 5.1, this 'unstretching' gives a factor $(s_i w^{s_i-1})^i$ to the polynomial, and each edge in the resulting $(i, 1)$ -bounce receives a weight of w/s_i in the modified graph. Repeating this process for every bounce simplifies the Bounce graph into a Bounce graph of length l that is generated by a sequence of 1s (). Let $\phi(G \otimes B_S)$ denote the graph in which one Bounce graphs has been simplified. By transitivity, we have the weight shift

$$Z_0(G \otimes B_S; 0, w) = \left(\prod_{i=1}^l (s_i w^{s_i-1})^i \right) \cdot Z_0(\phi(G \otimes B_S); 0, \mathbf{w}'),$$

where \mathbf{w}' takes the value w/s_i on every edge of the i th bounce of the simplified Bounce graph, and the old value w outside the simplified Bounce graph. Next, we successively replace each of its $(i, 1)$ -bounces by a single edge to get a simple path () of length l . This transformation is just an 'unthickening' of each $(i, 1)$ -bounce, and from (5.17) of Corollary 5.1 we know that it does not produce any new factors for the polynomial, but the weight of the i th edge in this path becomes

$$w_i = (1 + w/s_i)^i - 1.$$

Finally, we compress the path into a single edge e . Then the claim in (5.18) follows by a single application of Lemma 5.4. ■

We now show that Bounce inflations provide a rich enough class of weight shifts. The ranges of w for which we prove this is general enough to allow for interpolation on the whole reliability line, and we make no attempt at extending the ranges. The proof for

$w > 9$ is due to Husfeldt and Taslaman [HT10].

Lemma 5.7. *Let w be a rational number with $w \in (-1, 0)$ or $w \in (9, \infty)$. For all integers $m \geq 1$, there exist sequences S_0, \dots, S_m of positive integers such that*

(i) $|E(B_{S_i})| \leq O(\log^2 m)$ for all i , and

(ii) $w_{S_i} \neq w_{S_j}$ for all $i \neq j$.

Furthermore, the sequences S_i can be computed in time polynomial in m .

Proof. We consider the set of sequences $S = \langle s_1, \dots, s_l \rangle$ of length $l = r \log(m+1)$, with $s_i \in \{2, 3\}$ for all i which are positive integer multiples of r , and $s_i = 2$ for all other i . Here r is a positive integer and will be chosen later, only depending on w . Since r is a constant, this construction satisfies (i).

Now consider any two distinct sequences $S = \langle s_i \rangle$ and $T = \langle t_i \rangle$. To show (ii), we consider the difference

$$\Delta = \frac{1}{w_S} - \frac{1}{w_T},$$

and show that $\Delta \neq 0$.

Using Lemma 5.6 we get a sum expression for Δ .

$$\begin{aligned} \Delta &= \sum_{i=1}^l \frac{1}{(1+w/s_i)^i - 1} - \sum_{i=1}^l \frac{1}{(1+w/t_i)^i - 1} \\ &= \sum_{i=1}^l g\left((1+w/s_i)^i\right) - \sum_{i=1}^l g\left((1+w/t_i)^i\right), \end{aligned} \tag{5.19}$$

where g is the function $g(x) = \frac{1}{x-1}$. This function is negative and strictly decreasing on $(0, 1)$ and positive and strictly decreasing on $(1, \infty)$. It is convenient to set $a, b \in \{(1+w/3), (1+w/2)\}$ so that $a < b$. By the monotonicity of g , we have $g(a^i) > g(b^i)$ for all positive i .

Case 1: $w > 9$. Here we have $a = (1+w/3)$ and $b = (1+w/2)$. We set $r = 1$ and let k be the smallest index for which the sequences differ, i.e., $s_k \neq t_k$. We assume w.l.o.g. that $s_k = 3$ and $t_k = 2$, otherwise we exchange the roles of S and T . In (5.19), terms of the sum for $i < k$ cancel. The terms corresponding to $i = k$ are $g(a^k) - g(b^k) > 0$. We apply the monotonicity of g to the terms for $i > k$, which allows us to lower bound Δ as follows.

$$\Delta \geq g(a^k) + \sum_{i=k+1}^l g(b^i) - g(b^k) - \sum_{i=k+1}^l g(a^i) = f(a) - f(b),$$

where

$$f(x) = g(x^k) - \sum_{i=k+1}^l g(x^i) = \frac{1}{x^k - 1} - \sum_{i=k+1}^l \frac{1}{x^i - 1}. \tag{5.20}$$

We now claim that f is strictly decreasing in $(4, \infty)$. This implies that $\Delta > 0$ since $w > 9$ guarantees $a, b > 4$, and we get $\Delta \geq f(a) - f(b) > 0$. To prove the claim, we

5. Exponential Time Counting Complexity

show that the derivative of f is negative on $(4, \infty)$:

$$f'(x) = -\frac{kx^{k-1}}{(x^k - 1)^2} + \sum_{i=k+1}^l \frac{ix^{i-1}}{(x^i - 1)^2}. \quad (5.21)$$

The terms of the sum here, let us call them $T_i(x)$, satisfy

$$T_i(x) > 2 \cdot T_{i+1}(x)$$

for all i and all $x > 4$. To see this, note that the inequality is equivalent to

$$2 \left(1 + \frac{1}{i}\right) x < \left(x + \frac{x-1}{x^i - 1}\right)^2.$$

This statement is true for all reals $x > 4$ and all positive integers i since then we have that $\text{LHS} \leq 4x < x^2 \leq \text{RHS}$. Thus, for $x > 4$, we have

$$f'(x) < \frac{kx^{k-1}}{(x^k - 1)^2} \left(-1 + \sum_{i=k+1}^l \frac{1}{2^{i-k}}\right) < 0,$$

which suffices to prove the claim.

Case 2: $w \in (-1, 0)$. Here we have $a = (1 + w/2)$ and $b = (1 + w/3)$. We choose r to be a positive integer that satisfies $b^r < \frac{1}{4}$. Let rk be the smallest index for which the sequences differ, i.e., $s_{rk} \neq t_{rk}$. We assume w.l.o.g. that $s_{rk} = 3$ and $t_{rk} = 2$, otherwise we exchange the roles of S and T . In (5.19), terms of the sum for $i < rk$ cancel, and so do terms for those i 's which are not integer multiples of r . The terms corresponding to $i = rk$ are $g(b^{rk}) - g(a^{rk}) < 0$. We apply the monotonicity of g to the remaining terms for $i > rk$, which allows us to upper bound Δ as follows.

$$\Delta \leq g(b^{rk}) + \sum_{i=k+1}^{l/r} g(a^{ri}) - g(a^{rk}) - \sum_{i=k+1}^{l/r} g(b^{ri})$$

For $x \in (0, 1)$, we can expand $g(x)$ into the geometric series

$$g(x) = \frac{1}{x-1} = -\sum_{j=0}^{\infty} x^j.$$

Applying this representation to our estimate for Δ and rearranging terms, we arrive at

$$\Delta \leq \sum_{j=0}^{\infty} \left((a^{rj})^k - (b^{rj})^k + \sum_{i=k+1}^{l/r} \left((b^{rj})^i - (a^{rj})^i \right) \right) = \sum_{j=0}^{\infty} \left(F(a^{rj}) - F(b^{rj}) \right),$$

where F is the function

$$F(y) = y^k - \sum_{i=k+1}^{l/r} y^i.$$

We claim that F is strictly increasing on $(0, \frac{1}{4})$. This implies $\Delta < 0$ since the choice of r makes sure that a^{rj} and b^{rj} are in the range $(0, \frac{1}{4})$ for all positive integers j . Thus, since the term for $j = 0$ is 0 and $F(a^{rj}) - F(b^{rj}) < 0$ for $j \geq 1$, the claim indeed implies $\Delta < 0$.

It remains to argue the claim. We show that the derivative of F is positive.

$$F'(y) = ky^{k-1} - \sum_{i=k+1}^{l/r} iy^{i-1}.$$

We obtain $F'(y) > 0$ from the following calculation, using the fact that $y \in (0, \frac{1}{4})$.

$$\begin{aligned} (ky^{k-1})^{-1} \cdot \sum_{i=k+1}^{l/r} iy^{i-1} &= \sum_{i=k+1}^{l/r} \frac{i}{k} y^{i-k} = \sum_{i=1}^{l/r-k} \left(1 + \frac{i}{k}\right) y^i \\ &\leq \sum_{i=1}^{l/r-k} (1+i) y^i \leq \sum_{i=1}^{\infty} y^i + \sum_{i=1}^{\infty} iy^i \\ &= \frac{1}{1-y} - 1 + \frac{y}{(1-y)^2} \leq \frac{4}{3} - 1 + \frac{4}{9} < 1. \quad \blacksquare \end{aligned}$$

Points on the Reliability Line

We prove Theorem 1.11 (iii). For $w > 0$, this is due to Husfeldt and Taslamán [HT10].

Proposition 5.5. *Let w be a rational number with $w \neq 0$. Computing $Z_0(G; 0, w)$ for a given simple graph G requires time $\exp(\Omega(m/\log^2 m))$ under $\#ETH$.*

Proof. If $w < 0$, we can pick a positive integer k big enough such that

$$w' := w/k > -1.$$

This weight shift corresponds to the k -stretch of G (Corollary 5.1). On the other hand, if $w > 0$, we can pick a positive integer k such that

$$w' := (w/2 + 1)^k - 1 > 9.$$

This is the weight shift that corresponds to the 2-stretch of the k -thickening of G (Corollary 5.1). In any case we can compute $Z(G; w', q)$ from $Z(G'; w, q)$. The graph remains simple after any of these transformations, and the number of edges is only increased by a constant factor of at most $2k$.

By the above, we can assume w.l.o.g. that $w \in (-1, 0)$ or $w > 9$. Lemma 5.7 then allows us to construct $m+1$ bounce graphs B_S such that the corresponding weight shifts w_S are all distinct by property (ii). By Lemma 5.6, we can compute the values $Z_0(G; 0, w_S)$ from $Z_0(G \otimes B_S; 0, w)$, i.e., we get evaluations of $v \mapsto Z_0(G; 0, v)$ at $m+1$ distinct points. Since the degree of this polynomial is m , we obtain its coefficients by interpolation. By Proposition 5.2, evaluating these coefficients requires time $\exp(\Omega(m))$ under

5. Exponential Time Counting Complexity

#ETH. By Lemma 5.7 (i), each $G \otimes B_S$ has at most $O(m \log^2 m)$ edges, which implies that computing $Z_0(G; 0, w)$ for given G requires time $\exp(\Omega(m/\log^2 m))$ as claimed. ■

Notes

Results in this chapter are joint work with Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlén. Most results appeared in [DHW10], but the hardness of the Tutte polynomial at $x = 1$ and $y > 1$ is from [HT10].

6. Summary and Open Problems

In the first part of this thesis, we introduced a model of communication that captures various settings of interest in the theory of computing. For NP-complete problems like d -SAT, d -VERTEX COVER, d -CLIQUE, and d -SET MATCHING, we showed that trivial protocols are essentially optimal as function of the witness size, unless the polynomial-time hierarchy collapses. Under the hypothesis that the latter does not happen, the result implies tight lower bounds for parameters captured by the communication model, including the size of PCPs, and polynomial-time sparsification, kernelization, and lossy compression. Under stronger hypotheses similar results hold for larger time bounds.

Future directions include more applications with an active oracle, exploiting the full power of our oracle communication model; we presented some in Section 3.6. Another direction regards the extension to the randomized setting with false negatives, and with false positives as well as false negatives; we know how to handle false positives only. In light of the hardness results for OR-problems, it is natural to ask whether an analogue for AND-problems exists, and such a result would have consequences for the kernelizability of problems like computing the tree-width. Finally, can we relax the hypothesis $\text{coNP} \not\subseteq \text{NP/poly}$ to the minimal $\text{P} \neq \text{NP}$?

Our results for the Tutte polynomial leave open the line $y = 1$ except for the point $(1, 1)$, even in the case of multigraphs. That line corresponds to counting the number of forest weighted by the number of edges, i.e., $T(G; 1+1/w, 1) \sim F(G; w) = \sum_{\text{forests } F} w^{|F|}$. Thickening and Theta inflation with the analysis in the proof of Lemma 5.6 suffice to show that every point is as hard as computing the coefficients of $F(G; w)$ without increasing the number of vertices for multigraphs and with an increase in the number of edges by a factor of $O(\log^2 m)$ in the case of simple graphs. However, we do not know that computing those coefficients requires exponential time. And of course, it would be nice to improve our conditional lower bounds $\exp(\Omega(n/\text{poly log } n))$ to match the corresponding upper bounds $\exp(O(n))$.

A. Behrend's Construction

We now prove Lemma 4.1, following an elegant construction due to Behrend [Beh46], which improves on the original construction due to Salem and Spencer [SS42].

Lemma 4.1 (restated). *For every positive integer p there exists a subset $A \subseteq \mathbb{Z}_p$ of size at least $p^{1-o(1)}$ which contains no nontrivial arithmetic progressions of length three. Furthermore, such a set A can be determined in time polynomial in p .*

Proof. Let p be a positive integer. We want to construct a set $A \subseteq \mathbb{Z}_p$ of size $p^{1-o(1)}$ that contains no nontrivial arithmetic progressions of length three over \mathbb{Z}_p .

For positive integers d, m and a real r to be chosen later, let $S_r \subseteq \mathbb{R}^d$ denote the d -dimensional sphere of radius r restricted to vectors whose components are from \mathbb{Z}_m :

$$S_r = \left\{ (a_1, \dots, a_d) \in \mathbb{Z}_m^d \mid a_1^2 + \dots + a_d^2 = r^2 \right\}.$$

The midpoint between any two distinct points \vec{a} and \vec{b} on a sphere is not itself on the sphere. This means that

$$\vec{a} + \vec{b} \neq 2\vec{c} \quad \text{for all distinct } \vec{a}, \vec{b}, \vec{c} \in S_r. \quad (\text{A.1})$$

This is the type of property we need except that we want it for a subset of integers rather than vectors with integer coordinates. We can transform S_r into a set of integers and maintain (A.1) by applying a linear mapping $\langle \cdot \rangle : \mathbb{N}^d \rightarrow \mathbb{N}$ that is 1-to-1 on \mathbb{Z}_{2m-1}^d . Then the set $\langle S_r \rangle = \{ \langle \vec{a} \rangle \mid \vec{a} \in S_r \}$ satisfies

$$\langle \vec{a} \rangle + \langle \vec{b} \rangle = \langle \vec{a} + \vec{b} \rangle \neq \langle 2\vec{c} \rangle = 2\langle \vec{c} \rangle \quad \text{for all distinct } \langle \vec{a} \rangle, \langle \vec{b} \rangle, \langle \vec{c} \rangle \in \langle S_r \rangle. \quad (\text{A.2})$$

Moreover, if

$$\max_{\vec{a} \in \mathbb{Z}_{2m-1}^d} \langle \vec{a} \rangle < p \quad (\text{A.3})$$

then $\langle S_r \rangle \subseteq \mathbb{Z}_p$ and (A.2) implies that

$$\langle \vec{a} \rangle + \langle \vec{b} \rangle \not\equiv 2\langle \vec{c} \rangle \pmod{p} \quad \text{for all distinct } \langle \vec{a} \rangle, \langle \vec{b} \rangle, \langle \vec{c} \rangle \in \langle S_r \rangle.$$

That is, $\langle S_r \rangle \subseteq \mathbb{Z}_p$ contains no nontrivial arithmetic progressions of length three over \mathbb{Z}_p .

We define the function $\langle \cdot \rangle$ by interpreting a vector $\vec{a} = (a_1, \dots, a_d) \in \mathbb{Z}_{2m-1}^d$ as a d -digit number in base $2m-1$, i.e., $\langle \vec{a} \rangle = \sum_{i=1}^d a_i (2m-1)^{i-1}$. This yields a linear function from \mathbb{N}^d to \mathbb{N} which is 1-to-1 on \mathbb{Z}_{2m-1}^d and achieves a maximum value of $(2m-1)^d - 1$ on \mathbb{Z}_{2m-1}^d . Thus, (A.3) is satisfied if $(2m-1)^d \leq p$.

A. Behrend's Construction

It remains to choose d, r, m such that $(2m - 1)^d \leq p$ and $|\langle S_r \rangle| = |S_r| \geq p^{1-o(1)}$. For this, note that the sets S_r partition the set \mathbb{Z}_m^d . The number of r for which S_r has a non-empty intersection with \mathbb{Z}_m^d is less than dm^2 . By averaging, for each m there exists an r for which $|S_r| \geq |\mathbb{Z}_m^d|/(dm^2) = m^{d-2}/d$. Setting $d = \sqrt{\log p}$ and $m = 2^{\sqrt{\log p}-1}$ ensures that $(2m - 1)^d \leq p$ and that $m^{d-2}/d = (2^{\sqrt{\log p}-1})^{(\sqrt{\log p}-2)}/\sqrt{\log p} \geq p^{1-O(1/\sqrt{\log p})}$. We set r^* as the first r for which $|S_r| \geq m^{d-2}/d$. We can compute r^* and $\langle S_{r^*} \rangle$ in time polynomial in p . Thus, setting $A = \langle S_{r^*} \rangle$ satisfies all the requirements. ■

We point out that the construction in the proof of Lemma 4.1 guarantees that the cardinality of the set A is at least $p^{1-O(1/\sqrt{\log p})}$ rather than just $p^{1-o(1)}$. By considering a thin annulus rather than a sphere for the set S , Elkin [Elk10, GW08] recently further improved the cardinality by a factor of the form $\log^c p$ for some positive constant c . However, the analysis becomes more complicated and Behrend's already suffices for our application.

B. Sunflower Kernelization for Set Matching

We sketch a modern proof of Theorem 1.4, that d -SET MATCHING has kernels with $O(k^d)$ hyperedges.

Proof (Sketch). A *sunflower* with p petals is a set of p hyperedges whose pairwise intersections are equal. By the sunflower lemma, any d -uniform hypergraph G with more than $d! \cdot r^d$ edges has a sunflower with $r + 1$ petals. We set $r = dk$ and observe that, in any sunflower with $r + 1$ petals, we can arbitrarily choose an edge e of the sunflower and remove it from the graph. To see this, assume we have a matching M of G with k edges. If M does not contain e , then M is still a matching of size k in $G - e$. On the other hand, if M contains e , there must be a petal that does not intersect M since we have $dk + 1$ petals but M involves only dk vertices. Thus we can replace e in the matching by the edge that corresponds to that petal, and we obtain a matching of G' that consists of k hyperedges. This establishes the completeness of the reduction. The soundness is clear since any matching of G' is a matching of G . ■

C. The Sparsification Lemma

Sparsification is the process of reducing the density of graphs, formulas, or other combinatorial objects, while some properties of the objects like the answer to a computational problem are preserved. From an algorithmic perspective, efficient sparsification procedures can be used as kernelization algorithms to make input instances sparse and thus possibly simpler and smaller, such that only the core information about the input remains. From a complexity-theoretic point of view, sparsification is a tool to identify those instances of a problem that are computationally the hardest. If an NP-hard problem admits efficient sparsification, the hardest instances are sparse.

In the context of the exponential-time hypothesis, the sparsification lemma provides a way to show that the hardest instances of d -SAT are sparse and thus the parameter n can be replaced with m in the statement of the exponential-time hypothesis. The following is the sparsification lemma as formulated in [FG06, Lemma 16.17].

Lemma C.1 (Sparsification Lemma). *Let $d \geq 2$. There exists a computable function $f : \mathbb{N}^2 \rightarrow \mathbb{N}$ such that for every $k \in \mathbb{N}$ and every d -CNF formula γ with n variables, we can find a formula*

$$\beta = \bigvee_{i \in [t]} \gamma_i$$

such that:

- (1) β is equivalent to γ ,
- (2) $t \leq 2^{n/k}$ and
- (3) each γ_i is a subformula of γ in which each variable occurs at most $f(d, k)$ times.

Furthermore, β can be computed from γ and k in time $t \cdot \text{poly}(n)$.

We sketch below a small modification in the proof of the sparsification lemma that allows us to replace (1) with the condition

$$(1') \text{ sat}(\gamma) = \dot{\bigcup}_i \text{sat}(\gamma_i),$$

where $\text{sat}(\varphi)$ is the set of assignments that satisfy the formula φ . In particular, (1') implies $\#\text{SAT}(\gamma) = \sum_i \#\text{SAT}(\gamma_i)$, which means that the sparsification lemma can be used for the counting version of 3-SAT.

Proof (sketch). We adapt the terminology of [FG06, Proof of Lemma 16.17] and we follow their construction precisely, except for a small change in the sparsification algorithm.

C. The Sparsification Lemma

When the algorithm decides to branch for a CNF-formula γ and a flower $\alpha = \{\delta_1, \dots, \delta_p\}$, the original algorithm would branch on the two formulas

$$\begin{aligned}\gamma_{\text{heart}}^\alpha &= \gamma \setminus \{\delta_1, \dots, \delta_p\} \cup \{\delta\}, \\ \gamma_{\text{petals}}^\alpha &= \gamma \setminus \{\delta_1, \dots, \delta_p\} \cup \{\delta_1 \setminus \delta, \dots, \delta_p \setminus \delta\}.\end{aligned}$$

We modify the branching on the petals to read

$$\gamma_{\text{petals}}^\alpha = \gamma \setminus \{\delta_1, \dots, \delta_p\} \cup \{\delta_1 \setminus \delta, \dots, \delta_p \setminus \delta\} \cup \{ \{-l\} : l \in \delta \}.$$

This way, the satisfying assignments become disjoint: In each branching step, we guess whether the heart contains a literal set to true, or whether all literals in the heart are set to false and each petal contains a literals set to true.

Now we have that, for all CNF-formulas γ , all assignments σ to the variables of γ , and all flowers α of γ ,

- (i) σ satisfies γ if and only if σ satisfies $\gamma_{\text{heart}}^\alpha \vee \gamma_{\text{petals}}^\alpha$, and
- (ii) σ does not satisfy $\gamma_{\text{heart}}^\alpha$ or σ does not satisfy $\gamma_{\text{petals}}^\alpha$.

By induction, we see that at the end of the algorithm,

- (i) σ satisfies γ if and only if σ satisfies some γ_i , and
- (ii) σ satisfies at most one γ_i .

This implies that $\text{sat}(\gamma) = \dot{\bigcup}_{i \in [t]} \text{sat}(\gamma_i)$.

Notice that our new construction adds at most n clauses of size 1 to the formulas γ_i compared to the old one. Furthermore, our construction does not make t any larger because the REDUCE-step removes all clauses that properly contain $\{-l\}$ and thus these unit clauses never appear in a flower. ■

Proof (of Theorem 1.8). For all integers $d \geq 3$ and $k \geq 1$, the sparsification lemma gives an oracle reduction from $\#d$ -SAT to $\#d$ -SAT that, on input a formula γ with n variables, only queries formulas with $m' = O(n)$ clauses, such that the reduction runs in time $\exp(O(n/k))$. Now, if for every $c > 0$ there is an algorithm for $\#d$ -SAT that runs in time $\exp(cm)$, we can combine this algorithm and the above oracle reduction to obtain an algorithm for $\#d$ -SAT that runs in time $\exp(O(n/k) + c \cdot m') = \exp(O(n/k) + c \cdot O(n))$. Since this holds for all small $c > 0$ and large k , we have for every $c' > 0$ an algorithm for $\#d$ -SAT running in time $\exp(c' \cdot n)$. This proves that for all $d \geq 3$, $\#d$ -SAT can be solved in variable-subexponential time if and only if it can be solved in clause-subexponential time.

To establish the equivalence between different d 's, we transform an instance φ of $\#d$ -SAT into an instance φ' of $\#3$ -SAT that has the same number of satisfying assignments. The formula φ' is constructed as in the standard width-reduction for d -CNF formulas, i.e., by introducing a constant number of new variables for every clause of φ . Thus, since the number of clauses of φ' is $O(m)$, any clause-subexponential algorithm for $\#3$ -SAT implies a clause-subexponential algorithm for $\#d$ -SAT. ■

D. Hardness of 3-Colouring and 3-Terminal MinCut

The purpose of this section is to show that the standard reductions from 3-SAT to the computational problems 3-COLOURING, NAE-SAT, MAXCUT, and 3-TERMINAL MINCUT already preserve the number of solutions and increase the number of clauses or edges of the instances by at most a constant factor. This then implies that the corresponding counting problems require time exponential in the number of clauses or edges unless #ETH fails.

Theorem D.1. *Deterministically computing the problems #NAE-SAT, #MAXCUT, and #3-TERMINAL MINCUT requires time $\exp(\Omega(m))$ unless #ETH fails.*

In the following, we formally define the problems, sketch the standard NP-hardness reductions, and provide their analyses as needed to prove Theorem D.1.

Name #NAE-SAT

Input 3-CNF formula φ .

Output The number of truth assignments, so that no clause $\{a, b, c\} \in \varphi$ contains only literals with the same truth value.

Lemma D.1. *There is a polynomial-time reduction from #3-SAT to #NAE-SAT that maps formulas with m clauses to formulas with $O(m)$ clauses.*

Proof. Let φ be a 3-CNF formula with n variables and m clauses. To construct the instance φ' to NAE-SAT, we first replace every trivariate clause $(a \vee b \vee c)$ with the clauses

$$(x \vee \bar{a}) \wedge (x \vee \bar{b}) \wedge (\bar{x} \vee a \vee b) \wedge (x \vee c),$$

where x is a fresh variable. These clauses force x to have the value of $a \vee b$ in a satisfying assignment. It can be checked that these clauses are satisfied exactly if the original clause was satisfied and moreover that the trivariate clause is never all-false or all-true. In total, we increase the number of clauses four-fold.

Finally, introduce yet another fresh variable z and add this single variable (positively) to every mono- and bivariate clause. It is well-known that this modification turns the instance into an instance of NAE-SAT (see [Pap94, Theorem 3]). The resulting instance φ'' has $4m$ clauses and $\#\text{NAE-SAT}(\varphi'') = \#\text{3-SAT}(\varphi)$. ■

Name #MAXCUT

Input Simple undirected graph G .

D. Hardness of 3-Colouring and 3-Terminal MinCut

Output The number of maximum cuts.

A maximum cut is a set $C \subseteq V(G)$ that maximizes the number $|E(C, \overline{C})|$ of edges of G that cross the cut.

Lemma D.2. *There is a polynomial-time reduction from #NAE-SAT to #MAXCUT that maps formulas with m clauses to graphs with $O(m)$ edges.*

Proof. We follow the reduction in [Pap94, Theorem 9.5]. Given an instance to NAE-SAT with n variables and m clauses, the reduction produces an instance to MAXCUT, a graph with $2n$ vertices and at most $3m + 3m = 6m$ edges. Furthermore, the number of solutions is equal. ■

Name #3-TERMINAL MINCUT

Input Simple undirected graph $G = (V, E)$ with three distinguished vertices (“terminals”) $t_1, t_2, t_3 \in V$.

Output The number of cuts of minimal size that separate t_1 from t_2 , t_2 from t_3 , and t_3 from t_1 .

Lemma D.3. *There is a polynomial-time reduction from #MAXCUT to #3-TERMINAL MINCUT that maps graphs with m edges to graphs with $O(m)$ edges.*

Proof. We follow the reduction in [DJP⁺94]. So let $G = (V, E)$ be a graph with n vertices and m edges. It is made explicit in [DJP⁺94] that the construction builds a graph F with $n' = 3 + n + 4m = O(m)$ vertices. For the number of edges, every $uv \in E$ results in a gadget graph C with 18 edges, so the number of edges in F is $18m = O(m)$. The construction is such that the number of minimum 3-terminal cuts of F equals the number of maximum cuts of G . ■

Proof (of Theorem D.1). Assume one of the problems can be solved in time $\exp(cm)$ for every $c > 0$. Then 3-SAT can be solved by first applying the applicable reductions of the preceding lemmas and then invoking the assumed algorithm. This gives for every $c > 0$ an algorithm for 3-SAT that runs in time $\exp(O(cm))$, which implies that #ETH fails. ■

Hardness of Colouring and Other Individual Points on the Chromatic Line

Theorem 1.11 (ii) cannot be handled by the proof of Proposition 5.3 because thickenings do not produce enough points for the interpolation. Instead, we use Linial’s reduction [Lin86] along this line. For this, we need to observe that #3-COLOURING is hard under #ETH.

Name #3-COLOURING

Input Simple undirected graph G .

Output The number of proper vertex-colourings with three colours.

Lemma D.4. *There is a polynomial-time reduction from #NAE-SAT to #3-COLOURING that maps formulas with m clauses to graphs with $O(m)$ edges.*

Proof. The hardness of 3-COLOURING under ETH is already observed in [IPZ01] but without mentioning that it even holds if we use m instead of n to measure the size of the instance. For the counting variant, observe that the graph G that is constructed in [Pap94, Theorem 9.8] from an NAE-SAT-instance φ with n variables and m clauses has $n' = 1 + 2n + 3m$ vertices and $m' = 3n + 6m$ edges. Furthermore the number of proper 3-colourings is equal to $\#\text{NAE-SAT}(\varphi)$. ■

The chromatic polynomial $\chi(G; q)$ of G is the polynomial in q with the property that, for all $c \in \mathbb{N}$, the evaluation $\chi(G; c)$ is the number of proper c -colourings of the vertices of G . We write $\chi(q)$ for the function $G \mapsto \chi(G; q)$. The Tutte polynomial specializes to the chromatic polynomial for $y = 0$:

$$\chi(G; q) = (-1)^{n(G)-k(G)} q^{k(G)} T(G; 1 - q, 0). \quad (\text{D.1})$$

We now prove Theorem 1.11 (ii).

Proposition D.1. *Let $x \in \{-2, -3, \dots\}$.*

Then $\text{TUTTE}^{01}(x, 0)$ requires time $\exp(\Omega(m))$ under $\#ETH$.

Proof. Set $q = 1 - x$. We use (D.1) to see that evaluating the chromatic polynomial $\chi(q)$ is equivalent to evaluating $\text{TUTTE}(x, 0)$ if $q \neq 0$. Since $\chi(3)$ is the number of 3-colourings, the case $q = 3$ requires time $\exp(\Omega(m))$ under $\#ETH$, even for simple graphs (cf. App. D). For $i \in \{1, 2, \dots\}$ and all real r , Linial's identity is

$$\chi(G + K_i; r) = r(r - 1) \dots (r - i + 1) \cdot \chi(G; r - i), \quad (\text{D.2})$$

where $G + K_i$ is the simple graph consisting of G and a clique K_i on i vertices, each of which is adjacent to every vertex of G .

For $q \in \{4, 5, \dots\}$, we can set $i = q - 3$ and directly compute $\chi(G; 3) = \chi(G; q - i) = \chi(G + K_i; q) / [q(q - 1) \dots 4]$. Since $m(G + K_i) = m(G) + i \cdot n(G) + \binom{i}{2} \leq O(m(G))$, it follows that $\chi(q)$ requires time $\exp(\Omega(m))$ under $\#ETH$, even for simple graphs. ■

Proposition D.2. *Let $x \notin \mathbb{Q} \setminus \{1, 0, -1, -2, -3, \dots\}$.*

Then $\text{TUTTE}^{01}(x, 0)$ requires time $\exp(\Omega(n))$ under $\#ETH$.

Proof. Set $q = 1 - x$. We show that $\text{TUTTE}^{01}(x, 0)$ requires time $\exp(\Omega(n))$ under $\#ETH$. Indeed, with access to $\chi(q)$, we can compute $\chi(G; q - i)$ for all $i = 0, \dots, n$, noting that all prefactors in (D.2) nonzero. From these $n + 1$ values, we interpolate to get the coefficients of the polynomial $r \mapsto \chi(G; r)$, which in turn allows us evaluate $\chi(G; 3)$. In this case, the size of the oracle queries depends nonlinearly on the size of G , in particular $m(G + K_n) \sim n^2$. However, the number of vertices is $n(G + K_i) \leq 2n \leq O(m(G))$. Thus, since $\chi(3)$ requires time $\exp(\Omega(n))$ under $\#ETH$, this also holds for $\chi(q)$, even for simple graphs. ■

The only points on the x -axis not covered here are $x \in \{1, 0, -1\}$. Two of these admit polynomial time algorithms, so we expect no hardness result. By Theorem 1.11 (iii), evaluating the Tutte polynomial at the point $(1, 0)$ requires time $\exp(\Omega(m/\log^2 m))$ under $\#ETH$.

Bibliography

- [AB09] Arora, Sanjeev; Barak, Boaz: *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 2009. ISBN 0521424267, 9780521424264.
- [AFKS00] Alon, Noga; Fischer, Eldar; Krivelevich, Michael; Szegedy, Mario: Efficient testing of large graphs. In: *Combinatorica*, volume 20(4):pp. 451–476, 2000. doi:10.1109/SFCS.1999.814642.
- [Agr06] Agrawal, Manindra: Determinant versus permanent. In: *Proceedings of the 25th International Congress of Mathematicians, ICM 2006*, volume 3, pp. 985–997. 2006.
- [AKKR08] Alon, Noga; Kaufman, Tali; Krivelevich, Michael; Ron, Dana: Testing triangle-freeness in general graphs. In: *SIAM Journal on Discrete Mathematics*, volume 22(2):pp. 786–819, 2008. doi:10.1145/1109557.1109589.
- [Alo02] Alon, Noga: Testing subgraphs in large graphs. In: *Random structures and algorithms*, volume 21(3–4):pp. 359–370, 2002. doi:10.1002/rsa.10056.
- [AM07] Achlioptas, Dimitris; Moore, Christopher: Random k -SAT: two moments suffice to cross a sharp threshold. In: *SIAM Journal on Computing*, volume 36(3):pp. 740–762, 2007. doi:10.1137/S0097539703434231.
- [AP04] Achlioptas, Dimitris; Peres, Yuval: The threshold for random k -SAT is $2^k \log 2 - O(k)$. In: *Journal of the American Mathematical Society*, volume 17(4):pp. 947–973, 2004. doi:10.1090/S0894-0347-04-00464-3.
- [AS04] Alon, Noga; Shapira, Asaf: Testing subgraphs in directed graphs. In: *Journal of Computer and System Sciences*, volume 69(3):pp. 354–382, 2004. doi:10.1016/j.jcss.2004.04.008.
- [AS05] Alon, Noga; Shapira, Asaf: Linear equations, arithmetic progressions and hypergraph property testing. In: *Theory of Computing*, volume 1(1):pp. 177–216, 2005. doi:10.4086/toc.2005.v001a009.
- [AS06] Alon, Noga; Shapira, Asaf: A characterization of easily testable induced subgraphs. In: *Combinatorics, Probability and Computing*, volume 15(6):pp. 791–805, 2006. doi:10.1017/S0963548306007759.

Bibliography

- [BD07] Bläser, Markus; Dell, Holger: Complexity of the cover polynomial. In: *Proceedings of the 34th International Colloquium on Automata, Languages and Programming, ICALP 2007*, volume 4596 of *Lecture Notes in Computer Science*, pp. 801–812. Springer, 2007. doi:10.1007/978-3-540-73420-8_69.
- [BDFH09] Bodlaender, Hans L.; Downey, Rodney G.; Fellows, Michael R.; Hermelin, Danny: On problems without polynomial kernels. In: *Journal of Computer and System Sciences*, volume 75(8):pp. 423–434, 2009. ISSN 0022-0000. doi:10.1016/j.jcss.2009.04.001.
- [Beh46] Behrend, Felix A.: On sets of integers which contain no three terms in arithmetic progression. In: *Proceedings of the National Academy of Sciences, USA*, volume 32(12):pp. 331–332, 1946. doi:10.1073/pnas.32.12.331.
- [Ber84] Berkowitz, Stuart J.: On computing the determinant in small parallel time using a small number of processors. In: *Information Processing Letters*, volume 18(3):pp. 147–150, 1984. doi:10.1016/0020-0190(84)90018-8.
- [BG93] Buss, Jonathan F.; Goldsmith, Judy: Nondeterminism within P. In: *SIAM Journal on Computing*, volume 22(3):pp. 560–572, 1993. doi:10.1007/BFb0020811.
- [BH08a] Björklund, Andreas; Husfeldt, Thore: Exact algorithms for exact satisfiability and number of perfect matchings. In: *Algorithmica*, volume 52(2):pp. 226–249, 2008. doi:10.1007/s00453-007-9149-8.
- [BH08b] Buhrman, Harry; Hitchcock, John M.: NP-hard sets are exponentially dense unless NP is contained in coNP/poly. In: *Proceedings of the 23rd IEEE Conference on Computational Complexity, CCC 2008*, pp. 1–7. IEEE Computer Society, 2008. doi:10.1109/CCC.2008.21.
- [BHKK08] Björklund, Andreas; Husfeldt, Thore; Kaski, Petteri; Koivisto, Mikko: Computing the Tutte polynomial in vertex-exponential time. In: *Proceedings of the 47th annual IEEE Symposium on Foundations of Computer Science, FOCS 2008*, pp. 677–686. 2008. doi:10.1109/FOCS.2008.40.
- [Bry82] Brylawski, Thomas: The Tutte polynomial, Matroid theory and its applications. In: *Centro Internazionale Matematico Estivo*, pp. 125–275, 1982.
- [BTY09] Bodlaender, Hans L.; Thomassé, Stéphan; Yeo, Anders: Kernel bounds for disjoint cycles and disjoint paths. In: *Proceedings of the 17th Annual European Symposium on Algorithms, ESA 2009*, volume 5757 of *Lecture Notes in Computer Science*, pp. 635–646. Springer, 2009. doi:10.1007/978-3-642-04128-0_57.
- [CFL83] Chandra, Ashok K.; Furst, Merrick L.; Lipton, Richard J.: Multi-party protocols. In: *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, STOC 1983*, pp. 94–99. ACM, 1983. doi:10.1145/800061.808737.

- [CFM07] Chen, Yijia; Flum, Jörg; Müller, Moritz: Lower bounds for kernelizations. In: *Electronic Colloquium on Computational Complexity (ECCC)*, volume 14(137), 2007. URL <http://eccc.hpi-web.de/eccc-reports/2007/TR07-137/index.html>.
- [CIKP03] Calabro, Chris; Impagliazzo, Russell; Kabanets, Valentine; Paturi, Ramamohan: The Complexity of Unique k -SAT: An Isolation Lemma for k -CNFs. In: *Proceedings of the 18th IEEE Conference on Computational Complexity, CCC 2003*, p. 135. IEEE Computer Society, 2003. ISBN 0769518796. ISSN 1093-0159. doi:10.1109/CCC.2003.1214416.
- [CIP06] Calabro, Chris; Impagliazzo, Russell; Paturi, Ramamohan: A duality between clause width and clause density for SAT. In: *Proceedings of the 21st IEEE Conference on Computational Complexity, CCC 2006*, pp. 252–260. IEEE Computer Society, 2006. doi:10.1109/CCC.2006.6.
- [Coo71] Cook, Stephen A.: The complexity of theorem-proving procedures. In: *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, STOC 1971*, pp. 151–158. ACM, 1971. doi:10.1145/800157.805047.
- [CW90] Coppersmith, Don; Winograd, Shmuel: Matrix multiplication via arithmetic progressions. In: *Journal of Symbolic Computation*, volume 9(3):pp. 251–280, 1990. doi:10.1016/S0747-7171(08)80013-2.
- [DF99] Downey, Rodney G.; Fellows, Michael R.: *Parameterized complexity*. Springer New York, 1999.
- [DHW10] Dell, Holger; Husfeldt, Thore; Wahlén, Martin: Exponential time complexity of the permanent and the Tutte polynomial. In: *Proceedings of the 37th International Colloquium on Automata, Languages and Programming, ICALP 2010*, volume 6198 of *Lecture Notes in Computer Science*, pp. 426–437. Springer, 2010. doi:10.1007/978-3-642-14165-2_37.
- [Din07] Dinur, Irit: The PCP theorem by gap amplification. In: *Journal of the ACM*, volume 54(3):p. 12, 2007. doi:10.1145/1236457.1236459.
- [DJP⁺94] Dahlhaus, Elias; Johnson, David S.; Papadimitriou, Christos H.; Seymour, Paul D.; Yannakakis, Mihalis: The complexity of multiterminal cuts. In: *SIAM Journal on Computing*, volume 23(4):pp. 864–894, 1994. doi:10.1137/S0097539792225297.
- [DLS09] Dom, Michael; Lokshtanov, Daniel; Saurabh, Saket: Incompressibility through colors and IDs. In: *Proceedings of the 36th International Colloquium on Automata, Languages and Programming, ICALP 2009*, volume 5555 of *Lecture Notes in Computer Science*, pp. 378–389. Springer, 2009. doi:10.1007/978-3-642-02927-1_32.

Bibliography

- [DvM10] Dell, Holger; van Melkebeek, Dieter: Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In: *Proceedings of the 42th Annual ACM Symposium on Theory of Computing, STOC 2010*, pp. 251–260. ACM, 2010. doi:10.1145/1806689.1806725.
- [Elk10] Elkin, Michael: An improved construction of progression-free sets. In: *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*, pp. 886–905. SIAM, 2010.
- [ER60] Erdős, Paul; Rado, Richard: Intersection theorems for systems of sets. In: *Journal of the London Mathematical Society*, volume 35:pp. 85–90, 1960. doi:10.1112/jlms/s1-35.1.85.
- [FB99] Friedgut, Ehud; Bourgain, Jean: Sharp thresholds of graph properties, and the k -SAT problem. In: *Journal of the American Mathematical Society*, volume 12(4):pp. 1017–1054, 1999. doi:10.1090/S0894-0347-99-00305-7.
- [FFL⁺09] Fernau, Henning; Fomin, Fedor V.; Lokshtanov, Daniel; Raible, Daniel; Saurabh, Saket; Villanger, Yngve: Kernel(s) for problems with no kernel: On out-trees with many leaves. In: *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009*, volume 09001 of *Dagstuhl Seminar Proceedings*, pp. 421–432. 2009. doi:10.4230/LIPIcs.STACS.2009.1843.
- [FG06] Flum, Jörg; Grohe, Martin: *Parameterized Complexity Theory*. Springer, 2006.
- [FGMN09] Fellows, Michael R.; Guo, Jiong; Moser, Hannes; Niedermeier, Rolf: A generalization of Nemhauser and Trotter’s local optimization theorem. In: *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009*, volume 09001 of *Dagstuhl Seminar Proceedings*, pp. 409–420. 2009. doi:10.4230/LIPIcs.STACS.2009.1820.
- [FHR⁺04] Fellows, Michael R.; Heggernes, Pinar; Rosamond, Frances A.; Sloper, Christian; Telle, Jan Arne: Finding k disjoint triangles in an arbitrary graph. In: *Proceedings of the 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2004)*, volume 3353 of *Lecture Notes in Computer Science*, pp. 235–244. Springer, 2004. doi:10.1007/b104584.
- [FK72] Fortuin, Cees M.; Kasteleyn, Pieter W.: On the random-cluster model: I. Introduction and relation to other models. In: *Physica*, volume 57(4):pp. 536–564, 1972. ISSN 0031-8914. doi:10.1016/0031-8914(72)90045-6.
- [FKN⁺08] Fellows, Michael R.; Knauer, Christian; Nishimura, Naomi; Ragde, Prabhakar; Rosamond, Frances A.; Stege, Ulrike; Thilikos, Dimitrios M.; Whitesides, Sue: Faster fixed-parameter tractable algorithms for matching and packing problems. In: *Algorithmica*, volume 52:pp. 167–176, 2008. ISSN 0178-4617. doi:10.1007/s00453-007-9146-y.

- [FR09] Fernau, Henning; Raible, Daniel: A parameterized perspective on packing paths of length two. In: *Journal of Combinatorial Optimization*, volume 18(4):pp. 319–341, 2009. doi:10.1007/s10878-009-9230-0.
- [FS08] Fortnow, Lance; Santhanam, Rahul: Infeasibility of instance compression and succinct PCPs for NP. In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC 2008*, pp. 133–142. ACM, 2008. doi:10.1145/1374376.1374398.
- [GGR98] Goldreich, Oded; Goldwasser, Shafi; Ron, Dana: Property testing and its connection to learning and approximation. In: *Journal of the ACM*, volume 45(4):pp. 653–750, 1998. doi:10.1145/285055.285060.
- [GHN06] Giménez, Omer; Hliněný, Petr; Noy, Marc: Computing the Tutte polynomial on graphs of bounded clique-width. In: *SIAM Journal on Discrete Mathematics*, volume 20:pp. 932–946, 2006. doi:10.1007/11604686_6.
- [GJ07] Goldberg, Leslie Ann; Jerrum, Mark: The complexity of ferromagnetic Ising with local fields. In: *Combinatorics, Probability and Computing*, volume 16(1):pp. 43–61, 2007. doi:10.1017/S096354830600767X.
- [GJ08] Goldberg, Leslie Ann; Jerrum, Mark: Inapproximability of the Tutte polynomial. In: *Information and Computation*, volume 206(7):pp. 908–929, 2008. doi:10.1016/j.ic.2008.04.003.
- [GN07] Guo, Jiong; Niedermeier, Rolf: Invitation to data reduction and problem kernelization. In: *SIGACT News*, volume 38(1):pp. 31–45, 2007. ISSN 0163-5700. doi:10.1145/1233481.1233493.
- [Gol08] Goldreich, Oded: *Computational complexity: a conceptual perspective*. ACM New York, NY, USA, 2008.
- [GR01] Godsil, Chris; Royle, Gordon: *Algebraic Graph Theory*. Graduate Texts in Mathematics. Springer, April 2001. ISBN 0387952209.
- [GW08] Green, Ben; Wolf, Julia: A note on Elkin’s improvement of Behrend’s construction, 2008. arXiv:0810.0732.
- [HHN⁺95] Hemaspaandra, Lane A.; Hoene, Albrecht; Naik, Ashish V.; Ogihara, Mitsunori; Selman, Alan L.; Thierauf, Thomas; Wang, Jie: Nondeterministically selective sets. In: *International Journal of Foundations of Computer Science*, volume 6(4):pp. 403–416, 1995. doi:10.1142/S0129054195000214.
- [HN06] Harnik, Danny; Naor, Moni: On the compressibility of NP instances and cryptographic applications. In: *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2006*, pp. 719–728. 2006. doi:10.1109/FOCS.2006.54.

Bibliography

- [Hof10] Hoffmann, Christian: Exponential time complexity of weighted counting of independent sets. In: *Proceedings of the 5th International Symposium on Parameterized and Exact Complexity, IPEC 2010*, volume 6478 of *Lecture Notes in Computer Science*, pp. 180–191. Springer, 2010. doi:10.1007/978-3-642-17493-3_18.
- [HT10] Husfeldt, Thore; Taslaman, Nina: The exponential time complexity of computing the probability that a graph is connected. In: *Proceedings of the 5th International Symposium on Parameterized and Exact Complexity, IPEC 2010*, volume 6478 of *Lecture Notes in Computer Science*, pp. 192–203. Springer, 2010. doi:10.1007/978-3-642-17493-3_19.
- [HW03] Håstad, Johan; Wigderson, Avi: Simple analysis of graph tests for linearity and PCP. In: *Random Structures and Algorithms*, volume 22(2):pp. 139–160, 2003. doi:10.1002/rsa.10068.
- [IPZ01] Impagliazzo, Russell; Paturi, Ramamohan; Zane, Francis: Which problems have strongly exponential complexity? In: *Journal of Computer and System Sciences*, volume 63(4):pp. 512–530, 2001. ISSN 0022-0000. doi:10.1006/jcss.2001.1774.
- [Ist00] Istrail, Sorin: Statistical mechanics, three-dimensionality and NP-completeness. I. Universality of intractability for the partition function of the Ising model across non-planar lattices. In: *Proceedings of the 32nd annual ACM Symposium on Theory of Computing, STOC 2000*, pp. 87–96. ACM, 2000. doi:10.1145/335305.335316.
- [JS82] Jerrum, Mark; Snir, Marc: Some exact complexity results for straight-line computations over semirings. In: *Journal of the ACM*, volume 29(3):pp. 874–897, 1982. doi:10.1145/322326.322341.
- [JS93] Jerrum, Mark; Sinclair, Alistair: Polynomial-time approximation algorithms for the Ising model. In: *SIAM Journal on Computing*, volume 22(5):pp. 1087–1116, 1993. doi:10.1137/0222066.
- [JVW90] Jaeger, François; Vertigan, Dirk L.; Welsh, Dominic J.A.: On the computational complexity of the Jones and Tutte polynomials. In: *Mathematical proceedings of the Cambridge Philosophical Society*, volume 108(1):pp. 35–53, 1990. doi:10.1017/S0305004100068936.
- [Kar72] Karp, Richard M.: Reducibility among combinatorial problems. In: *Complexity of computer computations*, volume 43:pp. 85–103, 1972.
- [KD79] Krishnamoorthy, Mukkai S.; Deo, Narsingh: Node-deletion NP-complete problems. In: *SIAM Journal on Computing*, volume 8(4):pp. 619–625, 1979. doi:10.1137/0208049.

- [KH78] Kirkpatrick, David G.; Hell, Pavol: On the completeness of a generalized matching problem. In: *Proceedings of the 10th annual ACM Symposium on Theory of Computing, STOC 1978*, pp. 240–245. ACM, 1978. doi:10.1145/800133.804353.
- [KMW10] Kratsch, Stefan; Marx, Dániel; Wahlström, Magnus: Parameterized complexity and kernelizability of max ones and exact ones problems. In: *Proceedings of the 35th International Symposium on Mathematical Foundations of Computer Science, MFCS 2010*, pp. 489–500. 2010. doi:10.1007/978-3-642-15155-2_43.
- [Ko83] Ko, Ker-I: On self-reducibility and weak P-selectivity. In: *Journal of Computer and System Sciences*, volume 26(2):pp. 209–221, 1983. doi:10.1016/0022-0000(83)90013-2.
- [Koi09] Koivisto, Mikko: Partitioning into sets of bounded cardinality. In: *Proceedings of the 4th International Workshop on Parameterized and Exact Complexity, IWPEC 2009*, volume 5917 of *Lecture Notes in Computer Science*, pp. 258–263. Springer, 2009. doi:10.1007/978-3-642-11269-0_21.
- [Kot00] Kotlov, Andrei: Short proof of the Gallai-Edmonds structure theorem, 2000. arXiv:math/0011204.
- [Kut07] Kutzkov, Konstantin: New upper bound for the #3-SAT problem. In: *Information Processing Letters*, volume 105(1):pp. 1–5, 2007. ISSN 0020-0190. doi:10.1016/j.ipl.2007.06.017.
- [KW09] Kratsch, Stefan; Wahlström, Magnus: Two edge modification problems without polynomial kernels. In: *Proceedings of the 4th International Workshop on Parameterized and Exact Computation, IWPEC 2009*, volume 5917 of *Lecture Notes in Computer Science*, pp. 264–275. Springer, 2009. doi:10.1007/978-3-642-11269-0_22.
- [KW10] Kratsch, Stefan; Wahlström, Magnus: Preprocessing of min ones problems: A dichotomy. In: *Proceedings of the 37th Colloquium on Automata, Languages and Programming, ICALP 2010*, volume 6198 of *Lecture Notes in Computer Science*, pp. 653–665. Springer, 2010. doi:10.1007/978-3-642-14165-2_55.
- [Law76] Lawler, Eugene L.: A note on the complexity of the chromatic number problem. In: *Information Processing Letters*, volume 5:pp. 66–67, 1976.
- [Lev73] Levin, Leonid A.: Universal search problems (Russian: Universal’nye perebornye zadachi). In: *Problems of Information Transmission (Russian: Problemy Peredachi Informatsii)*, volume 9(3):pp. 265–266, 1973.

Bibliography

- [Lin86] Linial, Nathan: Hard enumeration problems in geometry and combinatorics. In: *SIAM Journal on Algebraic and Discrete Methods*, volume 7(2):pp. 331–335, 1986.
- [LY80] Lewis, John M.; Yannakakis, Mihalis: The node-deletion problem for hereditary properties is NP-complete. In: *Journal of Computer and System Sciences*, volume 20(2):pp. 219–230, 1980. doi:10.1016/0022-0000(80)90060-4.
- [Mos09] Moser, Hannes: A problem kernelization for graph packing. In: *Proceedings of the 35th Conference on Current Trends on Theory and Practice of Computer Science, SOFSEM 2009*, volume 5404 of *Lecture Notes in Computer Science*, pp. 401–412. Springer, 2009. doi:10.1007/978-3-540-95891-8_37.
- [MPS04] Mathieson, Luke; Prieto, Elena; Shaw, Peter: Packing edge disjoint triangles: A parameterized view. In: *Proceedings of the First International Workshop on Parameterized and Exact Computation, IWPEC 2004*, volume 3162 of *Lecture Notes in Computer Science*, pp. 127–137. Springer, 2004. doi:10.1007/978-3-540-28639-4_12.
- [Nie06] Niedermeier, Rolf: *Invitation to fixed-parameter algorithms*. Oxford University Press, USA, 2006.
- [NT74] Nemhauser, George L.; Trotter Jr., Leslie E.: Properties of vertex packing and independence system polyhedra. In: *Mathematical Programming*, volume 6(1):pp. 48–61, 1974. doi:10.1007/BF01580222.
- [Pap94] Papadimitriou, Christos M.: *Computational Complexity*. Addison-Wesley, Reading, Massachusetts, 1994. ISBN 0201530821.
- [PS04] Prieto, Elena; Sloper, Christian: Looking at the stars. In: *Proceedings of the First International Workshop on Parameterized and Exact Computation, IWPEC 2004*, volume 3162 of *Lecture Notes in Computer Science*, pp. 138–148. Springer, 2004. doi:10.1007/978-3-540-28639-4_13.
- [Raz09] Raz, Ran: Multi-linear formulas for permanent and determinant are of super-polynomial size. In: *Journal of the ACM*, volume 56(2):pp. 1–17, 2009. ISSN 0004-5411. doi:10.1145/1502793.1502797.
- [RS78] Ruzsa, Imre Z.; Szemerédi, Endre: Triple systems with no six points carrying three triangles. In: *Combinatorics (Proceedings of the Fifth Hungarian Colloquium, Keszthely, 1976), Vol. II*, volume 18 of *Colloquia Mathematica Societatis János Bolyai*, pp. 939–945. North-Holland, Amsterdam, 1978.
- [Rys63] Ryser, Herbert J.: Combinatorial mathematics. In: *Number 14 in Carus Math. Monographs*. Mathematical Association of America, 1963.
- [SIT95] Sekine, Kyoko; Imai, Hiroshi; Tani, Seiichiro: Computing the Tutte polynomial of a graph of moderate size. In: *Proceedings of the 6th International*

- Symposium on Algorithms and Computation, ISAAC 1995*, number 1004 in Lecture Notes in Computer Science, pp. 224–233. Springer, 1995. doi:10.1007/BFb0015427.
- [Sok04] Sokal, Alan D.: Chromatic roots are dense in the whole complex plane. In: *Combinatorics, Probability and Computing*, volume 13(02):pp. 221–261, 2004.
- [Sok05] Sokal, Alan D.: The multivariate Tutte polynomial (alias Potts model) for graphs and matroids. In: *Surveys in Combinatorics*, volume 327 of *London Mathematical Society Lecture Note Series*, pp. 173–226. Cambridge University Press, 2005.
- [SS42] Salem, Raphaël; Spencer, Donald C.: On sets of integers which contain no three terms in arithmetical progression. In: *Proceedings of the National Academy of Sciences, USA*, volume 28(12):pp. 561–563, 1942.
- [Tho09] Thomassé, Stéphan: A quadratic kernel for feedback vertex set. In: *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009*, pp. 115–119. SIAM, 2009. doi:10.1145/1496770.1496783.
- [Val79] Valiant, Leslie G.: The complexity of computing the permanent. In: *Theoretical Computer Science*, volume 8(2):pp. 189–201, 1979. doi:10.1016/0304-3975(79)90044-6.
- [Weg87] Wegener, Ingo: *The Complexity of Boolean Functions*. B. G. Teubner, and John Wiley & Sons, 1987. URL <http://citeseer.ist.psu.edu/700371.html>.
- [WNFC10] Wang, Jianxin; Ning, Dan; Feng, Qilong; Chen, Jianer: An improved kernelization for P_2 -packing. In: *Information Processing Letters*, volume 110(5):pp. 188–192, 2010. doi:10.1016/j.ipl.2009.12.002.
- [Yap83] Yap, Chee-Keng: Some consequences of non-uniform conditions on uniform classes. In: *Theoretical computer science*, volume 26(3):pp. 287–300, 1983. doi:10.1016/0304-3975(83)90020-8.
- [Yus07] Yuster, Raphael: Combinatorial and computational aspects of graph packing and graph decomposition. In: *Computer Science Review*, volume 1(1):pp. 12–26, 2007.

Erklärung

Hiermit erkläre ich,

- dass ich die vorliegende Arbeit mit dem Titel „Sparse Instances of Hard Problems“ selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt und sie an keiner anderen Universität eingereicht habe,
- dass mir die Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät II der Humboldt-Universität zu Berlin vom 17.01.2005, zuletzt geändert am 13.02.2006, veröffentlicht im Amtlichen Mitteilungsblatt Nr. 34/2006, bekannt ist,
- dass ich keinen Doktorgrad im Fach Informatik besitze.

Berlin, den 01.06.2011

Holger Dell