Die einfachen, kleinen und langsamen Dinge zählen: Über die parametrisierte Komplexität von Zählproblemen[1]

Radu Curticapean[2]

Abstract: Wir untersuchen kombinatorische Zählprobleme hinsichtlich ihrer parametrisierten und Exponentialzeit-Komplexität. Im Vordergrund stehen hierbei das Zählen von Paarungen in strukturell einfachen Graphen, das Zählen kleiner Subgraph-Muster in sehr großen Zielgraphen, sowie exponentielle untere Schranken an die Laufzeit, die zum Lösen von Zählproblemen benötigt wird.

1 Einführung

Viele Probleme der theoretischen Informatik gehen der Frage nach, ob bestimmte mit der Eingabe verbundene Strukturen *existieren*. Ein prominentes Beispiel hierfür sind die Probleme in NP, wie etwa das NP-vollständige Erfüllbarkeitsproblem SAT für aussagenlogische Formeln. Oftmals ist es allerdings ebenso wichtig, eine bestimmte Struktur zu *finden*, alle Strukturen *aufzulisten* oder die *Anzahl* solcher Strukturen zu bestimmen. Es ist diese letzte Aufgabe, der sich die vorliegende Dissertation widmet. Beispielsweise lässt sich ein Zählproblem #SAT definieren, in dem auf Eingabe einer Formel φ nach der *Anzahl* der erfüllenden Belegungen von φ gefragt wird. Dies ist offensichtlich schwerer als das Entscheiden der Erfüllbarkeit von φ, und aufgrund von Todas Theorem [To91] lassen sich mit einem Orakel für #SAT sogar effizient Probleme entscheiden, die wir außerhalb von NP vermuten, nämlich die gesamte Polynomialzeit-Hierarchie.

Zählprobleme treten allerdings auch in der freien Natur außerhalb der theoretischen Informatik auf: In der statistischen Physik etwa lassen sich Aussagen über die thermodynamischen Eigenschaften von Systemen treffen, indem ihre *Partitionsfunktionen* berechnet werden [Ka67]. Dies entspricht dem (gewichteten) Zählen von Konfigurationen des Systems. Da es typischerweise eine exponentielle Anzahl von Konfigurationen gibt, sind effizientere Methoden als das Abzählen sämtlicher Konfigurationen nötig – und in einzelnen Fällen tatsächlich möglich [TF61, Ka61]. In der Bioinformatik oder der Analyse von Netzwerken ergeben sich Zählprobleme, wenn etwa nachgewiesen werden soll, dass ein gewisses Muster in einem Netzwerk mit signifikanter Häufigkeit auftritt [GS10, UBK13].

Für die meisten dieser Probleme sind jedoch keine effizienten Algorithmen bekannt, wodurch eine Komplexitätstheorie von Zählproblemen notwendig wurde. Diese wurde durch Valiant [Va79] eingeführt, indem er eine Komplexitätsklasse #P definierte, für die beispielsweise #SAT ein vollständiges Problem darstellt. Interessanterweise zeigte Valiant

[1] Englischer Titel der Arbeit: “The simple, little and slow things count: On parameterized counting complexity”

[2] Simons Institute for the Theory of Computing, UC Berkeley, curticapean@berkeley.edu

auch, dass es (unter polynomiellen Turing-Reduktionen) natürliche #P-vollständige Probleme gibt, deren Entscheidungsversion effizient lösbar ist: So zeigte er etwa die #P-Vollständigkeit des Zählens perfekter Paarungen in Graphen, obwohl die Existenz einer perfekten Paarung bekanntermaßen in Polynomialzeit entschieden werden kann. Eine *Paarung* in einem Graphen $G = (V,E)$ ist hierbei eine Teilmenge M von E, so dass jeder Knoten mit höchstens einer Kante in M inzident ist. Die Paarung ist *perfekt*, wenn jeder Knoten mit *exakt* einer Kante inzident ist.

Die Analyse von Zählproblemen hat sich seit diesem programmatischen Resultat zu einem klassischen Teilgebiet der Komplexitätstheorie etabliert, und das spezifische Problem des Zählens perfekter Paarungen spielte in dieser Entwicklung durchgehend eine besondere Rolle. Wir kürzen dieses Problem im Folgenden durch PerfMatch ab. Beispielsweise wurde bereits vor Valiants Resultat gezeigt, dass PerfMatch auf *planaren* Graphen in Polynomialzeit lösbar ist [TF61, Ka61, Ka67]. Darauf aufbauend hat Valiant [Va08] kürzlich das Konzept der sogenannten *holographischen Algorithmen* eingeführt, die es erlauben, eine Reihe von Zählproblemen auf diesen positiven Fall zu reduzieren. In der Kombinatorik und der algebraischen Komplexitätstheorie ist die Anzahl perfekter Paarungen in bipartiten Graphen G bekannt als die *Permanente* der Bi-Adjazenzmatrix von G, und eine algebraische Variante des "P vs. NP"-Problems zielt darauf ab, die Komplexitäten der Permanente und der Determinante zu unterscheiden [Ag06].

Da PerfMatch wie viele interessante Zählprobleme #P-schwer ist, wurden im Laufe der Zeit verfeinerte Versionen betrachtet. Klassische Strategien für einen verfeinerten Umgang mit Zählproblemen sind die Einschränkung auf bestimmte Klassen von Eingaben (etwa planare Graphen), das Zählen modulo kleiner Primzahlen (so lässt sich PerfMatch etwa modulo 2 in Polynomialzeit berechnen) und das approximative Zählen (ein berühmter Algorithmus [JSV04] erlaubt eine effiziente Approximation von PerfMatch für bipartite Graphen). In dieser Dissertation widmen wir uns zwei der jüngsten Verfeinerungen der Analyse von Zählproblemen, nämlich der *parametrisierten* Komplexitätstheorie von Zählproblemen (eingeführt von Flum und Grohe [FG04]) und ihrer *Exponentialzeit*-Komplexitätstheorie (eingeführt von Dell et al. [De14]).

1.1 Parametrisierte Komplexitätstheorie von Zählproblemen

Der Gegenstandsbereich der parametrisierten Komplexitätstheorie sind *parametrisierte Probleme*. Dies sind Berechnungsprobleme mit Eingaben $x \in \{0,1\}^*$, die mit *Parametrisierungen* $\kappa : \{0,1\}^* \to \mathbb{N}$ versehen werden, so dass in typischen Anwendungsfällen der Parameterwert $\kappa(x)$ deutlich kleiner als die Eingabelänge $|x|$ ist. Die konkrete Wahl der Parametrisierung hängt von der erwünschten Anwendung ab; typische Parameter für Graphenprobleme lassen sich aber in zwei Gruppen einteilen:

- **Strukturelle Parameter** messen die Komplexität des Eingabegraphens; wir wenden sie in dieser Dissertation auf das Problem PerfMatch an. Ein kleiner Parameterwert bedeutet hier, dass der Graph eine einfache Struktur aufweist, die sich potentiell

algorithmisch nutzen lässt. Beispiele hierfür sind etwa der Maximalgrad von G, die Kreuzungszahl oder der Genus von G.[3]

- Für viele Zählprobleme enthält die Eingabe selbst bereits eine Zahl $k \in \mathbb{N}$, so dass Strukturen der Größe k gezählt werden sollen; in solchen Fällen ist eine **Parametrisierung durch** k oft sinnvoll. Beispielsweise kann man, gegeben einen Graphen G und $k \in \mathbb{N}$, nach der Anzahl von Paarungen mit exakt k Kanten oder der Anzahl von Knotenüberdeckungen mit exakt k Knoten fragen.[4] Parametrisiert man nun durch k, so bedeutet das intuitiv, dass man Teilstrukturen zählen möchte, die deutlich kleiner als der eigentliche Eingabegraph sind – dies macht etwa Sinn, wenn kleine Muster in riesigen Datenbanken gesucht werden.

Wurde ein sinnvoller Parameter für ein gegebenes #P-schweres Problem identifiziert, so stellt sich die Frage, inwieweit dieser sich tatsächlich algorithmisch nutzen lässt. Hier können nun, je nach Problem, drei wesentliche Fälle auftreten [FG06]:

1. Im schlechtesten Fall ist das Problem für konstante Werte von $\kappa(x)$ bereits #P-schwer. So ist etwa PerfMatch in Graphen mit Maximalgrad 3 bereits #P-vollständig.

2. Bessere Aussichten erhalten wir, wenn sich das Problem für jeden festen Wert von $\kappa(x)$ in Polynomialzeit lösen lässt – oder besser, wenn wir einen Algorithmus finden können, der das Problem in Zeit $O(|x|^{f(\kappa(x))})$ löst, wobei f eine berechenbare Funktion ist. Wir sprechen dann von einem **XP-Algorithmus**. Derartige Algorithmen sind etwa für das Zählen von Paarungen mit k Kanten in Graphen mit n Knoten gegeben, da ein einfacher Brute-Force-Ansatz eine Laufzeit von $n^{O(k)}$ garantiert.

3. Im idealen Fall erhalten wir sogar einen Algorithmus, dessen Laufzeit für jeden festen Wert von $\kappa(x)$ durch $O(|x|^c)$ beschränkt ist, wobei $c \in \mathbb{N}$ eine feste Konstante ist. Wir können also im Vergleich zum vorherigen Fall zusätzlich den Parameter $\kappa(x)$ aus dem Exponenten von $|x|$ entfernen. Allerdings darf (und muss) hierbei die in der O-Notation verborgene Konstante super-polynomiell von $\kappa(x)$ abhängen, was zum Begriff der **fixed-parameter tractability (FPT)** führt: Ein Problem mit Parameter κ ist FPT, wenn es auf Eingaben x in Zeit $f(\kappa(x)) \cdot |x|^c$ gelöst werden kann, wobei c eine feste Konstante und f eine beliebige berechenbare Funktion ist. Beispielsweise lassen sich Knotenüberdeckungen der Größe k auf Graphen mit n Knoten in Zeit $O(2^k \cdot n)$ zählen [FG04]. Weiterhin lassen sich perfekte Paarungen auf Graphen vom Genus g in Zeit $O(4^g \cdot n^3)$ zählen [GL98].

Wir können nun eines der Hauptanliegen der parametrisierten Algorithmik formaler benennen: Gegeben ein parametrisiertes Problem, können wir einen XP-Algorithmus finden? Und falls ja, ist sogar ein FPT-Algorithmus möglich? Sollte eine dieser Fragen negativ beantwortet werden, tritt die Komplexitätstheorie ins Spiel und versucht, die erwünschten

[3] Die Kreuzungszahl von G ist die minimale Anzahl von Kreuzungen über alle Zeichnungen von G in der Ebene. Der Genus von G ist der minimale Genus einer Oberfläche, auf die G kreuzungsfrei gezeichnet werden kann.

[4] Eine Knotenüberdeckung eines Graphens $G = (V,E)$ ist eine Teilmenge $S \subseteq V$, so dass jede Kante $e \in E$ mindestens einen Knoten aus S enthält.

Algorithmen auszuschließen. So können XP-Algorithmen für parametrisierte Probleme bereits mittels klassischer Komplexitätstheorie ausgeschlossen werden: Es reicht hierfür aus, die #P-Schwere des Problems für einen konstanten Wert des Parameters zu zeigen. Anders stellt sich die Situation bei Problemen dar, die immerhin XP-Algorithmen zulassen: Da diese für jede feste Wahl des Parameters in Polynomialzeit lösbar sind, lässt die klassische Komplexitätstheorie hier keine Aussage zu. Um dennoch die Abwesenheit von FPT-Algorithmen für das Problem erklären zu können, wurde die Komplexitätsklasse #W[1] (analog zu #P) und der Begriff von #W[1]-schweren Problemen eingeführt [FG04].

Die Komplexitätsklasse **#W[1]** kann definiert werden als die Menge aller parametrisierten Probleme, die sich über **parametrisierte Reduktionen** auf das Zählen von k-Cliquen in Graphen (parametrisiert durch k) reduzieren lassen. Eine parametrisierte Reduktion von einem Problem A mit Parameter κ auf ein Problem B mit Parameter τ ist hierbei ein Algorithmus, der das Problem A auf Eingaben x in Zeit $f(\kappa(x)) \cdot |x|^{O(1)}$ löst, wobei Orakelzugriff auf das Problem B gegeben ist. Hier müssen allerdings alle Anfragen y an das Orakel die Parameterbeschränkung $\tau(y) \leq g(\kappa(x))$ aufweisen. Die Funktionen f und g sind hierbei beliebige berechenbare Funktionen. Durch parametrisierte Reduktionen vom Zählen von k-Cliquen lässt sich beispielsweise zeigen, dass das Zählen von Pfaden und Kreisen der Größe k ebenfalls #W[1]-vollständig ist [FG04]. Folgt man nun der weitläufigen Annahme, dass FPT und #W[1] unterschiedliche Klassen sind, so hat keines dieser #W[1]-vollständigen Probleme einen FPT-Algorithmus.

1.2 Exponentialzeit-Komplexität für Zählprobleme

Wurde ein Problem als FPT oder #W[1]-schwer klassifiziert, so wissen wir, ob wir Laufzeiten vom Typ $f(k) \cdot n^{O(1)}$ oder $n^{f(k)}$ erwarten sollen. In der Entwicklung der parametrisierten Komplexitätstheorie zeigte sich jedoch, dass eine noch feinere Analyse möglich ist: Durch die *Exponentialzeit-Komplexitätstheorie* [IPZ01, De14] lässt sich oftmals präzise angeben, welches Wachstum für f zu erwarten ist. Hierzu nutzen wir die *Exponentialzeit-Hypothese* ETH und ihre Zählversion #ETH, welche postulieren, dass SAT bzw. #SAT für Formeln mit n Variablen nicht in Zeit $2^{o(n)}$ gelöst werden kann. Eine Falsifikation von ETH oder #ETH würde einen fundamentalen und unerwarteten Durchbruch für die Theorie von SAT-Algorithmen darstellen.

Betrachten wir nun ein Problem mit Parameter κ, und nehmen wir an, wir können in Polynomialzeit #SAT für Formeln φ mit n Variablen auf Instanzen x des Problems reduzieren, so dass $\kappa(x) = O(n)$ gilt. Dann schließt #ETH Algorithmen mit Laufzeit $2^{o(\kappa(x))} \cdot |x|^{O(1)}$ für das Problem aus. Ein derartiger Ansatz erlaubt es etwa, einen Algorithmus mit Laufzeit $2^{o(g)} \cdot n^{O(1)}$ für PerfMatch in Graphen von Genus g auszuschließen [CM16] und komplementiert somit die zuvor genannte obere Schranke von $O(4^g \cdot n^3)$.

1.3 Inhalte der Dissertation

Die vorliegende Dissertation [Cu15b] befasst sich mit verschiedenen Aspekten der parametrisierten Komplexität und der Exponentialzeit-Komplexität von Zählproblemen. Im

Einleitungsteil werden zunächst Grundlagen aus der Komplexitäts- und Graphentheorie sowie der Algebra und Kombinatorik gesammelt. Ebenfalls enthalten ist eine Einführung in das sogenannte Holant-Framework, das für die meisten nachfolgenden Resultate benötigt wird. Darauf aufbauend wird im ersten Hauptteil das Problem PerfMatch unter strukturellen Parametern analysiert. Im zweiten Hauptteil widmen wir uns dann dem Zählen kleiner Subgraph-Muster in großen Eingabegraphen. Im letzten Hauptteil befassen wir uns eingehend mit exponentiellen unteren Schranken für Zählprobleme wie PerfMatch. Jeder Hauptteil entspricht einem Adjektiv im Titel der Dissertation.

2 Das Holant-Framework

Ein Ziel der Dissertation liegt in der Entwicklung von allgemein nutzbaren Techniken für Zählprobleme. Hierfür erweitern wir im einleitenden Teil zunächst die Theorie der sogenannten *Holant-Probleme* [Va08, CLX09]. Die Eingaben für solche Probleme sind Graphen $G = (V, E)$, deren Knoten $v \in V$ mit *Signaturen* f_v versehen sind: Ist $I(v)$ die Menge der mit v inzidenten Kanten, so weist f_v jeder Belegung $\{0,1\}^{I(v)}$ einen Wert zu. Auf Eingabe solcher Graphen gilt es dann, den Wert $\mathrm{Holant}(G)$ zu berechnen: Dies ist eine Summe über alle Belegungen $x \in \{0,1\}^E$, wobei jede Belegung x mit dem Produkt der Auswertungen $f_v(x|_{I(v)})$ über alle Knoten v gewichtet wird. Hier ist $x|_{I(v)}$ die Einschränkung der Belegung x auf die Kanten $I(v)$. Wir erhalten also, formaler ausgedrückt:

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

Beispielsweise lässt sich PerfMatch durch geeignete Wahl der Signaturen auf natürliche Weise als Holant-Problem ausdrücken: Jede Signatur muss nur überprüfen, ob exakt eine Kante mit Belegung 1 am Knoten anliegt. Das Problem #SAT lässt sich ähnlich als Holant-Problem ausdrücken.

Wir führen zwei neue Techniken für Holant-Probleme ein, die in der Dissertation und nachfolgenden Arbeiten [CX15, CM16] genutzt werden: Erstens konstruieren wir eine uniforme Reduktion von beliebigen Holant-Problemen auf PerfMatch. Hierbei ersetzen wir die Knoten im Graphen des Ausgangsproblems durch Gadgets, die deren Signaturen simulieren. Wir können insbesondere zeigen, dass derartige Gadgets immer existieren. Zweitens führen wir (gemeinsam mit Mingji Xia) Linearkombinationen von Signaturen für parametrisierte Reduktionen zwischen Holant-Problemen ein. Enthält ein Graph G eine kleine Anzahl k von "besonderen" Signaturen, die sich als Linearkombinationen von $c \in \mathbb{N}$ "gewöhnlichen" Signaturen darstellen lassen, so können wir $\mathrm{Holant}(G)$ als Linearkombination von c^k Werten vom Typ $\mathrm{Holant}(G')$ ausdrücken, wobei in den Graphen G' nur gewöhnliche Signaturen auftreten. Die entstehende Laufzeit von $O(c^k)$ erlaubt in unseren Anwendungen parametrisierte Reduktionen.

3 Perfekte Paarungen in strukturell einfachen Graphen zählen

Im ersten Hauptteil beschäftigen wir uns mit PerfMatch auf Graphen, die feste Minoren[5] ausschließen. Dies lässt sich als Verallgemeinerung des planaren Falles betrachten, welcher in Polynomialzeit gelöst werden kann [TF61, Ka61, Ka67]. Der Begriff des Minors spielt eine zentrale Rolle in der Graphentheorie und führte zum sogenannten Graphminorentheorem [RS04], das besagt, dass jede unter Minoren abgeschlossene Klasse von Graphen $\mathscr{C}$ durch eine endliche Menge $F(\mathscr{C})$ von verbotenen Minoren ausgedrückt werden kann. Dies gilt beispielsweise für die Klasse von planaren Graphen, die exakt die Graphen sind, die $K_{3,3}$ und K_5 als Minoren ausschließen.

Es stellt sich heraus, dass alle bekannten Klassen, in denen PerfMatch in Polynomialzeit gelöst werden kann, einen festen Minor ausschließen: Dies gilt für planare Graphen, für $K_{3,3}$-freie Graphen [Li74], für Graphen von beschränktem Genus [GL98], und für K_5-freie Graphen [STW14]. Von diesen einzelnen Resultaten ausgehend, fragen wir uns, wie das Ausschließen beliebiger Minoren H die Komplexität des Problems PerfMatch beeinflusst, wenn wir durch die Größe von H parametrisieren. Hierfür orientieren wir uns am Graphstrukturtheorem [RS03], das die Struktur von Graphen G beschreibt, die einen festen Minor H ausschließen: Für jeden Graphen H gibt es eine Konstante $k = k(H)$, so dass sich die H-freien Graphen aus bestimmten elementaren Graphen zusammensetzen lassen. Ein elementarer Graph ist hierbei ein Graph vom Genus k, in welchen lokal k sogenannte Strudel (bestimmte nicht-planare Graphen) eingesetzt werden dürfen. Anschließend dürfen k sogenannte Apex-Knoten an den entstehenden Graphen angefügt werden, die beliebig mit anderen Knoten verbunden sein können.

Wie wir wissen, ist PerfMatch auf Graphen mit beschränktem Genus FPT [GL98]. Um den generellen Fall zu verstehen, müssen wir also die Auswirkungen von Apex-Knoten und Strudeln auf die Komplexität verstehen. Wir zeigen (mit Mingji Xia), dass bereits k Apex-Knoten an einem planaren Graphen FPT-Algorithmen ausschließen. Hierfür nutzen wir unsere neue Technik der Linearkombinationen von Signaturen.

Theorem 1. *Das folgende Problem ist* #W[1]*-schwer: Gegeben einen Graphen G und eine Knotenmenge $A \subseteq V(G)$, so dass $G - A$ planar ist, bestimme die Anzahl perfekter Paarungen in G. Hierbei parametrisieren wir durch $|A|$.*

Hieraus folgt eine Reihe von Aussagen, so etwa die #W[1]-Schwere von PerfMatch, wenn wir durch die Größe des minimalen ausgeschlossenen Minors H parametrisieren. Dennoch ist es möglich, FPT-Algorithmen für eingeschränkte Klassen von ausgeschlossenen Minoren zu erhalten, und tatsächlich können wir eine derartige Klasse identifizieren: Alle Minoren, die sich mit höchstens einer Kreuzung in die Ebene zeichnen lassen. Dies umfasst beispielsweise $K_{3,3}$ und K_5 und verallgemeinert einige der oben genannten Algorithmen:

Theorem 2. *Ist H ein Graph, der sich mit höchstens einer Kreuzung in die Ebene zeichnen lässt, so lässt sich* PerfMatch *auf H-freien Graphen in Zeit $O(f(H) \cdot n^4)$ lösen.*

[5] Ein Graph H ist ein Minor von G, wenn sich H aus G durch Löschung von Knoten und Kanten sowie durch Kontraktionen von Kanten erhalten lässt.

Die Dissertation hinterlässt die Frage, ob sich PerfMatch in Zeit $n^{f(H)}$ auf allgemeinen H-freien Graphen lösen lässt. Kürzlich konnte diese Frage durch den Autor negativ beantwortet werden.

4 Kleine Subgraphen in großen Graphen zählen

Im zweiten Hauptteil zählen wir kleine Subgraph-Muster H auf k Knoten (etwa Kreise oder Pfade der Größe k) in allgemeinen Graphen G mit n Knoten, parametrisiert durch k. Ein simpler Brute-Force-Ansatz liefert hierfür eine Laufzeit von $n^{O(k)}$, welche allerdings selbst für kleine Werte von k oft nicht mehr vertretbar ist. Wir fragen wir uns daher, welche Eigenschaften von H sich ausnutzen lassen, um FPT-Algorithmen zu erhalten. Hierzu führen wir formal das folgende Problem $\#\mathsf{Sub}(\mathscr{H})$ für feste Graphklassen $\mathscr{H}$ ein: Gegeben Graphen G und $H \in \mathscr{H}$, zähle die Subgraphen in G, die isomorph zu H sind, parametrisiert durch $|V(H)|$. Unser Ziel ist es, für jede Klasse $\mathscr{H}$ anzugeben, ob $\#\mathsf{Sub}(\mathscr{H})$ FPT oder $\#\mathsf{W}[1]$-schwer ist.

Ist $\mathscr{H}$ beispielsweise die Klasse von Kreisen oder die Klasse von Pfaden, so wurde die $\#\mathsf{W}[1]$-Vollständigkeit von $\#\mathsf{Sub}(\mathscr{H})$ bereits von Flum und Grohe [FG04] bewiesen. Die selben Autoren vermuteten auch, dass $\#\mathsf{Sub}(\mathscr{M})$ für die Klasse $\mathscr{M}$ der Paarungen $\#\mathsf{W}[1]$-vollständig ist. Wir zeigten zunächst (mit Markus Bläser), dass eine gewichtete Variante dieses Problems $\#\mathsf{W}[1]$-schwer ist [BC12]. Anschließend konnte der Autor der Dissertation die $\#\mathsf{W}[1]$-Vollständigkeit des ungewichteten Problems zeigen [Cu13]. Im Hauptteil der Dissertation zeigen wir einen vereinfachten Beweis (mit Dániel Marx), der die im einleitenden Teil eingeführten Linearkombinationen von Signaturen ausnutzt [CM14]. Wir erhalten somit:

Theorem 3. *Das Problem* $\#\mathsf{Sub}(\mathscr{M})$ *ist* $\#\mathsf{W}[1]$*-vollständig. Gegeben einen Graphen* G *und* $k \in \mathbb{N}$*, ist es also* $\#\mathsf{W}[1]$*-vollständig, die Paarungen mit* k *Kanten in* G *zu zählen.*

Dies lässt sich als die parametrisierte Variante der $\#\mathsf{P}$-Vollständigkeit von PerfMatch auffassen und stellt ein nützliches Ausgangsproblem für weitere Reduktionen dar. Insbesondere erlaubt es jedoch, die Probleme $\#\mathsf{Sub}(\mathscr{H})$ vollständig zu klassifizieren, da $\#\mathsf{Sub}(\mathscr{M})$ hierfür einen minimalen schweren Fall darstellt: Es ist bekannt, dass $\#\mathsf{Sub}(\mathscr{H})$ einen Algorithmus mit Laufzeit $n^{O(1)}$ gestattet, wenn die Graphen in $\mathscr{H}$ nur Paarungen konstanter Größe enthalten [WW13]. Dies trifft beispielsweise auf die Klasse $\mathscr{S}$ der Sterne[6] zu. Ist nun andererseits eine Klasse $\mathscr{H}$ gegeben, in deren Graphen sich beliebig große Paarungen finden lassen, so zeigen wir (mit Dániel Marx), dass sich $\#\mathsf{Sub}(\mathscr{M})$ auf $\#\mathsf{Sub}(\mathscr{H})$ reduzieren lässt. Dies zeigt:

Theorem 4. *Es sei* $\mathscr{H}$ *eine rekursiv aufzählbare Klasse von Graphen. Gibt es eine Konstante* $c \in \mathbb{N}$*, so dass kein Graph in* $\mathscr{H}$ *eine Paarung mit mehr als* c *Kanten enthält, so lässt sich* $\#\mathsf{Sub}(\mathscr{H})$ *in Zeit* $O(n^d)$ *lösen, wobei* d *von* c *abhängt. Lassen sich hingegen beliebig große Paarungen in* $\mathscr{H}$ *finden, so ist* $\#\mathsf{Sub}(\mathscr{H})$ *ein* $\#\mathsf{W}[1]$*-vollständiges Problem.*

[6] Ein Stern besteht aus Knoten $w, v_1 \ldots, v_n$ und den Kanten wv_i für alle i.

Es ist anzumerken, dass das Theorem für jedes Problem #Sub($\mathscr{H}$) zeigt, ob es in *Polynomialzeit* lösbar oder #W[1]-vollständig ist. Unter der Annahme, dass FPT und #W[1] nicht zusammenfallen, erhalten wir somit eine exakte Klassifikation der Probleme #Sub($\mathscr{H}$), die in Polynomialzeit lösbar sind. Tatsächlich wäre dies mit klassischen Methoden nicht möglich, da es #P-intermediäre Fälle für #Sub($\mathscr{H}$) gibt.

5 Untere Schranken für Zählprobleme unter #ETH

Im letzten Teil fragen wir uns, ob klassische Probleme wie PerfMatch in subexponentieller Zeit gelöst werden können, also in Zeit $2^{o(n)}$ für Graphen mit n Knoten. Hierfür nehmen wir die Exponentialzeit-Hypothese #ETH an.

Es lässt sich (mit den Techniken aus dem einleitenden Teil) einfach zeigen, dass eine gewichtete Variante von PerfMatch auf Graphen mit Kantengewichten 1 und -1 keinen Algorithmus mit Laufzeit $2^{o(n)}$ erlaubt, sofern #ETH gilt. Für Graphen $G = (V,E)$ mit Kantengewichten $w : E \to \{-1,1\}$ fragt diese Variante nach dem Wert von $\sum_M \prod_{e \in M} w(e)$, wobei M sich über alle perfekten Paarungen von G erstreckt. Es ist nun jedoch essentiell, das Gewicht -1 zu entfernen, da es beispielsweise für Reduktionen von PerfMatch auf Zielprobleme unklar ist, wie sich negative Kantengewichte simulieren lassen. Eine klassische Lösung hierfür nutzt das Prinzip der Polynominterpolation aus [De14]: Indem wir jedes Kantengewicht -1 durch eine Variable x ersetzen, erhalten wir einen Graphen G_x, so dass PerfMatch(G_x) ein Polynom in x vom Grad $\leq \frac{n}{2}$ ist und $p(-1) = \text{PerfMatch}(G)$ gilt. Sind die Werte $p(1), \ldots, p(\frac{n}{2}+1)$ bekannt, so können wir p interpolieren und $p(-1)$ auswerten. Diese Werte lassen sich nun jedoch durch eine einfache Reduktion auf den ungewichteten Fall von PerfMatch bestimmen: Setzen wir $x = t$ in G_x für positives $t \in \mathbb{N}$, so erhalten wir einen Graphen mit Gewichten 1 und t. Positive ganzzahlige Gewichte t lassen sich aber durch ungewichtete Gadgets mit $O(t)$ Knoten und Kanten simulieren. Wir erhalten so eine Reduktion von PerfMatch(G) mit Gewichten ± 1 auf $O(n)$ Instanzen des ungewichteten Problems auf Graphen G_t, wobei G_t höchstens $O(nt)$ Knoten hat. Der größte Graph G_t hat wegen $t = O(n)$ allerdings $O(n^2)$ Knoten. Eine scharfe untere Schranke unter #ETH können wir mit dieser Methode also nicht zeigen: Um einen Algorithmus mit Laufzeit $2^{o(n)}$ für PerfMatch mit Gewichten 1 und -1 zu erhalten, müssten wir einen Algorithmus mit Laufzeit $2^{o(\sqrt{n})}$ für den ungewichteten Fall finden.

In der Dissertation lösen wir dieses Problem auf zwei verschiedene Arten. In der ersten Lösung führen wir eine allgemeine Technik ein, die wir als "Block-Interpolation" bezeichnen [Cu15a]. Diese erlaubt es, viele klassische #P-Vollständigkeitsbeweise mit geringem Aufwand zu scharfen unteren Schranken unter #ETH zu erweitern. Zentral ist hierfür die Einsicht, dass sich Interpolations-Argumente, in denen wie oben *eine* Variable x eingeführt wird, oft zu multivariaten Versionen ausbauen lassen. Dies ermöglicht Reduktionen, die zwar $2^{o(n)}$ Zeit benötigen, aber nur auf Instanzen des Zielproblems abbilden, die $O(n)$ Knoten haben. Wir können somit die folgenden Schranken zeigen:

Theorem 5. *Sofern* #ETH *gilt, lassen sich perfekte Paarungen, Paarungen, unabhängige Mengen und Knotenüberdeckungen nicht in Zeit $2^{o(n)}$ auf Graphen mit n Knoten zählen. Dies gilt auch für die Auswertung der sogenannten Paarungs- und Tutte-Polynome.*

Unsere zweite Lösung trifft nur auf das spezifische Problem PerfMatch zu: Für jeden Graphen G mit n Knoten und m Kanten mit Gewichten ± 1 zeigen wir die Existenz zweier ungewichteter Graphen G_1 und G_2 mit $O(n+m)$ Knoten und Kanten, so dass $\text{PerfMatch}(G) = \text{PerfMatch}(G_1) - \text{PerfMatch}(G_2)$ gilt [Cu16]. Neben der daraus folgenden unteren Schranke für PerfMatch können wir zeigen, dass die folgenden Probleme unter polynomiellen Many-One-Reduktionen äquivalent sind: Erstens, gegeben zwei Formeln φ_1, φ_2, entscheide, ob diese die gleiche Anzahl erfüllender Belegungen haben. Zweitens, gegeben zwei Graphen G_1, G_2, entscheide, ob diese die gleiche Anzahl perfekter Paarungen haben.

Literaturverzeichnis

[Ag06] Agrawal, Manindra: Determinant versus permanent. In: Proceedings of the 25th International Congress of Mathematicians, ICM 2006. Jgg. 3, S. 985–997, 2006.

[BC12] Bläser, Markus; Curticapean, Radu: Weighted Counting of k-Matchings Is #W[1]-Hard. In: IPEC 2012. S. 171–181, 2012.

[CLX09] Cai, Jin-yi; Lu, Pinyan; Xia, Mingji: Holant problems and counting CSP. In: STOC 2009. S. 715–724, 2009.

[CM14] Curticapean, Radu; Marx, Dániel: Complexity of Counting Subgraphs: Only the Boundedness of the Vertex-Cover Number Counts. In: FOCS 2014. S. 130–139, 2014.

[CM16] Curticapean, Radu; Marx, Dániel: Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. In: SODA 2016. S. 1650–1669, 2016.

[Cu13] Curticapean, Radu: Counting Matchings of Size k Is #W[1]-Hard. In: ICALP 2013. S. 352–363, 2013.

[Cu15a] Curticapean, Radu: Block Interpolation: A Framework for Tight Exponential-Time Counting Complexity. In: ICALP 2015. S. 380–392, 2015.

[Cu15b] Curticapean, Radu: The simple, little and slow things count: on parameterized counting complexity. Dissertation, Universität des Saarlandes, 2015.

[Cu16] Curticapean, Radu: Parity Separation: A Scientifically Proven Method for Permanent Weight Loss. In: ICALP 2016. 2016.

[CX15] Curticapean, Radu; Xia, Mingji: Parameterizing the Permanent: Genus, Apices, Minors, Evaluation Mod 2k. In: FOCS 2015. S. 994–1009, 2015.

[De14] Dell, Holger; Husfeldt, Thore; Marx, Dániel; Taslaman, Nina; Wahlen, Martin: Exponential Time Complexity of the Permanent and the Tutte Polynomial. ACM Transactions on Algorithms, 10(4):21, 2014.

[FG04] Flum, Jörg; Grohe, Martin: The parameterized complexity of counting problems. SIAM Journal on Computing, (4):892–922, 2004.

[FG06] Flum, J.; Grohe, M.: Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[GL98] Galluccio, Anna; Loebl, Martin: On the Theory of Pfaffian Orientations. I. Perfect Matchings and Permanents. Electronic Journal of Combinatorics, 6, 1998.

[GS10] Guillemot, Sylvain; Sikora, Florian: Finding and Counting Vertex-Colored Subtrees. In: MFCS 2010. S. 405–416, 2010.

[HO02] Hemaspaandra, Lane A.; Ogihara, Mitsunori: The Complexity Theory Companion. Springer, 2002.

[IPZ01] Impagliazzo, Russell; Paturi, Ramamohan; Zane, Francis: Which problems have strongly exponential complexity? J. Computer and Sys. Sci., 63(4):512–530, 2001.

[JSV04] Jerrum, Mark; Sinclair, Alistair; Vigoda, Eric: A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. J. ACM, 51(4):671–697, 2004.

[Ka61] Kasteleyn, Pieter W.: The statistics of dimers on a lattice: I. The number of dimer arrangements on a quadratic lattice. Physica, 27(12):1209 – 1225, 1961.

[Ka67] Kasteleyn, Pieter W.: Graph Theory and Crystal Physics. In: Graph Theory and Theoretical Physics, S. 43–110. Academic Press, 1967.

[Li74] Little, Charles: An Extension of Kasteleyn's method of enumerating the 1-factors of planar graphs. In: Combinatorial Mathematics, LNCS, S. 63–72. 1974.

[RS03] Robertson, Neil; Seymour, Paul D.: Graph Minors. XVI. Excluding a non-planar graph. J. Comb. Theory, Ser. B, 89(1):43 – 76, 2003.

[RS04] Robertson, Neil; Seymour, Paul D.: Graph Minors. XX. Wagner's conjecture. J. Comb. Theory, Ser. B, 92(2):325–357, 2004.

[STW14] Straub, Simon; Thierauf, Thomas; Wagner, Fabian: Counting the Number of Perfect Matchings in K_5-Free Graphs. In: CCC 2014. S. 66–77, 2014.

[TF61] Temperley, H. N. V.; Fisher, Michael E.: Dimer problem in statistical mechanics - an exact result. Philosophical Magazine, 6(68):1478–6435, 1961.

[To91] Toda, Seinosuke: PP is as hard as the polynomial-time hierarchy. SIAM Journal on Computing, 20(5):865–877, 1991.

[UBK13] Ugander, Johan; Backstrom, Lars; Kleinberg, Jon M.: Subgraph frequencies: mapping the empirical and extremal geography of large graph collections. In: WWW 2013. S. 1307–1318, 2013.

[Va79] Valiant, Leslie G.: The complexity of computing the permanent. Theoret. Comput. Sci., 8(2):189–201, 1979.

[Va08] Valiant, Leslie G.: Holographic Algorithms. SIAM J. Comput., 37(5):1565–1594, 2008.

[WW13] Williams, Virginia Vassilevska; Williams, Ryan: Finding, Minimizing, and Counting Weighted Subgraphs. SIAM J. Comput., 42(3):831–854, 2013.

Radu Curticapean studierte an der Universität des Saarlandes in Saarbrücken Informatik, bis er sämtliche in der Mensa angebotenen Gerichte auswendig kannte und 2015 unter der Betreuung von Prof. Dr. Markus Bläser promovierte. Dann zog es ihn nach Budapest zum SZTAKI-Institut der ungarischen Akademie der Wissenschaften, unterbrochen von zwei Semestern in Kalifornien, in denen er am Simons-Institut für theoretische Informatik in Berkeley arbeitete.