**Gottfried Wilhelm**
**Leibniz Universität Hannover**
**Fakultät für Elektrotechnik und Informatik**
**Institut für Theoretische Informatik**

# Inkrementelle Berechnungskomplexität

## Bachelorarbeit

im Studiengang B. Sc. Informatik

von

**Kamillo Hugh Ferry**
**Matrikelnr: 3208590**

**Prüfer: Prof. Dr. Vollmer**
**Zweitprüfer: PD Dr. Meier**
**Betreuer: PD Dr. Meier**

**Hannover, 15. August 2020**

# Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 15. August 2020

_
Kamillo Hugh Ferry

# Danksagung

Zunächst möchte ich meiner Familie für das akademisch fruchtbare Umfeld und die Unterstüzung danken, wann immer ich wieder einmal voller Begeisterung ein Thema verfolge (und für das offene Gehör, falls diese Begeisterung ihre Aufmerksamkeit fordert).

Desweiteren möchte ich mich bei meinem Betreuer Arne Meier bedanken, überhaupt für die Betreuung dieser Arbeit und konkret für seine Anmerkungen zum wissenschaftlichen Schreiben, zur korrekten Typographie und zum Umgang mit LaTeX. Unsere Zusammenarbeit hat mir sehr große Freude bereitet und das Interesse am wissenschaftlichen Arbeiten noch weiter vergrößert.

Weiterhin möchte ich mich bei Daniel, Florian, Jan, Jannik, Kay, Marcel, Sabrina, Stefan und Vivian bedanken. Eure aufmerksamen Augen und kritischen Anmerkungen haben dieser Arbeit den verdienten Feinschliff verpasst.

Und zuletzt möchte ich mich bei meiner Lerngruppe *Coronaphase (geb. Klausurenphase) Survival* bedanken. So wie Georg Wilhelm Friedrich Hegels *Grundlinien der Philosophie des Rechts* durch dessen Randbemerkungen und die Notizen seiner Schüler bereichert wurde, so haben Eure Erfahrungen mich bis jetzt sicher durch das Abenteuer Bachelorstudium gebracht und werden mir auch mit Sicherheit weiterhin eine große Hilfe sein.

# Inhaltsverzeichnis

# Kurzfassung

Oftmals müssen Algorithmen nicht nur einmalig ein Objekt verarbeiten, sondern auch auf viele kleine Änderungen reagieren, da das betreffende Objekt z. B. durch einen Benutzer erst Schritt für Schritt aufgebaut wird.

In diesem Kontext ist es wünschenswert, dass diese Änderungen möglichst schnell verarbeitet werden. Jedoch wird ein Algorithmus in der klassischen Komplexitätstheorie nur hinsichtlich seiner statischen Effizienz analysiert, also wie schnell er ein großes, unveränderliches Objekt verarbeitet.

Wir betrachten daher inkrementelle Probleme als eine Erweiterung parametrisierter Probleme, für die zwei zusätzliche Eingaben in Form von Änderungen an Instanzen $\delta$ und einem Zeugen $S$ (neben der üblichen Instanz $x$ und dem Parameter $k$) definiert werden.

In dieser Arbeit vergleichen wir zunächst die Komplexität statischer und inkrementeller parametrisierter Probleme. Wir kommen dabei zum Ergebnis, dass einige Probleme inkrementell leicht zu handhaben sind, andere dagegen genauso schwer wie ihr statisches Pendant bleiben. Ausschlaggebend hierfür sind die Klassen von Änderungen, auf die ein Algorithmus reagieren muss.

Außerdem betrachten wir das für festgehaltene Parameter effizient lösbare Problem Vertex-Cover und präsentieren einen Algorithmus, der speziell auf den inkrementellen Kontext ausgelegt ist, als Verbesserung zum statischen Algoritmus.

# Notationen

Mit $\mathbb{N}_0$ bezeichnen wir die natürlichen Zahlen einschließlich Null, also

$$\mathbb{N}_0 = \{\, 0, 1, 2, \dots \}.$$

Entsprechend bezeichne $\mathbb{N} = \mathbb{N}_0 \setminus \{0\}$.

Für die *Menge der Polynome in der Unbekannten $X$ mit Koeffizienten in $M$* für eine Menge $M$ schreiben wir $M[X]$, also

$$M[X] = \left\{ \sum_{i=0}^{n} a_i X^i \;\middle|\; a_i \in M, n \in \mathbb{N} \right\}.$$

Dann bezeichnet der *Grad* $\deg(p)$ eines Polynoms $p$ das größte $i \in \mathbb{N}$, für das $a_i \neq 0$.

Für einen Graphen $G = (V, E)$ und eine Knotenteilmenge $V' \subseteq V$ bezeichnen wir mit dem von $V'$ *induzierten Teilgraphen* den Graphen $H = (V', E \cap V' \times V')$. $N(v)$ für einen Knoten $v$ sei die *offene Nachbarschaft* von $v$, also $\{\, u \in V \mid \{v, u\} \in E\}$, und $N[v]$ die *geschlossene Nachbarschaft*, also $N(v) \cup \{v\}$.

Für aussagenlogische Formeln definieren wir induktiv folgende Formelklassen.

$$\begin{aligned}
\Gamma_{0,d} &= \{\, \lambda_1 \wedge \lambda_2 \wedge \dots \wedge \lambda_c \mid 1 \leq c \leq d, \lambda_i \text{ Literale} \,\} \\
\Delta_{0,d} &= \{\, \lambda_1 \vee \lambda_2 \vee \dots \vee \lambda_c \mid 1 \leq c \leq d, \lambda_i \text{ Literale} \,\} \\
\Gamma_{t+1,d} &= \left\{ \bigwedge_{i \in I} \delta_i \;\middle|\; \delta_i \in \Delta_{t,d}, I \neq \emptyset \text{ endlich} \right\} \\
\Delta_{t+1,d} &= \left\{ \bigvee_{i \in I} \gamma_i \;\middle|\; \gamma_i \in \Gamma_{t,d}, I \neq \emptyset \text{ endlich} \right\}
\end{aligned}$$

Für bekannte Formelklassen wie $\mathsf{KNF}$ oder $\mathsf{DNF}$ erhalten wir, dass $\mathsf{KNF} \mathrel{\widehat{=}} \Gamma_{2,1}$, $\mathsf{DNF} \mathrel{\widehat{=}} \Delta_{2,1}$ und für die entsprechenden Einschränkungen $\mathsf{d\text{-}DNF} \mathrel{\widehat{=}} \Delta_{1,\mathsf{d}}$ und $\mathsf{d\text{-}KNF} \mathrel{\widehat{=}} \Gamma_{1,\mathsf{d}}$. Außerdem schreiben wir $\Phi^+$ für die Unterklasse von $\Phi \in \{\, \Gamma, \Delta \,\}$, deren Formeln nur positive Literale enthalten, und $\Phi^-$ für die Unterklasse von $\Phi$, deren Formeln nur negierte Literale enthalten.

Ein *Boole'scher Schaltkreis* ist ein gerichteter, azyklischer Graph, bei dem die inneren Knoten mit Boole'schen Funktionen oder Familien von Boole'schen Funktionen, Blätter mit Konstanten oder einer Eingabevariablen und ein Knoten als Ausgabe markiert werden. Wir bezeichen mit $\mathsf{CIRC}$ die Menge aller Boole'schen Schaltkreise.

# 1 Einleitung

In der klassischen Komplexitätstheorie werden üblicherweise statische Probleme betrachtet. Das bedeutet, dass ein entsprechender Algorithmus einmalig eine Eingabe bekommt, die er sinnvoll verarbeitet, und am Ende ein Ergebnis ausgibt.

Viele relevante Berechnungsaufgaben folgen jedoch nicht diesem Schema. Stattdessen wird ein Objekt über eine gewisse Zeitspanne verarbeitet und dabei kontinuierlich verändert. Daher ist die Berechnung, die ein Algorithmus durchführt, auch nicht einmalig, sondern wird für kleine Änderungen oft wiederholt. Damit exisitieren aber auch Informationen aus früheren Berechnungen, die möglicherweise verwertet werden können.

An folgendem Beispiel verdeutlichen wir die Bedeutung inkrementeller Probleme in unserem Alltag.

Wir stellen uns vor, dass wir ein LaTeX-Dokument vor uns haben, z. B. zu einer Abschlussarbeit, die wir verfassen wollen. Nun kommen wir irgendwann an den Punkt, an dem wir dieses Dokument kompilieren wollen, um die Früchte unserer Arbeit zu begutachten. Da wir eine besonders umfangreiche Arbeit verfasst haben, benötigt dieser Vorgang seine Zeit.

Fassen wir diese Situation als ein Problem im Sinne der klassischen Komplexitätstheorie auf, dann hätten wir unsere Arbeit in einem Stück verfasst, nur ganz am Ende das gesamte Werk einmal kompiliert, und uns für die Dauer dieses Vorgangs in Abhängigkeit des Umfangs unseres Dokuments interessiert.

Nun ist es möglich, aber doch unwahrscheinlich, dass wir ein solches Werk am Stück verfassen. In der Realität tippen wir ein paar Passagen und kompilieren dann das unfertige Dokument, um unseren Zwischenstand zu betrachten, nehmen dann weitere Änderungen vor und wiederholen diesen Zyklus bis zum Schluss.

In genau solch einer Situation ist es sinnvoll, dieses Problem als ein inkrementelles Problem aufzufassen. Zum einen haben wir Änderungen an unserem Dokument auf die unser LaTeX-Compiler reagieren muss und zum anderen bleiben nach einem Kompilationsvorgang Nebenprodukte übrig.

Denn oft ändern wir zwischen zwei Kompilationsvorgängen nur wenige Zeilen, die Änderungen sind also vergleichsweise klein. Daher erwarten wir auch, dass unsere neue Ausgabe relativ schnell aktualisiert wird und sich die Nebenprodukte teilweise wiederverwenden lassen. Damit haben wir auch bereits alle Aspekte eines inkrementellen Problems beisammen.

Die Bedeutung inkrementeller Probleme wurde auch bereits in der Literatur erkannt, denn es existiert bereits eine breite Auswahl an Resultaten, obwohl es sich bei der

inkrementellen Komplexitätstheorie um ein relativ junges Gebiet handelt, da nach einer Literaturübersicht von Ramalingam und Reps [RR93] die ersten Arbeiten hierzu um etwa 1990 verfasst wurden.

Aktuelle Arbeit an inkrementellen Algorithmen umfasst unter anderem die inkrementelle Berechnung von Voronoi-Diagrammen [HH09] oder die inkrementelle Berechnung von PageRank für Graphen [Des+05]. PageRank bezeichnet einen Algorithmus aus dem Bereich der Suchmaschinenoptimierung zur Berechnung der Relevanz verlinkter Dokumente für eine gegebene Suchanfrage. Die grundlegende Idee basiert darauf, dass eine Seite relevant ist, falls viele Seiten mit eigener hoher Relevanz auf diese verlinken [Pag+99].

Außerdem existieren auch Ansätze für inkrementelle Komplexitätsklassen und Berechnungsmodelle, wie sie bei Miltersen et al. [Mil+94] zu finden sind. Und auch im Rahmen der deskriptiven Komplexitätstheorie findet sich bei Immerman und Patnaik [PI97] ein Ansatz, die Komplexität inkrementeller Probleme greifbar zu machen.

In dieser Arbeit betrachten wir für grundlegende Probleme der Komplexitätstheorie die zugehörige inktrementelle Variante und nutzen dazu das Gerüst der parametrisierten Komplexitätstheorie.

Daher findet sich in Kapitel 2 eine Einführung in die Theorie der parametrisierten Komplexität und die nötigen Komplexitätsklassen sowie den verwendeten Reduktionsbegriff. Außerdem definieren wir formal den Begriff des *inkrementellen Problems*. Diese Definitionen richten sich dabei nach der Arbeit von Mans und Mathieson [MM17], bedienen sich jedoch einiger Definitionen nach Creignou et al. [Cre+19] und Goldreich [Gol06] für eine stabile formale Grundlage.

In Kapitel 3 beschäftigen wir uns mit statischen parametrisierten Problemen und ihrer Komplexität. Dazu werden einerseits Resultate von Mans und Mathieson [MM17, Abschnitt 2] zu klassischen, parametrisierten Problemen betrachtet und andererseits Resultate aus der Literatur hinzugezogen [FG06][DF13].

In Kapitel 4 betrachten wir dann die Komplexität der entsprechenden inkrementellen Varianten. Die Resultate orientieren sich an der Arbeit von Mans und Mathieson [MM17, Abschnitt 3] bezüglich inkrementeller Probleme und werden hier in ausführlicherer Form gebracht.

Zum Abschluss betrachten wir in Kapitel 5 einen Ausblick über die Literatur zu inkrementellen Problemen.

# 2 Grundlagen

In diesem Kapitel geben wir einen Einblick in die parametrisierte Komplexitätstheorie, um darauf aufbauend inkrementelle Probleme zu betrachten. Dazu klären wir grundlegende Begrifflichkeiten, bevor wir uns im nächsten Kapitel mit konkreten Problemen beschäftigen.

## 2.1 Grundlagen parametrisierter Komplexität

Die hier verwendete Definition von parametrisierten Problemen richtet sich nach dem Buch von Downey und Fellows [DF13, Abschnitt 2.1].

**Definition 2.1.1** (Parametrisierte Probleme)**.** Sei $\Sigma$ ein endliches Alphabet. Eine Sprache $L \subseteq \Sigma^* \times \mathbb{N}$ bezeichnen wir als *parametrisiertes Problem*.

Man führt in der parametrisierten Komplexitätstheorie den Parameter als zusätzliche, problemspezifische Größe ein, anhand derer die Instanzen eines Problems klassifiziert werden sollen, um den Ressourcenverbrauch von Algorithmen nicht nur in *Abhängigkeit der Eingabelänge* zu betrachten, wie in der klassischen Komplexitätstheorie üblich.

Hierbei erhofft man sich, nicht einfach nur Schwierigkeit von Problemen festzustellen, sondern auch zu erfassen, was genau ein gewisses Problem schwierig macht [DF99]. Man sucht dabei auch nach praxisrelevanten Parametern, die die Schwierigkeit eines Problemes genauer beschreiben.

*Beispiel.* Die parametrisierte Variante des Erfüllbarkeitsproblems für aussagenlogische Formeln ist wie folgt definiert.

p-SAT

| | |
|---|---|
| **Eingabe:** | Eine aussagenlogische Formel $\phi$ |
| **Parameter:** | Anzahl der Variablen in $\phi$ |
| **Frage:** | Hat $\phi$ eine erfüllende Belegung? |

Dazu sei gesagt, dass der Parameter *Anzahl der Variablen* $k$ einer Formel $\phi$ nicht sehr praxisrelevant ist, da auch der Brute-Force-Algorithmus, der alle Belegungen von $\phi$ ausprobiert, eine Laufzeit von höchstens $2^k \cdot |\phi|$ hat. Er genügt damit bereits unserem Begriff effizienter parametrisierter Algorithmen, den wir im folgenden Verlauf (Abschnitt 2.1.1) einführen werden [Bie+09, Beispiel 13.3.1].

Es sind daher solche Parameter von größerem Interesse, bei denen für einen festen Wert $k$ Formeln mit beliebig großer Zahl an Variablen auftreten können. Wir können zum Beispiel als Teil des Parameters eine Vorbearbeitung einer gegebene Formel $\phi$ durch einen Algorithmus betrachten, sodass die resultierende Formel $\phi'$ eine gewählte Eigenschaft erfüllt [Bie+09, Abschnitt 13.3.3].

Parametrisierungen treten auch natürlich auf, wie am Beispiel von p-Vertex-Cover zu sehen ist:

*Beispiel.* Für einen Graphen $G = (V, E)$ bezeichnen wir mit einer *Knotenüberdeckung* eine Teilmenge $V' \subseteq V$, sodass für alle Kanten $\{ u, v \} \in E$ gilt, dass $u \in V'$ oder $v \in V'$. Dann ist das Problem, eine solche Knotenüberdeckung zu finden, wie folgt definiert.

p-Vertex-Cover
**Eingabe:** Graph $G = (V, E)$, $k \in \mathbb{N}$
**Parameter:** $k$
**Frage:** Hat $G$ eine Knotenüberdeckung der Größe $\leq k$?

Wir sehen, dass sich p-Vertex-Cover von der nicht-parametrisierten Variante nur um die explizite Benennung des Parameters unterscheidet.

Man erkennt an der Definition der Parametrisierung, dass die Instanzen eines Problemes partitioniert werden. Diese Partitionen bezeichnet man üblicherweise als *Scheiben*, formal nachstehend definiert.

**Definition 2.1.2.** Sei $L \subseteq \Sigma^* \times \mathbb{N}$ ein parametrisiertes Problem und $k \in \mathbb{N}$. Dann nennen wir das klassische Problem $L_k$ eine *Scheibe von* $L$ mit

$$L_k = \{ (x, \ell) \in L \mid \ell = k \}.$$

### 2.1.1 Fixed-parameter tractability

So wie es in der klassischen Komplexitätstheorie den Begriff der *effizienten Lösbarkeit* gibt, existiert auch in der parametrisierten Komplexitätstheorie folgender vergleichbarer Begriff [DF13, Definition 2.1.1].

**Definition 2.1.3** (Fixed-parameter tractability)**.** Sei $L \subseteq \Sigma^* \times \mathbb{N}$ ein parametrisiertes Problem. Dann nennen wir $L$ *fixed-parameter tractable*, falls es einen deterministischen Algorithmus gibt, der $(x, k) \overset{?}{\in} L$ entscheidet und Laufzeit

$$f(k) \cdot |x|^{\mathcal{O}(1)}$$

hat, für eine berechenbare Funktion $f \colon \mathbb{N} \to \mathbb{N}$. Einen solchen Algorithmus bezeichnen wir auch als *fpt-Algorithmus*.

**Definition 2.1.4.** FPT ist die Klasse aller Probleme, die fixed-parameter-tractable sind.

*Beispiel.* Wie bereits im voherigen Abschnitt angesprochen, benötigt der triviale Brute-Force-Algorithmus, der entscheidet, ob eine Formel $\phi$ erfüllbar ist, höchstens $2^k \cdot |\phi|$ Schritte, ist also ein fpt-Algorithmus. Es gilt daher, dass $\mathsf{p\text{-}SAT} \in \mathsf{FPT}$.

Interessanterweise können wir die Multiplikation in der Definition der Klasse FPT durch eine Addition ersetzen und erhalten trotzdem die gleiche Komplexitätsklasse, wie folgendes Lemma zeigt, das leicht gekürzt aus dem Buch von Flum und Grohe entnommen wurde.

**Lemma 2.1.5** ([FG06, Prop. 1.34])**.** *Sei $L \subseteq \Sigma^* \times \mathbb{N}$ ein parametrisiertes Problem. $L$ ist genau dann in* FPT*, wenn ein Algorithmus existiert, der $(x,k) \overset{?}{\in} L$ entscheidet, mit Laufzeit*

$$f(k) + |x|^{\mathcal{O}(1)}$$

*für eine berechenbare Funktion $f\colon \mathbb{N} \to \mathbb{N}$.*

*Beweis.* Sei $p \in \mathbb{N}_0[X]$ ein Polynom.

„$\Longrightarrow$“: Wegen $a \cdot b \leq a^2 + b^2$ gilt:

$$f(k) \cdot p(|x|) \leq \underbrace{(f(k))^2}_{h(k)} + \underbrace{(p(|x|))^2}_{q(|x|)}$$

Da $f$ berechenbar ist, ist $h$ ebenso berechenbar und da $p$ ein Polynom ist, ist auch $q$ ein Polynom.

„$\Longleftarrow$“: Sei $d = \deg(p)$ der Grad des Polynoms $p$. Es gilt:

$$\begin{aligned} f(k) + p(|x|) &\leq f(k) + p(|x| + k) \\ &\leq f(k) + (|x| + k)^{d+1} \\ &\leq f(k) + (k \cdot (|x| + 1))^{d+1} \\ &\leq \underbrace{(f(k) + k^{d+1})}_{h(k)} \cdot \underbrace{(|x| + 1)^{d+1}}_{q(|x|)} \end{aligned}$$

Ein Algorithmus mit Laufzeit $f(k) + p(|x|)$ ist also auch ein fpt-Algorithmus. □

Die Bezeichnung *fixed-parameter tractable* weißt darauf hin, dass wir für festgehaltene Parameterwerte $k$ ein effizient lösbares Problem erhalten.

**Satz 2.1.6** ([FG06, Prop. 1.11])**.** *Sei $L \in \mathsf{FPT}$. Dann gilt für alle $k \in \mathbb{N}$, dass $L_k \in \mathsf{P}$.*

*Beweis.* Sei $k \in \mathbb{N}$, $x \in \Sigma^*$ und $f\colon \mathbb{N} \to \mathbb{N}$ eine berechenbare Funktion. Da $L \in \mathsf{FPT}$, ist $(x,k) \overset{?}{\in} L_k$ entscheidbar in Zeit $f(k) \cdot |x|^{\mathcal{O}(1)} \subseteq |x|^{\mathcal{O}(1)}$, denn $f(k)$ ist für festes $k$ ebenfalls eine Konstante. Damit ist $L_k \in \mathsf{P}$. □

Die Umkehrung dieser Aussage trifft jedoch nicht zu, also

$$L_k \in \mathsf{P} \text{ für alle } k \in \mathbb{N} \not\Longrightarrow L \in \mathsf{FPT}.$$

Stattdessen führt uns dies zur Klasse XP. Auf diesen Sachverhalt gehen wir in Abschnitt 2.1.2 noch ein.

Ebenso gibt es auch einen passenden parametrisierten Reduktionsbegriff [DF13, Definition 20.2.1].

**Definition 2.1.7** (*fpt*-Reduktionen)**.** Seien $L_1 \subseteq \Sigma_1^* \times \mathbb{N}$ und $L_2 \subseteq \Sigma_2^* \times \mathbb{N}$ zwei parametrisierte Probleme. Dann bezeichnen wir mit einer *fpt-Reduktion* von $L_1$ auf $L_2$ eine Abbildung

$$\begin{aligned} \varphi\colon \Sigma_1^* \times \mathbb{N} &\to \Sigma_2^* \times \mathbb{N} \\ (x,k) &\mapsto (x',k') \end{aligned}$$

mit folgenden Eigenschaften:

1. Für alle $(x,k) \in \Sigma_1^* \times \mathbb{N}$ gilt: $(x,k) \in L_1 \iff (x',k') \in L_2$
2. $\varphi$ ist berechenbar durch einen fpt-Algorithmus.
3. Es gibt eine berechenbare Funktion $g : \mathbb{N} \to \mathbb{N}$, sodass $k' \leq g(k)$.

Falls eine solche Abbildung $\varphi$ existiert, schreiben wir $L_1 \leq^{fpt} L_2$. Falls zusätzlich $L_2 \leq^{fpt} L_1$ gilt, schreiben wir $L_1 \equiv^{fpt} L_2$.

Die ersten beiden Bedingungen sind übliche Anforderungen an eine Reduktionen [Sip06, Definitionen 5.20 und 7.29]. Mit der dritten Bedingung fordern wir, dass der Parameter unserer neuen Instanzen von oben durch den ursprünglichen Parameter beschränkt ist, womit wir erreichen, dass die Klasse FPT abgeschlossen unter *fpt*-Reduktionen ist, wie wir im Folgenden zeigen werden.

Insgesamt sehen wir, dass fpt-Reduktionen auch tatsächlich die gewünschten Eigenschaften einer Reduktion erfüllen. Der Beweis hierzu wurde aus dem Buch von Flum und Grohe entnommen, wir bringen ihn jedoch in ausführlicherer Form.

**Lemma 2.1.8** ([FG06, Lemma 2.3])**.**

1. $\leq^{fpt}$ *ist reflexiv.*
2. $\leq^{fpt}$ *ist transitiv.*
3. FPT *ist abgeschlossen unter fpt-Reduktionen.*

*Beweis.*

1. Ist $L \subseteq \Sigma^* \times \mathbb{N}$ ein parametrisiertes Problem, so gilt $L \leq^{fpt} L$ via der Identitätsfunktion, also

$$\begin{aligned} \mathrm{id}_L\colon \Sigma^* \times \mathbb{N} &\to \Sigma^* \times \mathbb{N} \\ (x,k) &\mapsto (x,k). \end{aligned}$$

Offensichtlich gilt, dass $(x,k) \in L \iff \mathrm{id}_L(x,k) = (x,k) \in L$.

$\mathrm{id}_L$ ist berechenbar in Laufzeit $|x| + \log(k)$, also in fpt-Laufzeit nach Lemma 2.1.5.

Ebenso gilt für $g : \mathbb{N} \to \mathbb{N},\ n \mapsto n$, dass $k \leq g(k) = k$.

2. Seien $L_i \subseteq \Sigma_i^* \times \mathbb{N},\ i = 1,2,3$ parametrisierte Probleme, sodass $L_1 \leq^{fpt} L_2$ via

$$\begin{aligned} \varphi\colon \Sigma_1^* \times \mathbb{N} &\to \Sigma_2^* \times \mathbb{N} \\ (x,k) &\mapsto (x',k'), \end{aligned}$$

wobei $k' \leq h(k)$ für eine berechenbare Funktion $h\colon \mathbb{N} \to \mathbb{N}$, wobei $\varphi$ für eine berechenbare Funktion $f$ und ein Polynom $p$ in fpt-Laufzeit berechenbar ist und $L_2 \leq^{fpt} L_3$ via

$$\begin{aligned} \psi\colon \Sigma_2^* \times \mathbb{N} &\to \Sigma_3^* \times \mathbb{N} \\ (x,k) &\mapsto (x',k'), \end{aligned}$$

wobei $k' \leq i(k)$ für eine berechenbare Funktion $i\colon \mathbb{N} \to \mathbb{N}$, wobei $\psi$ für eine berechenbare Funktion $g$ und ein Polynom $q$ in fpt-Laufzeit berechenbar ist.

Dann gilt $L_1 \leq^{fpt} L_3$ vermöge der Funktion $\xi = \psi \circ \varphi$, also

$$\begin{aligned} \xi\colon \Sigma_1^* \times \mathbb{N} &\to \Sigma_3^* \times \mathbb{N} \\ (x,k) &\mapsto \psi(\varphi(x,k)) = (x',k'), \end{aligned}$$

denn $\xi$ ist in folgender Laufzeit berechenbar

$$g(f(k)) \cdot q(p(|x|)).$$

Für berechenbare Funktionen $f$ und $g$ ist die Komposition ebenso berechenbar und für zwei Polynome ist die Komposition ebenso ein Polynom. Ebenfalls gilt, dass

$$(x,k) \in L_1 \iff \varphi(x,k) \in L_2 \iff \psi(\varphi(x,k)) = \xi(x,k) \in L_3.$$

Ebenso ist der Parameter $\hat{k}$ durch den Parameter der ursprünglichen Instanz beschränkt, denn

$$k' \leq i(h(k)).$$

3. Seien $L_1 \subseteq \Sigma_1^* \times \mathbb{N}$ und $L_2 \subseteq \Sigma_2^* \times \mathbb{N}$ zwei parametrisierte Probleme mit $L_1 \leq^{fpt} L_2$ via

$$\begin{aligned} \varphi\colon \Sigma_1^* \times \mathbb{N} &\to \Sigma_2^* \times \mathbb{N} \\ (x,k) &\mapsto (x',k'), \end{aligned}$$

wobei $\varphi$ für eine berechenbare Funktion $f$ und ein Polynom $p$ in fpt-Laufzeit berechenbar ist und $k' \leq g(k)$ für eine berechenbare Funktion $g$. Sei $M'$ ein Algorithmus, der $L_2$ in höchstens $f'(k') \cdot p'(|x'|)$ Schritten entscheidet.

Wir konstruieren einen Algorithmus $M$, der $L_1$ entscheidet, indem wir die Reduktion $\varphi$ nutzen, um eine Instanz $(x', k') = \varphi(x, k)$ für $L_2$ zu produzieren. Damit entscheiden wir die Frage $(x', k') \overset{?}{\in} L_2$ mittels $M'$, womit wir ebenso $(x, k) \overset{?}{\in} L_1$ beantworten können.

Für die Laufzeit von $M$ ergibt sich

$$\begin{aligned}
&\overbrace{f(k) \cdot p(|x|)}^{\text{Laufzeit von } \varphi} + \overbrace{f'(k') \cdot p'(|x'|)}^{\text{Laufzeit von } M'} \\
&\leq f(k) \cdot p(|x|) + f'(g(k)) \cdot p'(f(k) \cdot p(|x|)) \\
&\leq f(k) \cdot p(|x|) + f'(g(k)) \cdot p'(f(k)) \cdot p'(p(|x|))) \\
&\leq \underbrace{(f(k) + f'(g(k)) \cdot p'(f(k)))}_{h(k)} \cdot \underbrace{p(|x|) \cdot p'(p(|x|))}_{q(|x|)},
\end{aligned}$$

also eine für einen fpt-Algorithmus erlaubte Laufzeit, daher $L_1 \in \mathsf{FPT}$. □

**Definition 2.1.9** ([FG06, Abschnitt 2.1])**.** Für ein parametrisiertes Problem $L$ schreiben wir die *Menge der auf $L$ reduzierbaren Probleme* (mittels einer fpt-Reduktion) als

$$[L]^{fpt} := \{ L' \mid L' \leq^{fpt} L \}.$$

Ist $\mathcal{C}$ eine Klasse parametrisierter Probleme, dann ist

$$[\mathcal{C}]^{fpt} := \bigcup_{L \in \mathcal{C}} [L]^{fpt}$$

der *Abschluss von $\mathcal{C}$ unter fpt-Reduktionen.*

**Definition 2.1.10** ([FG06, Abschnitt 2.1])**.** Ein parametrisiertes Problem $L$ heißt *vollständig unter fpt-Reduktionen* für eine Komplexitätsklasse $\mathcal{C}$ (oder $\mathcal{C}$-*vollständig*), falls $L \in \mathcal{C}$ und für alle $L' \in \mathcal{C}$ gilt, dass $L' \leq^{fpt} L$. Alternativ können wir auch schreiben, dass $[L]^{fpt} = \mathcal{C}$.

### 2.1.2 Die Klasse XP

Wie wir bereits im Zusammenhang mit Satz 2.1.6 angemerkt haben, gilt die Umkehrung der dortigen Aussage nicht.

*Bemerkung* 2.1.11. $L_k \in \mathsf{P}$ für alle $k \in \mathbb{N} \not\Rightarrow L \in \mathsf{FPT}$.

Betrachtet man die Klasse der parametrisierten Probleme, für die jede Scheibe in deterministischer Polynomialzeit entscheidbar ist, erhält man $\mathsf{XP}_{nu}$.

**Definition 2.1.12** ([FG06, Definition 2.19])**.** $\mathsf{XP}_{nu}$ ist die Klasse aller parametrisierten Probleme $L$, für die $L_k \in \mathsf{P}$ für alle $k \in \mathbb{N}$.

Hierbei stoßen wir auf die Tatsache, dass diese Definition *nicht uniform* ist, da jede Scheibe eines Problems von einer jeweils anderen Turingmaschine (TM) entschieden werden kann.

*Beispiel.* Sei $L = \{ (1^k, k+1) \mid k \in K \}$, wobei $K$ das spezielle Halteproblem bezeichnet. Jede Scheibe $L_k$ besteht aus genau einer Instanz. Daher genügt es, je Scheibe eine TM anzugeben, die entweder immer akzeptiert, falls $k \in K$, oder immer verwirft, falls $k \notin K$. Also ist jedes $L_k \in \mathsf{P}$ und $L \in \mathsf{XP}_{nu}$.

Es stellt sich also heraus, dass nicht-entscheidbare Probleme in $\mathsf{XP}_{nu}$ liegen können, weshalb wir die folgende, uniforme Definition von $\mathsf{XP}$ betrachten.

**Definition 2.1.13** ([FG06, Definition 2.22])**.** Die Klasse $\mathsf{XP}$ besteht aus allen parametrisierten Problemen $L$, für die es eine berechenbare Funktion $f\colon \mathbb{N} \to \mathbb{N}$ und eine DTM $M$ gibt, die $L$ entscheidet und dabei höchstens $|x|^{f(k)}$ Schritte benötigt.

Tatsächlich sieht man, dass $\mathsf{XP}$ intuitiv mit $\mathsf{EXP}$ aus der klassischen Komplexitätstheorie zu vergleichen ist, wie folgendes Beispiel für ein $\mathsf{XP}$-vollständiges Problem zeigt.

| p-EXP-DTM-HALT | |
|---|---|
| **Eingabe:** | DTM $M$, $x \in \Sigma^*$, $k \in \mathbb{N}$ |
| **Parameter:** | $k$ |
| **Frage:** | Akzeptiert $M$ auf Eingabe $x$ in weniger als $\lvert x\rvert^k$ Schritten? |

**Satz 2.1.14** ([FG06, Theorem 2.25])**.** p-EXP-DTM-HALT *ist* $\mathsf{XP}$*-vollständig unter fpt-Reduktionen.*

*Beweis.* p-EXP-DTM-HALT $\in \mathsf{XP}$ via einer TM $M'$, die $M$ auf Eingabe $x$ für $|x|^k$ Schritte simuliert und genau dann akzeptiert, falls $M$ während der Simulation akzeptiert.

Sei nun $L \in \mathsf{XP}$ über einer DTM $M'$, die $L$ in Laufzeit $|x|^{f(k)}$ entscheidet. Wir wählen eine berechenbare Funktion $g\colon \mathbb{N} \to \mathbb{N}$, sodass die Laufzeit von $M'$ höchstens $|x|^{g(k)}$ beträgt. Wir können nun $L$ auf p-EXP-DTM-HALT mittels folgender Abbildung reduzieren

$$\begin{aligned} \varphi\colon \Sigma^* \times \mathbb{N} &\to \Sigma^* \times \mathbb{N} \\ (x,k) &\mapsto (\langle M', x, g(k)\rangle, g(k)). \end{aligned}$$

Damit gilt $L \leq^{fpt}$ p-EXP-DTM-HALT und p-EXP-DTM-HALT ist $\mathsf{XP}$-vollständig. □

Insbesondere ist die nicht-parametrisierte Variante EXP-DTM-HALT $\mathsf{EXP}$-vollständig unter Polynomialzeitreduktionen. In der Literatur werden $\mathsf{XP}$-vollständige Probleme auch als „provable intractable“ bezeichnet [DF13, Kapitel 27.1].

**Korollar 2.1.15** ([FG06, Korollar 2.26])**.** FPT $\subsetneq$ XP

*Beweis.* Es gilt, dass FPT $\subseteq$ XP, da ein Problem, das o. B. d. A. in Laufzeit $f(k) \cdot |x|^c$ für ein $c \in \mathbb{N}$ entschieden werden kann, kann ebenfalls in Zeit $|x|^{g(k)}$ entschieden werden, für eine passende berechnebare Funktion $g(k)$.

Nehmen wir nun an, dass p-EXP-DTM-HALT wäre in FPT. Dann existiert ein Algorithmus, der p-EXP-DTM-HALT in Laufzeit $f(k) \cdot p(|x|)$ entscheidet, für eine berechenbare Funktion $f\colon \mathbb{N} \to \mathbb{N}$ und ein Polynom $p \in \mathbb{N}_0[X]$.

Insbesondere ist gerade für $d = \deg(p)$ jede Scheibe von p-EXP-DTM-HALT in DTIME$(n^d)$ entscheidbar. Damit liegt jedoch jede Scheibe p-EXP-DTM-HALT$_{d+1}$ in DTIME$(n^d)$.

Wegen Satz 2.1.14 beträgt die Laufzeit eines Algorithmus, der p-EXP-DTM-HALT$_{d+1}$ entscheidet, jedoch mindestens $|x|^{d+1}$, womit DTIME$(n^{d+1}) \subseteq$ DTIME$(n^d)$.

↯ Ein Widerspruch zum Zeithierarchiesatz. Und damit ist p-EXP-DTM-HALT $\notin$ FPT und FPT $\subsetneq$ XP. □

Um auf Bemerkung 2.1.11 eingangs dieses Abschnittes zurückzukommen, haben wir mit p-EXP-DTM-HALT tatsächlich ein Beispiel für ein Problem, das zwar scheibenweise in Polynomialzeit berechenbar ist, insgesamt jedoch nicht in FPT liegt.

### 2.1.3 W-Hierarchie

Während die Frage nach einem parametrisierten Analogon der Klasse P relativ einfach zu beantworten ist, ist die Wahl eines entsprechenden Kandidaten für nicht effizient lösbare parametrisierte Probleme schwerer [FG06, Kapitel 2].

Geht man analog zur Definition der Klasse NP vor und lässt in der Definition der Klasse FPT Nichtdeterminismus zu, so erhält man stattdessen die Klasse para-NP. Es stellt sich heraus, dass ein Problem bereits para-NP-vollständig ist, sobald eine Scheibe dieses Problems NP-vollständig bezüglich Polynomialzeitreduktionen ist [FG06, Korollar 2.16].

Ein Beispiel für ein para-NP-vollständiges Problem ist die Frage nach der $k$-Färbbarkeit eines Graphen [FG06, Beispiel 2.17].

p-Colorability

| | |
|---|---|
| **Eingabe:** | Graph $G = (V, E)$, $k \in \mathbb{N}$ |
| **Parameter:** | $k$ |
| **Frage:** | Existiert eine Funktion $f\colon V \to \{1, \ldots, k\}$ mit $f(u) \neq f(v)$ für alle $\{u, v\} \in E$? |

Bereits die klassische 3-Färbbarkeit ist NP-vollständig [Sto73] und damit ist p-Colorability para-NP-vollständig.

Eine wichtige Rolle bei der Betrachtung nicht-effizient lösbarer parametrisierter Probleme spielt die Klasse $\mathsf{W}[\mathsf{P}]$, die die Vereinigung einer ganzen Hierarchie von parametrisierten Komplexitätsklassen darstellt. $\mathsf{W}[\mathsf{P}]$ selbst erhalten wir, indem wir *fpt*-Algorithmen mit beschränkten Nichtdeterminismus betrachten.

**Definition 2.1.16** ([FG06, Definition 3.1.(1)])**.** Sei $L \subseteq \Sigma^* \times \mathbb{N}$ ein parametrisierte Problem. Eine NTM $M$ heißt *$k$-beschränkt*, falls für alle $(x,k) \in \Sigma^* \times \mathbb{N}$ jeder Berechnungspfad von $M$ auf Eingabe $(x,k)$ höchstens $h(k) \cdot \log|x|$ nichtdeterministische Verzweigungen hat, für eine berechenbare Funktionen $f\colon \mathbb{N} \to \mathbb{N}$.

**Definition 2.1.17** ([FG06, Definition 3.1.(2)])**.** Wir bezeichnen mit $\mathsf{W}[\mathsf{P}]$ die Klasse aller parametrisierten Probleme, die durch eine $k$-beschränkte NTM in *fpt*-Laufzeit entschieden werden können.

Wir können die zuvor betrachteten Komplexitätsklassen mit $\mathsf{W}[\mathsf{P}]$ in Beziehung setzen, wie wir in folgendem Satz sehen.

**Satz 2.1.18** ([FG06, Proposition 3.2])**.** $\mathsf{FPT} \subseteq \mathsf{W}[\mathsf{P}] \subseteq \mathsf{XP} \cap \mathsf{para\text{-}NP}$.

*Beweis.*

$\mathsf{FPT} \subseteq \mathsf{W}[\mathsf{P}]$*:* Ein deterministischer *fpt*-Algorithmus kann von einer NTM problemlos ebenfalls in *fpt*-Laufzeit ausgeführt werden. Eine solche NTM ist definitiv auch $k$-beschränkt, da für einen deterministischen Algorithmus keine nichtdeterministischen Verzweigungen benötigt werden.

$\mathsf{W}[\mathsf{P}] \subseteq \mathsf{para\text{-}NP}$*:* $\mathsf{para\text{-}NP}$ enthält gerade die Probleme, die in *fpt*-Laufzeit von einer NTM entschieden werden können. $\mathsf{W}[\mathsf{P}]$ hat als zusätzliche Bedingung nur die $k$-Beschränktheit einer solchen Maschine. $\mathsf{para\text{-}NP}$ enthält also alle Probleme aus $\mathsf{W}[\mathsf{P}]$.

$\mathsf{W}[\mathsf{P}] \subseteq \mathsf{XP}$*:* Für eine NTM $M$ mit $s$ Zuständen ist die Simulation von $h(k) \cdot \log n$ nichtdeterministischen Verzweigungen möglich in folgender Laufzeit

$$2^{\log\left(s^{h(k)\cdot\log n}\right)} = 2^{\log s \cdot h(k) \cdot \log n} = n^{\mathcal{O}(h(k)\cdot\log s)}.$$

Insbesondere ist die Anzahl der Zustände eine Konstante und der Exponent daher nur von $k$ abhängig.

□

Nun charakterisieren wir $\mathsf{W}[\mathsf{P}]$ auf eine ähnliche Art und Weise wie $\mathsf{NP}$ für klassische Probleme, nämlich als die Klasse der *fpt*-verifizierbaren Sprachen mit $k$-beschränkter Beweislänge.

**Lemma 2.1.19** ([FG06, Theorem 3.8])**.** *Für ein parametrisiertes Problem $L \subseteq \Sigma^* \times \mathbb{N}$ gilt genau dann $L \in \mathsf{W}[\mathsf{P}]$, wenn berechenbare Funktionen $f, h\colon \mathbb{N} \to \mathbb{N}$, ein Polynom $p \in \mathbb{N}_0[X]$ und eine Sprache $V \subseteq \Sigma^* \times \mathbb{N} \times \{0,1\}^*$ existieren, sodass*

1. $V \in \mathsf{FPT}$, *also* $(x,k,y) \overset{?}{\in} V$ *in Laufzeit* $f(k) \cdot p(|x|)$ *entscheidbar ist*

2. $(x,k,y) \in V \implies |y| = h(k) \cdot \log|x|$

3. *für alle* $x \in \Sigma^*$ *gilt, dass*

$$(x,k) \in L \iff \exists y \in \{0,1\}^* \colon (x,k,y) \in V$$

Intuitiv entspricht $y$ also dem Zertfikat aus der polynomiellen Überprüfbarkeit und anstatt einer Polynomialzeit-DTM benutzen wir eine DTM mit *fpt*-Laufzeit als *Verifizierer*. Da wir uns den akzeptierenden Berechnungspfad der $k$-beschränkten NTM als Zertifkat vorstellen können, soll die Länge des Zertifkats ebenfalls durch $k$ beschränkt sein.

*Beweis von Lemma 2.1.19.*

„$\implies$“: Sei $L \in \mathsf{W[P]}$ via $f', h', p', M'$ wie in Definition 2.1.16. Wir wissen, falls $x \in L$, dann akzeptiert $M'$ auf Eingabe $x$ mit höchstens $h'(k) \cdot \log|x|$ nichtdeterministischen Verzweigungen, und da es allerhöchstens so viele Verzweigungsmöglichkeiten wie Anzahl $s$ der Zustände von $M'$ gibt, können wir eine Verzweigung mit $\log s$-vielen Bits kodieren. Kodiert $y$ die nichtdeterministischen Verzweigungen auf einem Berechnungspfad, so ergibt sich $|y| = \log s \cdot h'(k) \cdot \log|x|$.

Nun können wir eine Sprache $Y \subseteq L \times \mathbb{N} \times \{0,1\}^*$ wie folgt definieren.

$$Y = \left\{ (x,k,y) \;\middle|\; \begin{array}{l} |y| = \log s \cdot h'(k) \cdot \log|x| \text{ und } y \text{ beschreibt die nicht-} \\ \text{deterministischen Schritte auf einem akzeptierenden} \\ \text{Pfad von } M' \text{ auf Eingabe } x. \end{array} \right\}$$

Setzen wir $h(k) = \log s \cdot h'(k)$. Da die Laufzeit von $M'$ durch $f'(k) \cdot p'(|x|)$ beschränkt ist, kann $Y$ ebenfalls in Laufzeit $f'(k) \cdot p'(|x|)$ entschieden werden, also $Y \in \mathsf{FPT}$.

„$\impliedby$“: Wir raten $y$, wofür wir genau $h(k) \cdot \log|x|$ nichtdeterministische Schritte benötigen und können dann in *fpt*-Laufzeit prüfen, ob $(x,k,y) \in V$. Daher gilt, dass

$$L = \{\, (x,k) \in \Sigma^* \times \mathbb{N} \mid \exists y \in \{0,1\}^* \colon (x,k,y) \in V \,\} \in \mathsf{W[P]}. \qquad \square$$

Wir haben $\mathsf{W[P]}$ als Vereinigung einer ganzen Hierarchie von Komplexitätsklassen angekündigt. Um diese Klassen zu beschreiben, definieren wir das gewichtete Erfüllbarkeitsproblem für aussagenlogische Formeln eingeschränkt auf eine Formelklasse $\Phi$. Mit dem *Gewicht* einer Belegung $\mathfrak{I}$ bezeichnen wir dabei die Anzahl der Variablen, die von $\mathfrak{I}$ auf Wahr gesetzt werden.

| $\mathsf{WSAT}(\Phi)$ | |
|---|---|
| **Eingabe:** | Eine Formel $\phi \in \Phi$, $k \in \mathbb{N}$ |
| **Parameter:** | $k$ |
| **Frage:** | Hat $\phi$ eine erfüllende Belegung mit Gewicht $k$? |

**Definition 2.1.20** ([FG06, Theorem 5.6])**.** $\mathsf{W}[\mathsf{t}] = [\mathsf{WSAT}(\Gamma_{\mathsf{t},\mathsf{d}})]^{fpt}$ für $\mathsf{t}+\mathsf{d} > 2$.

Üblicherweise werden die Klassen der W-Hierarchie über das gewichtete Erfüllbarkeitsproblem prädikatenlogischer Formeln (*weighted fagin definability*) definiert und dann die W[t]-Vollständigkeit der entsprechenden $\mathsf{WSAT}(\Gamma_{\mathsf{t},\mathsf{d}})$ nachgewiesen [FG06, Kapitel 5, 7].

Ebenso erhalten wir für die Klasse W[P] eine alternative Betrachtungsweise, indem wir das gewichtete Erfüllbarkeitsproblem für Boole'sche Schaltkreise betrachten.

| $\mathsf{WSAT}(\mathsf{CIRC})$ | |
|---|---|
| **Eingabe:** | Ein Boole'scher Schaltkreis $C$, $k \in \mathbb{N}$ |
| **Parameter:** | $k$ |
| **Frage:** | Hat $C$ eine erfüllende Belegung mit Gewicht $k$? |

**Lemma 2.1.21** ([FG06, Theorem 3.9])**.** $\mathsf{W}[\mathsf{P}] = [\mathsf{WSAT}(\mathsf{CIRC})]^{fpt}$.

Wir verzichten jedoch in diesem Kontext auf einen Beweis und dazu auf Abschnitt 3.7 im Buch von Flum und Grohe [FG06]. Die Idee basiert im Wesentlichen darauf, für einen Schaltkreis $\mathcal{C}$ eine erfüllende Belegung mit Gewicht $k$ zu raten, womit die Mitgliedschaft in W[P] nachgewiesen wäre.

Die W[P]-Schwere weist man nach, indem man für ein beliebiges Problem $L \in \mathsf{W}[\mathsf{P}]$ eine entsprechende DTM M betrachtet, die $(x,k) \overset{?}{\in} L$ zusammen mit einem $\kappa$-beschränkten Zertifikat entscheidet, und aus dieser Maschine einen Schaltkreis konstruiert, der genau dann erfüllbar ist, wenn $(x,k) \in L$.

Weitere Beispiele für W[t]-schwere Probleme finden sich in Kapitel 3.

## 2.2 Promise-Probleme

Wir führen noch den Begriff des *Promise* ein, den wir im Folgenden bei der Betrachtung der inkrementellen Probleme benötigen.

**Definition 2.2.1** ([Gol06, Definition 1])**.** Ein *Promise-Problem* $\Pi$ über einem Alphabet $\Sigma$ ist einem Paar disjunkter Mengen $(\Pi_{ja}, \Pi_{nein})$ mit $\Pi_{ja}, \Pi_{nein} \subset \Sigma^*$.

Ein Algorithmus, der $\Pi$ entscheidet, muss entsprechend Instanzen aus $\Pi_{ja}$ von Instanzen aus $\Pi_{nein}$ unterscheiden und darf sich auf Instanzen aus $\Sigma^* \setminus \Pi_{ja} \cup \Pi_{nein}$ beliebig verhalten.

Wir können auf diese Art die Eingaben eines Algortihmus, der ein Problem entscheidet, einschränken, sodass möglicherweise aufwändige Randfälle nicht betrachtet werden müssen, wie z. B. die korrekte Formatierung einer Eingabe.

## 2.3 Inkrementelle Probleme

Üblicherweise erhalten Algorithmen eine Eingabe und berechnen dazu eine passende Ausgabe. Wird die Eingabe verändert, beginnt der entsprechende Algorithmus von vorn, die neue Ausgabe zu berechnen. Solche Algorithmen bezeichen wir auch als *Batch-* oder *start-over*-Algorithmen [Ram96]. So wie beim Sprung von klassischer zur parametrisierten Komplexitätstheorie die Schwierigkeit von Problemen zusätzlich in Abhängigkeit des Parameters betrachtet wird, nehmen wir nun in unsere Betrachtung auf, dass sich die Eingabe im Nachhinein ändern kann und wir Informationen aus der vorherigen Berechnung wiederverwenden können.

Um den Begriff der *Änderungen an Eingaben* formal greifbar zu machen, orientieren wir uns an Definitionen von Creignou et al. [Cre+19, Definition 10] für *general operations* an Instanzen.

**Definition 2.3.1.** Sei $L \subseteq \Sigma^*$ eine Sprache. Wir bezeichnen mit *erlaubten Änderungen* an Instanzen von $L$ eine abzählbar unendliche Menge $\Delta(L) = \{ \omega_n \colon \Sigma^* \to \Sigma^* \mid n \in \mathbb{N} \}$ von Abbildungen über Instanzen von $L$. Wir schreiben auch nur $\Delta$, falls $L$ aus dem Kontext klar ist.

Die genaue Zusammensetzung von $\Delta$ hängt also stark vom betrachteten Problem ab. Für häufig betrachtete Arten von Problemen klären wir zunächst etwas Notation.

Wir betrachten folgende *Klassen von Änderungen*. Dabei bezeichne $\mathcal{G}$ die Menge aller Graphen und sei o. B. d. A. die Knotenmenge dieser Graphen von der Form $V = \{ v_i \mid 1 \leq i \leq n \}$. Außerdem bezeichne $\mathsf{Form}_{\mathsf{AL}}$ die Menge aller aussagenlogischen Formeln, die o. B. d. A. aus Variablen $\mathsf{Var}(\phi) = \{ x_i \mid 1 \leq i \leq n \}$ bestehen.

- Abbildungen $v_i^+ \colon \mathcal{G} \to \mathcal{G}$ mit $(V, E) \mapsto (V \cup \{v_i\}, E)$ für das *Hinzufügen von Knoten*.
- Abbildungen $v_i^- \colon \mathcal{G} \to \mathcal{G}$ mit $(V, E) \mapsto (V', E \cap (V' \times V'))$ und $V' = V \setminus \{v_i\}$ für das *Löschen von Knoten*.
- Abbildungen $e_{ij}^+ \colon \mathcal{G} \to \mathcal{G}$ mit $(V, E) \mapsto (V', E \cup \{v_i, v_j\})$ für das *Hinzufügen von Kanten*.
- Abbildungen $e_{ij}^- \colon \mathcal{G} \to \mathcal{G}$ mit $(V, E) \mapsto (V', E \setminus \{v_i, v_j\})$ für das *Löschen von Kanten*.
- Abbildungen $var_i^- \colon \mathsf{Form}_{\mathsf{AL}} \to \mathsf{Form}_{\mathsf{AL}}$ mit $\phi \mapsto \phi'$, wobei $\phi'$ aus $\phi$ hervorgeht, indem alle Vorkommen der Variable $x_i$ in $\phi$ gelöscht werden.
- Abbildungen $cl_i^- \colon \Phi_{t,d} \to \Phi_{t,d}$ mit $\Phi \in \{ \Gamma, \Delta \}$ und $\phi \mapsto \phi'$, wobei $\phi'$ aus $\phi$ hervorgeht, indem alle Vorkommen der Klausel $C_i$ in $\phi$ gelöscht werden. (Insbesondere ist $\Phi$ eine Formel in KNF oder einer Verallgemeinerung von KNF.)

Wir schreiben abkürzend für eine Klasse von Änderungen $\{\omega_i \mid i \in \mathbb{N}\}$ auch nur $\omega$ und $\omega^{\pm}$ für $\omega^+ \cup \omega^-$.

**Definition 2.3.2.** Für eine *Menge von Änderungen* $\delta = \{\omega_i \mid i \in I \subsetneq \mathbb{N}\} \subsetneq \Delta(L)$ an einer Instanz $x$ der Sprache $L$ bezeichnet $x \circ \delta = x \circ \omega_{i_1} \circ \cdots \circ \omega_{i_k} = \omega_{i_k}(\cdots(\omega_{i_1}(x))\cdots)$ die Instanz, die wir durch Anwendung der Änderungen aus $\delta$ erhalten.

Wir betonen hier noch einmal, dass $\delta$ nicht *alle* möglichen Änderung umfassen soll, weil die Verarbeitung unendlich vieler Änderungen nicht in endlicher Zeit möglich ist.

*Beispiel* ([Cre+19, Beispiel 1]). Sei $\mathcal{G} \subseteq \Sigma^*$ die Menge aller ungerichteten Graphen. Erlauben wir das Löschen und Hinzufügen von Kanten und Knoten, dann schreiben wir $\Delta(\mathcal{G}) = (v^{\pm}, e^{\pm})$. Insgesamt besteht $\Delta$ aus den Abbildungen

$$v_u^+ \colon \mathcal{G} \to \mathcal{G}, \quad v_u^- \colon \mathcal{G} \to \mathcal{G},$$
$$e_{\{u,w\}}^+ \colon \mathcal{G} \to \mathcal{G}, \quad e_{\{u,w\}}^- \colon \mathcal{G} \to \mathcal{G}.$$

Betrachten wir nun $G = (\{a,b,c\}, \{\{a,b\},\{b,c\}\})$ und $\delta = \{v_b^-, e_{\{a,c\}}^+\}$, dann ist

$$G \circ \delta = (\{a,c\}, \{\{a,c\}\}).$$

Bei der Definition der inkrementellen Probleme orientieren wir uns an der Arbeit von Mans und Mathieson [MM17], passen sie jedoch an, um unsere Definitionen von parametrisierten Problemen und Änderungen an Instanzen zu berücksichtigen.

**Definition 2.3.3** (Inkrementelle Probleme)**.**
Sei $L \subseteq \Sigma^* \times \mathbb{N}$ ein parametrisiertes Problem. Dann ist das dazugehörige *inkrementelle Problem* $\mathsf{incr}\text{-}L(\Delta) = (L', \Delta)$ gegeben durch die parametrisierte Sprache

$$L' = \{(x,k,\delta,S) \mid (x \circ \delta, k) \in L\} \subseteq \Sigma^* \times \mathbb{N} \times \mathcal{P}(\Delta) \times \{0,1\}^*$$

und eine Menge $\Delta$ an erlaubten Änderungen.

$S$ bezeichnet dabei einen *Zeugen* für $(x,k) \in L$. Wir können insbesondere o. B. d. A. annehmen, dass $S \in \{0,1\}^*$ vermöge einer Kodierung über einem Binäralphabet.

Parametrisierte Probleme, wie wir sie zuvor betrachtet haben, nennen wir im Gegensatz dazu *statische Probleme* nennen. Falls die genaue Zusammensetzung von $\Delta$ nicht relevant ist, schreiben wir auch nur $\mathsf{incr}\text{-}L$.

Insbesondere können wir ein inkrementelles Problem als Promise-Problem auffassen. Von den Instanzen $(x,k,\delta,S)$ fordern wir im Allgemeinen, dass bereits $(x,k) \in L$ und für den Zeugen $S$ können wir $L$ ähnlich zu Lemma 2.1.19 auch wie folgt betrachten.

$$L = \{(x,y) \in \Sigma^* \times \mathbb{N} \mid \exists S \in \{0,1\}^* \colon (x,k,S) \in V_L\}$$

Die Bedingung an $S$ ist gerade, dass $(x,y,S) \in V_L \iff (x,y) \in L$.

Man sieht, dass unter günstigen Bedingungen wie in Lemma 2.1.19 die Eigenschaft des Zeugen nachgeprüft werden kann, dies aber im Allgemeinen nicht der Fall ist. Es ist daher durchaus sinnvoll zu fordern, dass der Zeuge korrekt ist.

Die genaue Gestalt des Zeugen $S$ schränken wir hierbei jedoch nicht weiter ein, da er die Möglichkeit bietet, Informationen aus der Berechnung von $(x,k) \overset{?}{\in} L$ für die Frage nach $(x \circ \delta, k) \overset{?}{\in} L$ wiederzuverwenden. Der genaue Aufbau von $S$ ist damit sogar abhängig vom verwendeten Algorithmus.

Eine mögliche Intuition geben diesbezüglich Mitlersen et al. [Mil+94]. Demnach können wir davon ausgehen, dass $S$ als eine interne Datenstruktur durch den entsprechenden Algorithmus für $(x,k) \overset{?}{\in} L$ angelegt wird. $S$ beschreibt dann den internen Zustand des Algorithmus, der auch für eine geänderte Instanz $(x \circ \delta, k)$ relevant bleibt.

Ebenso ist $S$ nach unserer Definition nicht längenbeschränkt, dennoch ergibt sich eine „praktische“ Beschränkung, da $S$ zumindest bei der initialen Berechnung von $x \overset{?}{\in} L$ durch einen Batch-Algorithmus konstruiert wird, und dementsprechend durch die Laufzeit des Batch-Algorithmus in Abhängigkeit von $|x|$ beschränkt ist. Da dies jedoch von der Wahl des verwendeten Algorithmus abhängt, haben wir damit keine belastbare obere Schranke.

Der inkrementelle Gedanke besteht dann darin, auf Änderungen $\delta$ an der ursprünglichen Eingabe $x$ zu reagieren und bei der Berechnung der neuen Ausgabe Informationen aus der vorherigen Berechnung in Form des Zeugen $S$ wiederzuverwenden [Ram96].

Ein inkrementelles Problem lässt sich dann wie im Folgenden darstellen.

incr-$L(\Delta)$
**Eingabe:** $x \in \Sigma^*$, $k \in \mathbb{N}$, $\delta \subsetneq \Delta$, $S \in \{0,1\}^*$
**Parameter:** $k + |\delta|$
**Frage:** $(x \circ \delta, k) \in L$?

*Beispiel.* Die inkrementelle Variante des gewichteten Erfüllbarkeitsproblems definieren wir wie folgt.

incr-WSAT$(\Phi, \Delta)$
**Eingabe:** Formel $\phi \in \Phi$, $k \in \mathbb{N}$, $\delta \subsetneq \Delta^*$, $S \in \{0,1\}^*$ ein Zeuge für eine erfüllende Belegung von $\phi$ mit Gewicht $k$
**Parameter:** $k + |\delta|$
**Frage:** Hat $\phi \circ \delta$ eine erfüllende Belegung mit Gewicht $k + |\delta|$?

Erlauben wir, dass sich Instanzen durch das Löschen von Klauseln ändern, so schreiben wir incr-WSAT$(\Phi, cl^-)$. Erlauben wir dagegen das Löschen von Variabeln, so schreiben wir incr-WSAT$(\Phi, var^-)$.

Wir erhalten unmittelbar eine obere Schranke für die Komplexität der inkrementellen

Variante eines Problems incr-$L$, da wir einen Algorithmus für $L$ ebenso für incr-$L$ verwenden können und dabei die Informationen aus $S$ ignorieren.

**Lemma 2.3.4.** *Sei $L$ ein parametrisiertes Problem. Dann gilt* incr-$L \leq^{fpt} L$.

*Beweis.* Sei incr-$L = (L', \Delta)$ ein inkrementelles Problem zu einem parametrisierten Problem $L \subset \Sigma^* \times \mathbb{N}$. Dann gilt incr-$L \leq^{fpt} L$ via der Abbildung

$$\begin{aligned} \varphi\colon \Sigma^* \times \mathbb{N} \times \mathcal{P}(\Delta) \times \{0,1\}^* &\to \Sigma^* \times \mathbb{N} \\ (x, k, \delta, S) &\mapsto (x \circ \delta, k), \end{aligned}$$

die wie folgt berechnet wird.

**Eingabe:** $(x, k, \delta, S)$

1: $x' \leftarrow x$ ▷ Laufzeit: $|x|$
2: $k' \leftarrow k$ ▷ Laufzeit: $\log k$
3: **for** $\omega \in \delta$ **do** ▷ Laufzeit: $|\delta|$
4: $\quad x' \leftarrow x' \circ \omega$ ▷ Laufzeit: $|x|$
5: **return** $(x', k')$

$\varphi$ ist berechenbar in *fpt*-Laufzeit (parametrisiert nach $k + |\delta|$), denn

$$\begin{aligned} |x| + \log k + |\delta| \cdot |x| &= (|\delta| + 1) \cdot |x| + \log k \\ &\leq (|x|) \cdot (|\delta| + 1 + \log k) \\ &\leq \underbrace{|x|}_{p(|x|)} \cdot \underbrace{(k + |\delta| + 1)}_{f(k+|\delta|)}. \end{aligned}$$

Es gilt $(x, k, \delta, S) \in$ incr-$L \iff (x \circ \delta, k) \in L$ wegen der Definition von incr-L. Ebenso gilt $k \leq k + |\delta|$. Damit ist der Parameter der neuen Instanz beschränkt durch den Parameter der ursprünglichen Instanz. □

Im vorhergehenden Beweis haben wir dabei folgende technische Überlegung genutzt. Wir haben zwar *fpt*-Reduktionen als Abbildungen $\Sigma_1^* \times \mathbb{N} \to \Sigma_2^* \times \mathbb{N}$ definiert, können diesen Reduktionsbegriff aber auch für unsere inkrementellen Probleme nutzen, indem wir die Bestandteile der Instanzen entsprechend „umordnen“, also

$$(x, k, \delta, S) \mathrel{\widehat{=}} ((x, \delta, S), k).$$

# 3 Eine Auswahl parametrisierter Probleme und ihre Schwierigkeit

Zunächst betrachten wir einige klassische Resultate zu parametrisierten Problemen, um sie dann in Kapitel 4 mit dem Verhalten ihrer inkrementellen Varianten zu vergleichen, da sich einige Probleme bei diesem Übergang tatsächlich sehr unterschiedlich verhalten und unter Umständen einfacher werden [MM17].

## 3.1 (Max/Min)-WSAT

Wir beginnen zunächst mit zwei Varianten des gewichteten Erfüllbarkeitsproblems $\mathsf{WSAT}(\Phi)$. Zum einen werden dadurch einige der folgenden Reduktionen in der Argumentation etwas einfacher. Zum anderen sind Aussagen über das Verhalten der inkrementellen Varianten von MAX-WSAT und MIN-WSAT klarer ersichtlich.

Die Beweise der folgenden beiden Lemmata orientieren sich dabei an der Arbeit von Mans und Mathieson [MM17].

| $\mathsf{MAX\text{-}WSAT}(\Phi)$ | |
|---|---|
| **Eingabe:** | Eine Formel $\phi \in \Phi$, $k \in \mathbb{N}$ |
| **Parameter:** | $k$ |
| **Frage:** | Hat $\phi$ eine erfüllende Belegung mit Gewicht $\geq k$? |

| $\mathsf{MIN\text{-}WSAT}(\Phi)$ | |
|---|---|
| **Eingabe:** | Eine Formel $\phi \in \Phi$, $k \in \mathbb{N}$ |
| **Parameter:** | $k$ |
| **Frage:** | Hat $\phi$ eine erfüllende Belegung mit Gewicht $\leq k$? |

**Lemma 3.1.1** ([MM17, Lemma 1])**.** *Für ungerade* $\mathsf{t}$ *und* $\mathsf{t}+\mathsf{d} > 2$ *ist* $\mathsf{MAX\text{-}WSAT}(\Gamma^-_{\mathsf{t},\mathsf{d}})$ $\mathsf{W}[\mathsf{t}]$*-vollständig.*

*Beweis.* Wir zeigen zunächst via der Identitätsabbildung, dass

$$\mathsf{WSAT}(\Gamma^-_{\mathsf{t},\mathsf{d}}) \leq^{fpt} \mathsf{MAX\text{-}WSAT}(\Gamma^-_{\mathsf{t},\mathsf{d}}).$$

Falls $\phi$ eine erfüllende Belegung mit Gewicht $k$ hat, dann auch mit Gewicht $\geq k$. Damit gilt $(\phi, k) \in \mathsf{WSAT}(\Gamma^-_{\mathsf{t},\mathsf{d}}) \implies (\phi, k) \in \mathsf{MAX\text{-}WSAT}(\Gamma^-_{\mathsf{t},\mathsf{d}})$.

Sei nun $A = \{ x \in \mathsf{Var}(\phi) \mid \mathfrak{I}(x) = 1 \}$ die Menge der Variablen, die von einer erfüllenden Belegung $\mathfrak{I}$ mit Gewicht $\geq k$ auf Wahr gesetzt werden. Da $\mathsf{t}$ ungerade, ist $\phi$ von der Form

$$\phi = \bigwedge_{I_t} \Big( \bigvee_{I_{t-1}} \big( \dots \big( \bigwedge_{I_1} \big( \bigvee_{i=1}^{d} \neg \lambda_\alpha \big) \big) \dots \big) \Big)$$

für $\alpha \in I_t \times I_{t-1} \times \dots \times I_1 \times \{1, \dots, d\}$.

Da alle Literale negiert sind, genügt auch jede Teilmenge $A' \subseteq A$ für eine erfüllende Belegung. Falls also $\mathfrak{I}$ gerade $|A| = \ell > k$ Variablen auf Wahr setzt, dann können wir $\ell - k$ Variablen stattdessen auf Falsch setzten und haben eine erfüllende Belegung mit Gewicht $k$. Also gilt $(\phi, k) \in \mathsf{WSAT}(\Gamma^-_{\mathsf{t},\mathsf{d}}) \impliedby (\phi, k) \in \mathsf{MAX\text{-}WSAT}(\Gamma^-_{\mathsf{t},\mathsf{d}})$.

Für $\mathsf{MAX\text{-}WSAT}(\Gamma^-_{\mathsf{t},\mathsf{d}}) \leq^{fpt} \mathsf{WSAT}(\Gamma^-_{\mathsf{t},\mathsf{d}})$ können wir ebenfalls die Identitätsabbildung verwenden und uns vorherige Überlegung zunutze machen.

Wir haben bereits in Definition 2.1.7 gesehen, dass die Identitätsabbildung in *fpt*-Laufzeit berechenbar ist. Damit folgt insgesamt, dass $\mathsf{WSAT}(\Gamma^-_{\mathsf{t},\mathsf{d}}) \leq^{fpt} \mathsf{MAX\text{-}WSAT}(\Gamma^-_{\mathsf{t},\mathsf{d}})$. □

**Lemma 3.1.2** ([MM17, Lemma 2])**.** $\mathsf{MIN\text{-}WSAT}(\Gamma^+_{\mathsf{t},\mathsf{d}})$ *ist* $\mathsf{W}[\mathsf{t}]$*-vollständig für gerades* $\mathsf{t}$ *und* $\mathsf{d} \geq 1$.

*Beweis.* Wir merken an, dass $\phi \in \Gamma^+_{\mathsf{t},\mathsf{d}}$ von der Form

$$\phi = \bigwedge_{I_t} \Big( \bigvee_{I_{t-1}} \big( \dots \big( \bigvee_{I_1} \big( \bigwedge_{i=1}^{d} \lambda_\alpha \big) \big) \dots \big) \Big)$$

für $\alpha \in I_\mathsf{t} \times I_{\mathsf{t}-1} \times \dots \times I_1 \times \{1, \dots, \mathsf{d}\}$ ist. Da alle Literale positiv sind, gilt für eine Menge $A$ von Variablen, die von einer Belegung $\mathfrak{I}$ auf Wahr gesetzt werden, dass auch jede Menge $A' \supseteq A$ für eine erfüllende Belegung verwendet werden kann. Ansonsten folgt der Beweis der Aussage einer ähnlichen Argumentation wie in Lemma 3.1.1. □

## 3.2 Independent-Set

Das Problem, eine unabhängige Knotenmenge in einem Graphen zu finden, ist wie folgt definiert.

p-Independent-Set

| | |
|---|---|
| **Eingabe:** | Graph $G = (V, E)$, $k \in \mathbb{N}$ |
| **Parameter:** | $k$ |
| **Frage:** | Existiert $V' \subseteq V$ mit $\lvert V' \rvert \geq k$, sodass für je zwei $u, v \in V'$ gilt, dass $\{u, v\} \notin E$? |

Folgendes Problem stellt eine Verallgemeinerung von p-Independent-Set auf Hypergraphen dar. Dabei bezeichnet ein Hypergraph $G$ ein Tupel $(V, E)$, sodass $E \subseteq \mathcal{P}(V)$.

Hypergraph d-Independent-Set
**Eingabe:** Hypergraph $G = (V, E)$, $k \in \mathbb{N}$
**Parameter:** $k$
**Frage:** Existiert eine Menge $V' \subseteq V$ mit $|V'| \geq k$, so dass für alle $e \in E$ gilt, dass $|e \cap V'| < d$?

Eine solche Menge $V'$ nennen wir auch *d-unabhängig*. Für $d = 2$ und falls $G$ ein Graph ist, entspricht dieses Problem dem gewöhnlichen p-Independent-Set. Der folgende Beweis orientiert sich am entsprechenden Resultat aus der Arbeit von Mans und Mathieson [MM17].

**Lemma 3.2.1** ([MM17, Lemma 3])**.** Hypergraph d-Independent-Set *ist* W[1]*-schwer für alle* $\mathsf{d} \geq 2$.

*Beweis.* Wir reduzieren von MAX-WSAT($\Gamma^-_{1,\mathsf{d}}$) mittels der folgenden Abbildung

$$\begin{aligned} \varphi \colon \Sigma^* \times \mathbb{N} &\to \Gamma^* \times \mathbb{N} \\ (\phi, k) &\mapsto (G, k) \end{aligned}$$

für $V(G) = \mathsf{Var}(\phi)$ und $E(G) = \mathrm{cl}(\phi)$. Insbesondere ist $\phi = \bigwedge_i (\bigvee_{j=1}^{d} \neg \ell_{ij})$ wegen $\phi \in \Gamma^-_{1,\mathsf{d}}$. Daher ist $\mathrm{cl}(\phi) = \{ C_i \mid i \in I \}$ und $C_i = \{ \ell_{ij} \mid 1 \leq j \leq d \}$.

Falls $\phi$ eine erfüllende Belegung $\mathfrak{I}$ mit Gewicht $\geq k$ hat, dann muss $\mathfrak{I}$ mindestens eine eine Variable pro Klausel auf Falsch setzen, da $\phi$ in d-KNF ist und nur negierte Literale enthält. Damit sind aber höchstens $\mathsf{d} - 1$ Variablen je Klausel auf Wahr gesetzt.

Sei nun $V' = \{ x \in \mathsf{Var}(\phi) \mid \mathfrak{I}(x) = 1 \}$. Dann ist $V'$ eine unabhängige Knotenmenge für $G$, denn aufgrund der vorherigen Überlegung wissen wir, dass $|e \cap V'| < \mathsf{d}$ für alle $e \in \mathrm{cl}(\phi)$. Damit gilt

$$(\phi, k) \in \text{MAX-WSAT}(\Gamma^-_{1,\mathsf{d}}) \implies (G, k) \in \text{Hypergraph d-Independent-Set}.$$

Umgekehrt gilt, wenn $V' \subseteq \mathsf{Var}(\phi)$ eine d-unabhängige Knotenteilmenge ist, darf jede Kante $e \in \mathrm{cl}(\phi)$ höchstens $\mathsf{d} - 1$ Variablen enthalten. Setzen wir genau die Variablen aus $V'$ auf Wahr, erhalten wir eine erfüllende Belegung für $\phi$ mit Gewicht $\geq k$. Daher gilt

$$(\phi, k) \in \text{MAX-WSAT}(\Gamma^-_{1,\mathsf{d}}) \iff (G, k) \in \text{Hypergraph d-Independent-Set}. \qquad \square$$

Als Korollar erhalten wir:

**Korollar 3.2.2.** p-Independent-Set *ist* W[1]*-schwer.*

*Beweis.* p-Independent-Set ist Hypergraph 2-Independent-Set und daher nach Lemma 3.2.1 W[1]-schwer. □

## 3.3 Clique

Ein *vollständiger Graph* $K_n$ ist ein Graph, in dem alle $n$ Knoten paarweise adjazent sind. Entsprechend können wir das Problem, ob ein Graph einen vollständigen Teilgraphen enthält, wie folgt formulieren.

p-Clique
**Eingabe:** Graph $G = (V, E)$, $k \in \mathbb{N}$
**Parameter:** $k$
**Frage:** Existiert $V' \subseteq V$ mit $|V'| \geq k$, sodass für je zwei $u, v \in V'$ gilt, dass $\{u, v\} \in E$?

Auch hier orientieren wir uns an der Arbeit von Mans und Mathieson und betrachten folgende Verallgemeinerung auf Hypergraphen.

Hypergraph d-Clique
**Eingabe:** Hypergraph $G = (V, E)$, $k \in \mathbb{N}$
**Parameter:** $k$
**Frage:** Gibt es $V' \subseteq V$ mit $|V'| \geq k$, sodass für alle $e \subseteq V'$ mit $|e| \leq d$ gilt, dass $e \in E$?

Ein solches $V'$ bezeichnen wir auch als *$d$-Clique* und für $d = 2$ auch einfach *Clique*.

**Lemma 3.3.1** ([MM17, Lemma 4])**.** Hypergraph d-Clique *ist* W[1]*-schwer für alle* $d \geq 2$.

*Beweis.* Wir reduzieren wieder von MAX-WSAT$(\Gamma^-_{1,\mathsf{d}})$ via folgender Abbildung

$$\varphi \colon \Sigma^* \times \mathbb{N} \to \Gamma^* \times \mathbb{N}$$
$$(\phi, k) \mapsto (G, k)$$

wobei $V(G) = \mathsf{Var}(\phi)$ und $E(G) = \left( \bigcup_{i=1}^{d} \mathsf{Var}(\phi)^i \right) \setminus \mathrm{cl}(\phi)$. Wir betrachten also einen Graphen, dessen Knoten den Variablen unserer Formel entsprechen und dessen Kanten allen möglichen Klauseln mit höchstens $\mathsf{d}$ Variablen entsprechen, die gerade nicht in $\phi$ vorkommen.

Falls $\phi$ eine erfüllende Belegung $\mathfrak{I}$ mit Gewicht $\geq k$ hat, dann enthält $\phi$ keine Klausel, die vollständig aus Variablen besteht, die von $\mathfrak{I}$ auf Wahr gesetzt werden. Damit sind aber alle solche Klauseln in $E(G)$ enthalten. Die Menge $V' = \{x \in \mathsf{Var}(\phi) \mid \mathfrak{I}(x) = 1\}$ ist

also eine $d$-Clique der Größe $|V'| \geq k$. Es gilt daher

$$(\phi, k) \in \text{MAX-WSAT}(\Gamma^-_{1,d}) \implies (G, k) \in \text{Hypergraph d-Clique}.$$

Falls umgekehrt $G$ eine d-Clique $V'$ der Größe $|V'| \geq k$ hat, dann können wir die Variablen aus $V'$ auf Wahr setzen und erhalten damit eine Belegung $\mathfrak{I}$ mit Gewicht $\leq k$. Vor allem ist $\mathfrak{I}$ eine erfüllende Belegung, da für alle Klauseln $C \not\subseteq V'$ und damit in jeder Klausel Variablen bleiben, die von $\mathfrak{I}$ auf Falsch gesetzt werden. Jede Klausel wird also erfüllt. Und damit gilt insgesamt

$$(\phi, k) \in \text{MAX-WSAT}(\Gamma^-_{1,\mathsf{d}}) \iff (G, k) \in \text{Hypergraph d-Clique}. \qquad \square$$

**Korollar 3.3.2.** p-Clique *ist* W[1]*-schwer.*

*Beweis.* p-Clique ist Hypergraph 2-Clique und daher nach Lemma 3.3.1 auch W[1]-schwer. $\square$

Insbesondere sieht man für $\mathsf{d} = 2$ die Beziehung zwischen p-Independent-Set und p-Clique, wie sie auch in den Beweisen zu Lemma 3.3.1 und Lemma 3.2.1 genutzt wurde, und zwar, dass eine Clique eine unabhängige Knotenmenge im komplementären Graph ist. Diesen Sachverhalt und insbesondere die dazugehörige Reduktion halten wir noch einmal gesondert fest, um uns später darauf zu beziehen.

**Lemma 3.3.3.** p-Independent-Set $\equiv^{fpt}$ p-Clique.

*Beweis.* Wir definieren für einen Graphen $G = (V, E)$ den *komplementären Graph* als $\overline{G} = (V, V \times V \setminus E)$. Damit können wir folgende Abbildung als *fpt*-Reduktion angeben.

$$\begin{aligned} \varphi \colon \Sigma^* \times \mathbb{N} &\to \Sigma^* \times \mathbb{N} \\ (G, k) &\mapsto (\overline{G}, k) \end{aligned}$$

Falls $(G, k) \in$ p-Independent-Set, dann sind keine $v, w \in S$ über eine Kante in $G$ verbunden. Dann befinden sich jedoch alle Kanten, die je zwei Knoten $v, w \in S$ verbinden, in $\overline{G}$, womit $S$ eine Clique für $\overline{G}$ ist und $(\overline{G}, k) \in$ p-Clique.

Da die verwendeten Argumente symmetrisch sind, erhalten wir zum einen, dass p-Independent-Set $\leq^{fpt}$ p-Clique, aber auch insgesamt p-Clique $\equiv^{fpt}$ p-Independent-Set, indem wir $\varphi$ auf Instanzen für p-Clique anwenden und dieselbe Argumentation verwenden. $\square$

Es sei außerdem angemerkt, dass p-Independent-Set und p-Clique sogar W[1]-vollständig sind, da wir die Mitgliedschaft von p-Clique in W[1] über die weighted fagin definability erhalten [FG06, Beispiel 4.39]. Die W[1]-Schwere von p-Clique erhält man über eine mehrschrittige Reduktion von p-MC($\Sigma_1$), einer Variante des Problems der Modellprüfung für Formeln mit einem Existenzquantor [FG06, Lemma 6.11, 6.13, 6.14].

## 3.4 Dominating-Set

In einem Graphen $G = (V, E)$ nennen wir eine Teilmenge von Knoten $V' \subseteq V$ *dominierend*, falls jeder Knoten $v \in V$ entweder bereits in $V'$ enthalten oder adjazent zu einem Knoten $u \in V'$ ist. Das parametrisierte Problem, in einem Graphen eine solche dominierende Menge zu finden, bezeichnen wir mit p-Dominating-Set und definieren es wie folgt.

p-Dominating-Set
**Eingabe:** Graph $G = (V, E)$, $k \in \mathbb{N}$
**Parameter:** $k$
**Frage:** Existiert eine dominierende Menge $V' \subseteq V$ mit $|V'| \leq k$?

Für den Beweis des folgenden Resultates haben wir die Reduktionen aus Theorem 7.14. und Beispiel 2.7. im Buch von Flum und Grohe [FG06] genutzt und über die Transitivität von *fpt*-Reduktionen zusammengefasst, um den Zwischenschritt über das Problem Hitting-Set zu überspringen, das wir nicht betrachten.

**Lemma 3.4.1.** p-Dominating-Set $\leq^{fpt}$ MIN-WSAT$(\Gamma^+_{2,1})$

*Beweis.* Sei also zunächst $(G, k)$ eine Instanz von p-Dominating-Set. Wir konstruieren eine KNF-Formel $\phi$, indem wir die Knoten von $G$ als Variablen verwenden. Dann ist $\phi$ wie folgt aufgebaut, wobei $N[v]$ die geschlossene Nachbarschaft des Knotens $v$ bezeichne.

$$\phi = \bigwedge_{v \in V} \bigvee_{u \in N[v]} u$$

Wir haben also für jeden Knoten $v$ eine Klausel, die Variablen für alle adjazenten Knoten und $v$ selbst enthält.

Wir behaupten, dass folgende Abbildung $\varphi$ eine *fpt*-Reduktion ist.

$$\varphi \colon \Sigma^* \times \mathbb{N} \to \Gamma^* \times \mathbb{N}$$
$$(G, k) \mapsto (\phi, k)$$

Die Abbildung $\varphi$ ist in Laufzeit $\mathcal{O}(|V(G)|^2)$ berechenbar, da wir für jede Nachbarschaft eines Knotens $v$ die entsprechende Klausel ausgeben müssen und dies nacheinander machen können.

Falls nun $G$ eine dominierende Knotenmenge $V'$ mit höchstens $k$ Knoten hat, dann gibt es für jeden Knoten $v$, der nicht bereits in $V'$ enthalten ist, einen adjazenten Knoten $u \in V'$. Damit ist aber in jeder Menge $N[v]$ ein solcher Knoten $u$ enthalten.

Setzen wir also jede Variable in $\phi$, die mit einem Knoten aus $V'$ korrespondiert, auf Wahr, dann erhalten wir eine erfüllende Belegung mit Gewicht höchstens $k$ für $\phi$, da wir mindestens eine Variable pro Klausel und insgesamt höchstens $k$ Variablen auf Wahr setzen.

Umgekehrt gilt ebenfalls, falls $\phi$ eine erfüllende Belegung mit Gewicht von höchstens $k$ hat, dann bildet die Menge $V' = \{ v \in V \mid \mathfrak{I}(v) = 1 \}$ eine dominierende Knotenmenge von $G$ mit höchstens $k$ Knoten. □

**Lemma 3.4.2.** p-Dominating-Set *ist* W[2]*-vollständig.*

*Beweis.* Es fehlt noch MIN-WSAT$(\Gamma^+_{2,1}) \leq^{fpt}$ p-Dominating-Set. Sei $\phi$ eine aussagenlogische Formel in KNF, die nur positive Literale enthält, also $\phi = \bigwedge_{i \in I} C_i$ und $C_i = \bigvee_{j \in J_i} \ell_{ij}$. Wir gehen außerdem o. B. d. A. davon aus, dass $\phi$ keine leeren Klauseln enthält. Ansonsten können wir solche Klauseln weglassen.

Wir konstruieren für $\phi$ einen Graphen $G$ wie folgt. Wir setzen $V(G) = \mathsf{Var}(\phi) \cup cl(\phi)$ und $E = E_1 \cup E_2$, wobei

$$E_1 = \{ \{x_k, C_i\} \mid C_i \in cl(\phi), x_k \in C_i \} \text{ und}$$
$$E_2 = \{ \{x_i, x_j\} \mid x_i, x_j \in \mathsf{Var}(\phi), i \neq j \}.$$

Wir verbinden im resultierenden Graphen also den Knoten einer Variable $x$ und einer Klausel $C$, falls $x$ in $C$ enthalten ist. Und außerdem verbinden wir die Knoten aller Variablen untereinander, ohne dabei einen Knoten mit sich selbst zu verbinden.

Wir behaupten, dass die Abbildung $\varphi$ eine *fpt*-Reduktion ist.

$$\varphi \colon \Sigma^* \times \mathbb{N} \to \Gamma^* \times \mathbb{N}$$
$$(\phi, k) \mapsto (G, k)$$

Die Menge $V(G)$ kann in Zeit $\mathcal{O}(|\phi|)$ berechnet werden, da $\phi$ nur einmal gelesen und dabei jede Variable und jede Klausel als Knoten ausgegeben werden muss. Die Berechnung von $V(G)$ kann mit der Berechnung von $E_1$ verbunden werden und da $|\mathsf{Var}(\phi)| \leq |\phi|$, kann $E_2$ ebenfalls in Zeit $\mathcal{O}(|\phi|^2)$ berechnet werden.

Es sei zunächst $\mathfrak{I}$ eine erfüllende Belegung für $\phi$ mit Gewicht höchstens $k$ und $A \subseteq \mathsf{Var}(\phi)$ die Menge der Variablen, die von $\mathfrak{I}$ auf Wahr gesetzt werden. Da $\phi$ nur positive Literale enthält, muss jede Klausel mindestens eine Variable aus $A$ enthalten. Dementsprechend ist in $G$ jeder Klauselknoten zu einem Knoten einer Variable aus $A$ adjazent.

Und weil alle Variablen untereinander verbunden sind, ist auch jeder Variablenknoten entweder bereits in $A$ oder zu einem Knoten in $A$ adjazent. Damit ist $A$ also auch eine dominierende Knotenmenge von $G$ mit höchstens $k$ Knoten.

Sei nun umgekehrt $(G, k) \in$ p-Dominating-Set und $A$ eine entsprechende dominierende Knotenmenge in $G$. Falls $A \subseteq \mathsf{Var}(\phi)$, dann können wir die Variablen aus $A$ auf Wahr setzen und erhalten somit eine erfüllende Belegung $\mathfrak{I}$ für $\phi$ mit Gewicht höchstens $k$, da jeder Knoten in $G$ zu mindestens einem Knoten in $A$ adjazent ist. Damit ist aber auch jeder Knoten einer Klausel mit einem Knoten in $A$ verbunden. Jede Klausel enthält also eine Variable, die durch $\mathfrak{I}$ auf Wahr gesetzt wird.

Ansonsten kann $A$ auch Knoten enthalten, die mit Klauseln aus $\phi$ korrespondieren, also $A \cap cl(\phi) \neq \emptyset$. Wir behaupten, dass für eine solche Menge $A$ eine weitere dominierende Knotenmenge $A' \subset \mathsf{Var}(\phi)$ existiert.

Wir stellen zunächst fest, dass eine Klausel $C$ in $G$ nur Kanten zu Variablen $v$ hat, es sind also keine zwei Klauseln untereinander verbunden. Außerdem sind alle Variablen in $G$ untereinander verbunden.

Daher erhalten wir $A'$, indem wir jeden Knoten für eine Klausel $C$ in $A$ durch einen Knoten ersetzen, der mit einer Variable $x \in C$ korrespondiert. Falls jedoch $C \subseteq A$, dann können wir stattdessen irgendeinen anderen Knoten hinzufügen. Auf $A'$ können wir die vorherige Arugmentation anwenden.

Insgesamt erhalten wir, dass genau dann $(G, k) \in$ p-Dominating-Set, wenn $(\phi, k) \in \mathsf{MIN\text{-}WSAT}(\Gamma^+_{2,1})$, und daher $\mathsf{MIN\text{-}WSAT}(\Gamma^+_{2,1}) \leq^{fpt}$ p-Dominating-Set. Zusammen mit Lemma 3.4.1 ist p-Dominating-Set also W[2]-vollständig ist. □

## 3.5 Vertex-Cover

Wir erinnern uns zunächst an die Definition von p-Vertex-Cover aus Abschnitt 2.1.

p-Vertex-Cover

| | |
|---|---|
| **Eingabe:** | Graph $G = (V, E)$, $k \in \mathbb{N}$ |
| **Parameter:** | $k$ |
| **Frage:** | Existiert ein $U \subseteq V$ der Größe $\leq k$, sodass für alle $\{v, u\} \in E$ gilt, dass $v \in U$ oder $u \in U$? |

Die parametrisierte Variante von Vertex-Cover ist in der parametrisierten Komplexitätstheorie sehr ausführlich erforscht worden, wobei die Laufzeit entsprechender Algorithmen stetig verbessert wurde. So beträgt die beste bekannte asymptotische Laufzeit $\mathcal{O}(1.2783^k + kn)$ nach Chen, Kanj und Xia [CKX06].

Als prominentes Beispiel für einen *fpt*-Algorithmus betrachten wir *Buss' Kernelization* genauer. Der wesentliche Gedanke hinter diesem Algorithmus ist die Eigenschaft von Problemen in FPT, eine sogenannte *Kernelization* zu besitzen [DF13, Proposition 4.7.1]. Damit ist gemeint, für ein prametrisiertes Problem eine gegebene Instanz auf eine kleinere Instanz desselben Problems zurückführen zu können.

Zunächst sprechen wir folgende Beobachtungen an, die uns eine solche Kernelization für p-Vertex-Cover ermöglichen, entnommen aus dem Buch von Downey und Fellows [DF99, Theorem 3.10].

**Lemma 3.5.1.** *Sei $G = (V, E)$ ein Graph und $v_i \in V$ mit $\deg(v_i) > k$. Es ist genau dann $(G, k) \in$ p-Vertex-Cover, wenn $(G \circ v_i^-, k-1) \in$ p-Vertex-Cover.*

*Beweis.* Falls $G$ eine Knotenüberdeckung $U \subseteq V$ der Größe $k$ besitzt, dann muss $v$ in $U$ enthalten sein. Wäre $v \notin U$, dann müssten alle Nachbarn von $v$ in $U$ enthalten sein, um

die inzidenten Kanten abzudecken. Wegen $\deg(v) > k$ wäre dann aber $|U| > k$, was jedoch im Widerspruch zu $|V'| = k$ steht.

Insbesondere ist $U \setminus \{v\}$ eine Knotenüberdeckung für $G \circ v_i^-$. Wir können also $v$ aus $G$ entfernen und in $G \circ v_i^-$ nach einer Knotenüberdeckung der Größe $k-1$ suchen. Falls eine solche Knotenüberdeckung existiert, dann können wir $v$ hinzufügen und erhalten eine Überdeckung der Größe $k$ für $G$. □

**Lemma 3.5.2.** *Sei $G = (V, E)$ ein Graph, $v_i \in V$ mit $\deg(v_i) = 0$. $(G, k)$ ist genau dann in* p-Vertex-Cover*, wenn $(G \circ v_i^-, k) \in$* p-Vertex-Cover*.*

*Beweis.* Der Knoten $v_i$ trägt nicht zur Knotenüberdeckung bei. Wir können ihn daher problemlos aus der Betrachtung entfernen. □

*Bemerkung* 3.5.3. Man kann diese Beobachtungen auch als *fpt*-Reduktionen zwischen Instanzen von p-Vertex-Cover auffassen, bei der die Größe und der Parameter der ursprünglichen Instanz verringert wird [DF13, Definition 4.7.1].

**Lemma 3.5.4.** *Sei $G = (V, E)$ ein Graph mit $\deg(v) \leq k$ für alle $v \in V$. $(G, k)$ ist genau dann in* p-Vertex-Cover*, wenn $|V| \leq k(k+1)$.*

*Beweis.* Wir gehen o. B. d. A. davon aus, dass $G$ keine Knoten mit $\deg(v) = 0$ hat, da Knoten ohne inzidente Kanten für eine Knotenüberdeckung nicht relevant sind und wir sie problemlos löschen können.

Falls also $G$ eine Überdeckung der Größe $k$ besitzt, können wir wegen $\deg(v) \leq k$ mit einem Knoten höchstens $k$ Kanten abdecken. Mit $k$ Knoten können wir entsprechend $k^2$ Kanten abdecken. Daher kann es also höchstens $k^2 + k = k(k+1)$ Knoten geben. □

Bevor wir uns mit der tatsächlichen Kernelization nach Buss befassen, betrachten wir zunächst das Verfahren der *bounded search trees*. Die Idee dabei ist, alle möglichen Knotenteilmengen eines Graphen nach einer Knotenüberdeckung zu durchsuchen. Da wir uns nur für Lösungen der Größe $\leq k$ interessieren, ist die Tiefe dieses Baumes durch $k$ beschränkt.

Ein Algorithmus nach diesem Prinzip erreicht nicht ganz die Effizienz, die wir uns wünschen. Wir können ihn jedoch als Hilfsmittel für unser abschließendes Resultat benutzen. Folgendes Resultat haben wir aus dem Buch von Downey und Fellows [DF13] und den dazugehörigen Algorithmus mit Laufzeitbetrachtung aus dem Buch von Flum und Grohe [FG06, Theorem 1.17, Algorithmus 1.4] entnommen.

**Lemma 3.5.5** ([DF13, Theorem 3.2.1])**.** p-Vertex-Cover *ist in Zeit $\mathcal{O}(2^k \cdot |V(G)|)$ lösbar.*

*Beweis.* Folgender Algorithmus berechnet eine Knotenüberdeckung der Größe $k$.

**Prozedur** SEARCH-VC$(G = (V, E), k \in \mathbb{N}, X \subseteq V)$

```
1: if E = ∅ then return X
2: else if k = 0 then return ∅
3: else
4:     Wähle e ∈ E
5:     U ← ∅
6:     for v_i ∈ e do
7:         U ← SEARCH-VC(G ∘ v_i^-, k − 1, X ∪ {v})
8:         if U ≠ ∅ then return U
9:     return ∅
```

Wir konstruieren einen Binärbaum der Tiefe $k$, indem wir die Wurzel mit $(\emptyset, G)$ markieren.

Die restlichen Knoten des Baumes markieren wir mit $(U, H)$, wobei $U$ die Knoten aus $G$ enthält, die in einer potentiellen Knotenüberdeckung enthalten sind, und $H$ den Teilgraphen von $G$ darstellt, dessen Kanten wir noch abdecken müssen.

Die Nachfolger eines Knoten erhalten wir, indem wir eine Kante $(v, u) \in E(H)$ auswählen, zum Beispiel über die lexikografisch kleinste Kante. In einer Überdeckung von $H$ muss entweder $v$ oder $u$ enthalten sein. Die beiden Nachfolger von $(U, H)$ sind also $(U \cup \{v_i\}, H \circ v_i^-)$ und $(U \cup \{v_j\}, H \circ v_j^-)$.

Insbesondere gilt, dass in Tiefe $\ell$ des Binärbaumes $|U| = \ell$. Falls wir also in Tiefe $\leq k$ einen Knoten $(U, H)$ mit $E(H) = \emptyset$ finden, dann ist $U$ eine Knotenüberdeckung von $G$.

Die Laufzeit des Algorithmus erhält man aus folgender induktiver Betrachtung. Wir bezeichnen mit $T(k, n)$ die maximale Laufzeit von SEARCH-VC$(G, k, X)$. Dann gilt für $k, c \in \mathbb{N}$ und $n = |V(G)|$ der folgende Zusammenhang.

$$T(0, n) = c \tag{3.1}$$

$$T(k, n) = 2 \cdot T(k - 1, n) + cn \tag{3.2}$$

Der Term $2 \cdot T(k - 1, n)$ bezieht sich dabei auf die rekursiven Aufrufe in Zeile 7 Wir behaupten, dass wir $T(k, n)$ für alle $k \geq 0$ wie folgt abschätzen können.

$$T(k, n) \leq (2^{k+1} - 1) \cdot cn$$

Für unseren Induktionsanfang $k = 0$ gilt $T(0, n) = c \leq cn = (2 - 1) \cdot cn$. Für $k > 0$

erhalten wir

$$\begin{aligned} T(k,n) &= 2 \cdot T(k-1,n) + cn \\ &\leq 2 \cdot (2^k - 1) \cdot cn + cn \\ &= (2 \cdot (2^k - 1) + 1) \cdot cn \\ &= (2^{k+1} - 1) \cdot cn \end{aligned}$$

Da $(2^{k+1} - 1) \cdot c \cdot n \leq 2c \cdot 2^k \cdot n$, benötigt SEARCH-VC$(G, k, X)$ eine Laufzeit von $\mathcal{O}(2^k \cdot n)$. □

**Satz 3.5.6** ([DF99, Korollar 3.11])**.** p-Vertex-Cover *ist lösbar in Zeit* $\mathcal{O}(|V|+|E|+2^k \cdot k^2)$.

*Beweis.* Sei $(G, k)$ eine Instanz für p-Vertex-Cover. Wir nutzen die vorherigen Überlegungen, um $(G, k)$ zunächst zu verkleinern. Der dazugehörige Algorithmus setzt die Beobachtungen aus den Lemmata 3.5.1 bis 3.5.4 wie im Folgenden um.

1. Wir suchen zunächst alle Knoten $v$ mit $\deg(v) > k$ und löschen sie, da solche Knoten für eine Knotenberdeckung der Größe $k$ benötigt werden. Die Menge dieser Knoten bezeichnen wir mit $V^{(>k)}$.

2. Wir reduzieren $k$ um die Anzahl der Knoten, die wir gerade gelöscht haben, also in jeder Knotenüberdeckung der Größe $\leq k$ enthalten sein müssen.

3. Wir löschen außerdem alle Knoten $v$ mit $\deg(v) = 0$, denn diese werden für eine Knotenüberdeckung nicht benötigt. Die Menge dieser Knoten bezeichnen wir mit $V^{(0)}$.

$G$ hat *keine* Knotenüberdeckung der Größe $k$, falls

*a)* $|V^{(>k)}| > k$, da wir dann zu viele Knoten für eine Überdeckung benötigen.

*b)* $|V \setminus (V^{(>k)} \cup V^{(0)})| > k'(k'+1)$, denn nach Lemma 3.5.4 hat $G'$ keine Knotenüberdeckung der Größe $k'$ und nach Lemmata 3.5.1 und 3.5.2 hat $G$ ebenfalls keine Knotenüberdeckung der Größe $k$.

Im Folgenden bezeichne $G^-$ dabei einen Graphen, der keine Knotenüberdeckung der Größe $k \leq 1$ besitzt. Folgender Graph erfüllt diese Bedingung.

$G^-$: 

**Prozedur** REDUCE($G = (V, E), k \in \mathbb{N}$)

// Schritt 1

1: $V^{(>k)} \leftarrow \{ v \in V \mid \deg(v) \geq k \}$ ▷ $\mathcal{O}(|E|)$

2: $V \leftarrow V \setminus V^{(>k)}$, $E \leftarrow E \setminus V^{(>k)} \times V^{(>k)}$ ▷ $\mathcal{O}(|V| + |E|)$

// Abbruchbedingung $a$)

3: **if** $|V^{(>k)}| > k$ **then return** $(G^-, 1)$

// Schritt 2

4: $k' \leftarrow k - |V^{(>k)}|$

// Schritt 3

5: $V^{(0)} \leftarrow \{ v \in V \mid \{v, u\} \in E, \deg(u) = 0 \}$ ▷ $\mathcal{O}(|E|)$

6: $V \leftarrow V \setminus V^{(0)}$ ▷ $\mathcal{O}(|V|)$

// Abbruchbedingung $b$)

7: **if** $|V| > k(k+1)$ **then return** $(G^-, 1)$

8: **return** $(G', k')$

Schließlich können wir in $G'$ nach einer Überdeckung der Größe $k'$ suchen und dann um die Knoten aus $V^{(>k)}$ ergänzen, um eine Knotenüberdeckung der Größe $k$ für $G$ zu erhalten. In diesem speziellen Fall nutzen wir dazu SEARCH-VC aus Lemma 3.5.5.

Die Berechnung von $(G', k')$ ist möglich in $\mathcal{O}(|V| + |E|)$ und da $G'$ höchstens $k(k+1)$ Knoten hat, benötigt SEARCH-VC eine Laufzeit von $\mathcal{O}(2^k \cdot (k^2 + k)) = \mathcal{O}(2^k \cdot k^2)$. □

Insbesondere können wir diesen Algorithmus nutzen, um in Kapitel 4 Aussagen über die Komplexität von incr-Vertex-Cover zu treffen.

Abschließend fassen wir in Abbildung 3.1 die Ergebnisse aus diesem Kapitel bezüglich der Komplexität der betrachteten Probleme zusammen. Wir beschränken die Übersicht dabei auf die bisherigen Probleme der W-Hierarchie, da diese im weiteren Verlauf von Interesse sind.

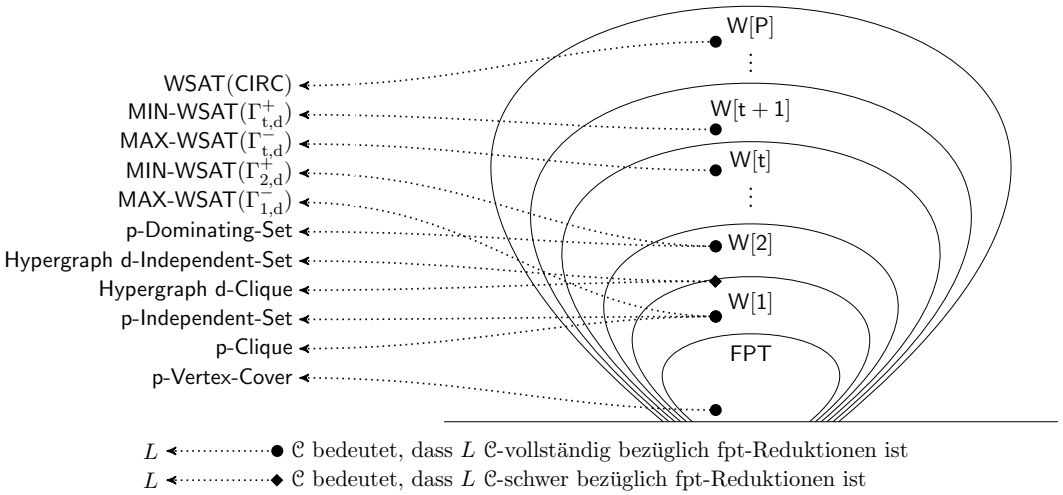



Abbildung (3.1): Übersicht der Vollständigkeits- und Schwereresultate

# 4 Schwierigkeit einzelner inkrementeller Probleme

Da es sich bei inkrementellen Problemen nur um spezielle parametrisierte Probleme handelt, können wir diese ebenso mit den Mitteln der parametrisierten Komplexitätstheorie betrachten.

Wir wollen jedoch keine erschöpfende Rundschau über die Komplexität inkrementeller Probleme geben, sondern einige Phänomene bezüglich der Komplexität beleuchten, die beim Übergang von statischen Problemen auf ihre inkrementelle Variante auftreten. Von besonderem Interesse ist auch die Auswirkung der erlaubten Änderungen auf die Komplexität.

Die meisten Resultate aus diesem Kapitel stammen aus der Arbeit von Mans und Mathieson [MM17] und werden hier in ausführlicherer Form präsentiert. Für Mitgliedschaftsresultate in FPT werden zunächst die Überlegungen zu den Auswirkungen der Änderungen an einer Instanz in einem eigenen Lemma gebracht und dann in einem *fpt*-Algorithmus umgesetzt.

## 4.1 incr-Clique

Auf den ersten Blick stehen einem inkrementellen Algorithmus mehr Informationen zur Verfügung als im statischen Fall. Die Vermutung liegt nahe, dass ein Problem dadurch einfacher wird. Im Fall von incr-Clique wird dies auch tatsächlich bestätigt.

incr-Clique($\Delta$)

| | |
|---|---|
| **Eingabe:** | Graph $G = (V, E)$, $k \in \mathbb{N}$, eine Menge von Änderungen $\delta \subsetneq \Delta$, $S$ ein Zeuge für $(G, k) \in$ p-Clique |
| **Parameter:** | $k + \lvert\delta\rvert$ |
| **Frage:** | Existiert $V' \subseteq V(G \circ \delta)$ mit $\lvert V'\rvert \geq k - \lvert\delta\rvert$, sodass für je zwei $u, v \in V'$ gilt, dass $\{u, v\} \in E(G \circ \delta)$? |

Zunächst betrachten wir die Auswirkungen von Änderungen an einem Graphen auf mögliche Cliquen.

**Lemma 4.1.1** ([MM17, Lemma 5])**.** *Für einen Graphen $G = (V, E)$ und $\delta \subsetneq \Delta$ hat $G \circ \delta$ eine Clique der Größe $\geq k - \lvert\delta\rvert$, falls $G$ eine Clique der Größe $\geq k$ hat.*

*Beweis.* Sei $S \subseteq V$ die Knotenteilmenge mit $|S| \geq k$, die einen vollständigen Teilgraphen induziert. Wir beweisen die Aussage via Induktion über $|\delta|$.

Für den Induktionsanfang betachten wir $|\delta| = 1$.

$\delta = v_i^-$: Es gilt, dass $|S| \geq k - 1$, denn

$$|S \circ v_i^-| = |S \setminus \{v_i\}| = \begin{cases} |S| & \text{, falls } v_i \notin S \\ |S| - 1 & \text{, falls } v_i \in S \end{cases}.$$

Insbesondere bleibt $S \setminus \{v_i\}$ eine Clique der Größe $\leq k-1$, da alle Knoten weiterhin paarweise untereinander verbunden sind.

$\delta = e_{ij}^-$: Falls o. B. d. A. $v_i \in S$ und $v_j \notin S$, also nur einer der inzidenten Knoten in $S$ ist, dann ändert sich $S$ nicht.

Falls $v_i, v_j \in S$, dann sind nach Entfernen der Kante $\{v_i, v_j\} \in E$ die beiden Knoten zwar nicht mehr untereinander, aber dafür immer noch mit allen Knoten $v \in S \setminus \{v_i, v_j\}$ der restlichen Knotenteilmenge verbunden. Wir erhalten damit also zwei neue Knotenmengen $S' = S \setminus \{v_j\}$ und $S'' = S \setminus \{v_i\}$, die ihrerseits einen vollständigen Teilgraphen der Größe $\geq k - 1$ induzieren.

Für die Fälle $\delta = e_{ij}^+$ und $\delta = v_i^+$ bemerken wir, dass solche Operationen die Größe von $S$ nicht verringern können.

Beim Induktionsschritt $|\delta| = |\delta| + 1$ können wir $\delta' = \delta \circ \omega$ zerlegen. Nach Induktionsvorraussetzung hat $G \circ \delta$ eine Clique mit höchstens $k - |\delta|$ Knoten, also hat $G \circ \delta \circ \omega$ eine Clique mit höchstens $k - |\delta| - 1$ Knoten. □

*Beispiel.* Wir betrachten den $K_4$, also den vollständigen Graphen mit vier Knoten. Es ist klar, dass $(K_4, 4) \in$ p-Clique.

Löschen wir nun beispielsweise die Kante $e = \{3, 4\}$, so ist $G = K_4 \circ e_{34}^-$ kein vollständiger Graph mehr, da ja gerade diese Kante $e$ fehlt.


(a) $K_4$ (b) $G = K_4 \circ e_{34}^-$ (c) Mögliche Cliquen der Größe 3 in $G$

Abbildung (4.1): Beispiel zum Löschen einer Kante und der Cliquen im resultierenden Graphen.

$G$ enthält jedoch weiterhin zwei Cliquen (Dreiecke) mit drei Knoten, indem wir entweder den Knoten 3 oder den Knoten 4 in eine Clique aufnehmen, wie in Abbildung 4.1 dargestellt ist.

Auf jeden Fall finden wir in $G$ Cliquen der Größe 3, da $K_4$ eine Clique der Größe 4 enthält.

Wir können nun die Überlegungen aus dem vorherigen Lemma benutzen, um incr-Clique zu entscheiden und die Informationen im Zeugen zu aktualisieren.

**Satz 4.1.2** ([MM17, Lemma 5])**.** incr-Clique$(v^{\pm}, e^{\pm}) \in$ FPT.

*Beweis.* Mit unseren Überlegungen aus Lemma 4.1.1 folgt, falls $(G, k) \in$ p-Clique, dann ist $(G \circ \delta, k - |\delta|) \in$ p-Clique für eine Menge von Änderungen $\delta$. Daraus folgt jedoch auch $(G, k, \delta, S) \in$ incr-Clique für ein $S \subseteq V(G)$, sodass $S$ einen vollständigen Teilgraphen von $G$ induziert.

Folgender Algorithmus entscheidet incr-Clique.

**Eingabe:** $G = (V, E), k \in \mathbb{N}, \delta \subsetneq \Delta, S$

1: **for** $w \in \delta$ **do** ▷ $\mathcal{O}(|\delta|)$
2: **switch** $\omega$ **do**
3: **case** $\omega = v_i^-$: $S \leftarrow S \setminus \{v_i\}$ ▷ $\mathcal{O}(|V|)$
4: **case** $\omega = e_{ij}^-$:
5: **if** $v_i \in S$ **and** $v_j \in S$ **then** $S \leftarrow S \setminus \{v_j\}$ ▷ $\mathcal{O}(|V|)$
6: **return ja**

Die Laufzeit dieses Algorithmus beträgt dann:

$$f(k + |\delta|) \cdot |G|^{\mathcal{O}(1)} + |\delta|(3 \cdot \mathcal{O}(|V|)) \subseteq f(k + |\delta| \cdot (|G|^{\mathcal{O}(1)} \cdot \mathcal{O}(|V|))$$

Also läuft unser Algorithmus in *fpt*-Laufzeit. □

An diesem konkreten Beispiel wird deutlich, warum wir inkrementelle Probleme als Promise-Probleme betrachten. Es wird nämlich nicht überprüft, ob $S$ auch tatsächlich sinnvoll aufgebaut ist, sondern vorrausgesetzt, dass $S$ korrekt ist und direkt damit gearbeitet. Wegen Lemma 4.1.1 muss dieser Algorithmus nur die Änderungen verarbeiten und kann dann akzeptieren.

In diesem Fall kann $S$ sogar geprüft werden, da das klassische Problem Clique NP-vollständig ist und eine Lösung deshalb effizient überprüft werden kann.

Außerdem sehen wir am entsprechenden Algorithmus, dass ein Großteil der Laufzeit darauf verwendet wird, die Informationen im Zeugen zu aktualisieren. Dies ist eine Gemeinsamkeit vieler effizient lösbarer Probleme, da die Frage, ob eine veränderte Instanz auch weiterhin eine Ja-Instanz ist, durch vorherige Argumentation sehr leicht zu lösen ist. Es muss dann nur noch der ursprüngliche Zeuge angepasst werden, um einen gültigen Zeugen für die neue Instanz zu erhalten.

## 4.2 incr-Independent-Set

So wie wir zuvor in Lemma 3.3.3 den Zusammenhang zwischen p-Independent-Set und p-Clique gesehen haben, können wir eine ähnliche Beziehung auch zwischen den

jeweiligen inkrementellen Varianten herstellen. Dabei ist die inkrementelle Version von p-Independent-Set wie folgt definiert.

incr-Independent-Set($\Delta$)
**Eingabe:** Graph $G = (V, E)$, $k \in \mathbb{N}$, eine Menge von Änderungen $\delta \subsetneq \Delta$, $S$ ein Zeuge für $(G, k) \in$ p-Independent-Set
**Parameter:** $k + |\delta|$
**Frage:** Existiert $V' \subseteq V(G \circ \delta)$ mit $|V'| \geq k - |\delta|$, sodass für je zwei $u, v \in V'$ gilt, dass $\{u, v\} \notin E(G \circ \delta)$?

**Korollar 4.2.1** ([MM17, Korollar 2])**.** incr-Independent-Set$(v^{\pm}, e^{\pm}) \in$ FPT

*Beweis.* Wir zeigen, dass incr-Independent-Set$(v^{\pm}, e^{\pm}) \leq^{fpt}$ incr-Clique$(v^{\pm}, e^{\pm})$ via folgender Abbildung

$$\psi \colon \Sigma^* \times \mathbb{N} \to \Gamma^* \times \mathbb{N}$$
$$(G, k, \delta, S) \mapsto (\overline{G}, k, \delta', S)$$

wobei $\delta'$ aus $\delta$ hervorgeht, indem das Löschen und Hinzufügen von Kanten vertauscht wird.

Dann gilt nämlich:

$$(G, k, \delta, S) \in \text{incr-Independent-Set} \iff (G \circ \delta, k - |\delta|) \in \text{p-Independent-Set} \tag{4.1}$$
$$\iff (\overline{G \circ \delta}, k - |\delta|) \in \text{p-Clique} \tag{4.2}$$
$$\iff (\overline{G}, k, \delta', S) \in \text{incr-Clique} \tag{4.3}$$

Die Äquivalenzen in 4.1 und 4.3 gelten wegen der Definition der jeweiligen inkrementellen Probleme und 4.2 gilt wegen Lemma 3.3.3. Und da nach Satz 4.1.2 incr-Clique $\in$ FPT, ist incr-Independent-Set ebenfalls in FPT. □

Nachdem wir für die inkrementellen Versionen von p-Independent-Set und p-Clique Mitgliedschaft in FPT feststellen konnten, stellt sich die Frage, ob sich die Verallgemeinerung auf Hypergraphen ähnlich verhält. Definiert sind die inkrementellen Varianten von Hypergraph d-Independent-Set und Hypergraph d-Clique wie im Folgenden.

incr-Hypergraph d-Clique($\Delta$)
**Eingabe:** Hypergraph $G = (V, E)$, $k \in \mathbb{N}$, eine Menge von Änderungen $\delta \subsetneq \Delta$, $S$ ein Zeuge für $(G, k) \in$ Hypergraph d-Clique
**Parameter:** $k + |\delta|$
**Frage:** Gibt es $V' \subseteq V(G \circ \delta)$ mit $|V'| \geq k - |\delta|$, sodass für alle $e \subseteq V'$ mit $|e| \leq d$ gilt, dass $e \in E$?

incr-Hypergraph d-Independent-Set($\Delta$)
**Eingabe:** Hypergraph $G = (V, E)$, $k \in \mathbb{N}$, eine Menge von Änderungen $\delta \subsetneq \Delta$, $S$ ein Zeuge für $(G, k) \in$ Hypergraph d-Independent-Set
**Parameter:** $k + |\delta|$
**Frage:** Existiert eine Menge $V' \subseteq V(G \circ \delta)$ mit $|V'| \geq k - |\delta|$, so dass für alle $e \in E(G \circ \delta)$ gilt, dass $|e \cap V'| < d$?

Die Situation stellt sich leider als nicht so leicht dar. Dafür benötigen wir noch Resultate bezüglich inkrementeller Varianten der gewichteten Erfüllbarkeitsprobleme, die wir nun im Folgenden betrachten wollen.

## 4.3 incr-WSAT

Die inkrementellen Versionen der Varianten von WSAT sind wie folgt definiert.

incr-MAX-WSAT($\Phi, \Delta$)
**Eingabe:** Eine Formel $\phi \in \Phi$, $k \in \mathbb{N}$, eine Menge von Änderungen $\delta \subsetneq \Delta$, Zeuge $S$ für $(\phi, k) \in$ MAX-WSAT($\Phi$)
**Parameter:** $k + |\delta|$
**Frage:** Hat $\phi \circ \delta$ eine erfüllende Belegung mit Gewicht $\geq k - |\delta|$?

incr-MIN-WSAT($\Phi, \Delta$)
**Eingabe:** Eine Formel $\phi \in \Phi$, $k \in \mathbb{N}$, eine Menge von Änderungen $\delta \subsetneq \Delta$, Zeuge $S$ für $(\phi, k) \in$ MIN-WSAT($\Phi$)
**Parameter:** $k + |\delta|$
**Frage:** Hat $\phi \circ \delta$ eine erfüllende Belegung mit Gewicht $\leq k + |\delta|$?

Eine effizient lösbare Variante dieser Probleme erhalten wir zunächst aus dem Zusammenhang zwischen p-Independent-Set und MAX-WSAT($\Gamma^-_{1,2}$).

**Korollar 4.3.1** ([MM17, Korollar 3])**.** incr-MAX-WSAT($\Gamma^-_{1,2}, var^-, cl^-$) $\in$ FPT.

*Beweis.* Wir können die Abbildung $\varphi$ aus Lemma 3.2.1 nutzen und müssen nur die zusätzlichen Eingabeparameter berücksichtigen. Zur Erinnerung, $\varphi$ ist definiert als

$$\begin{aligned} \varphi \colon \Sigma^* \times \mathbb{N} &\to \Gamma^* \times \mathbb{N} \\ (\phi, k) &\mapsto (G, k) \end{aligned}$$

für $V(G) = \mathsf{Var}(\phi)$ und $E(G) = \mathrm{cl}(\phi)$. Insbesondere ist $\phi$ hier eine Formel in 2-KNF, die nur negierte Literale enthält, also $\phi = \bigwedge_i (\neg \ell_{i1} \vee \neg \ell_{i2})$.

Wir zeigen also $\mathsf{incr\text{-}MAX\text{-}WSAT}(\Gamma^-_{1,2}, \mathsf{var}^-, \mathsf{cl}^-) \leq^{fpt} \mathsf{incr\text{-}Independent\text{-}Set}$ anhand der Abbildung

$$\psi \colon \Sigma^* \times \mathbb{N} \times \mathcal{P}(\Delta) \times \{0,1\}^* \to \Gamma^* \times \mathbb{N} \times \mathcal{P}(\Delta) \times \{0,1\}^*$$
$$(\phi, k, \delta, S) \mapsto (G, k, \delta', S),$$

wobei wir die Änderungen $\delta'$ der neuen Instanz folgendermaßen erhalten

$$\omega'_i = \begin{cases} v_i^- & \text{, falls } \omega_i = var_i^- \\ e_{ij}^- & \text{, falls } \omega_i = cl^-_{\{x_i, x_j\}} \end{cases}.$$

Insgesamt ergibt sich, dass

$$\begin{aligned} & (\phi, k, \delta, S) && \in \mathsf{incr\text{-}MAX\text{-}WSAT}(\Gamma^-_{1,2}, var^-, cl^-) \\ \iff & (\phi \circ \delta, k - |\delta|) && \in \mathsf{MAX\text{-}WSAT}(\Gamma^-_{1,2}) \\ \iff & (G \circ \delta', k - |\delta|) && \in \mathsf{p\text{-}Independent\text{-}Set} \\ \iff & (G, k, \delta', S) && \in \mathsf{incr\text{-}Independent\text{-}Set}. \end{aligned} \tag{4.4}$$

Für 4.4 nutzen wir $\varphi$, da wir Instanzen für das statische $\mathsf{MAX\text{-}WSAT}(\Gamma^-_{1,2})$ auf $\mathsf{p\text{-}Independent\text{-}Set}$ reduzieren können. □

Bei diesem Resultat handelt es sich jedoch um einen Spezialfall. Im Allgemeinen bleiben die inkrementellen Versionen der WSAT-Varianten nur unter folgender Bedingung einfach.

**Lemma 4.3.2** ([MM17, Lemma 8])**.** $\mathsf{incr\text{-}\{MAX, MIN\}\text{-}WSAT}(\Phi, cl^-) \in \mathsf{FPT}$

*Beweis.* Falls $\phi \in \Phi$ durch eine Belegung $\mathfrak{I}$ erfüllt wird, dann wird auch jede Teilformel $\phi'$, die durch Löschen von Klauseln aus $\phi$ hervorgeht, von $\mathfrak{I}$ erfüllt [MM17]. □

Die Schwierigkeit, das Löschen von Variablen zu behandeln, wollen wir an folgendem Beispiel verdeutlichen.

*Beispiel.* Wir betrachten folgende Formel in 3-KNF als Instanz von $\mathsf{MAX\text{-}WSAT}(\Gamma^-_{1,d})$ mit $k = 3$.

$$\phi \equiv (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_5)$$

Für diese Formel erhalten wir eine erfüllende Belegung $\mathfrak{I}$ mit Gewicht $4 \geq k$, indem wir die Variablen $x_2, \ldots, x_5$ auf Wahr setzen.

Löschen wir jedoch die Variable $x_1$, erhalten wir folgende Formel $\phi'$.

$$\phi \equiv (\neg x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_5)$$

Auch wenn $\mathfrak{I}$ weiterhin eine Belegung für $\phi'$ mit Gewicht $\geq 2 = 3 - 1$ ist, so ist sie doch keine *erfüllende Belegung* mehr. Die Informationen aus $\mathfrak{I}$ können auch nicht ohne Weiteres wiederverwenden, da aus algorithmischer Sicht nicht offensichtlich ist, welche Variablen nun auf FALSCH gesetzt werden müssen, um eine erfüllende Belegung zu erhalten.

Die entsprechenden Beweise der folgenden Resultate haben wir wegen der verwendeten Konstruktion aus dem Artikel von Mans und Mathieson entnommen [MM17].

**Lemma 4.3.3** ([MM17, Lemma 9])**.** *Für $d > 2$ ist* incr-MAX-WSAT$(\Gamma^-_{1,d}, var^-)$ W[1]*-vollständig.*

*Beweis.* Zunächst zeigen wir MAX-WSAT$(\Gamma^-_{1,d-1}) \leq^{fpt}$ incr-MAX-WSAT$(\Gamma^-_{1,d}, var^-)$. Sei $(\phi, k)$ eine Instanz von MAX-WSAT$(\Gamma^-_{1,d-1})$ und $y$ eine neue Variable, die nicht in $\phi$ enthalten ist. Außerdem können wir annehmen, dass $k < \mathsf{Var}(\phi)$ ist, da sonst $(\phi, k) \notin$ MAX-WSAT$(\Gamma^-_{1,d-1})$ und wir in diesem Fall auf eine triviale Instanz abbilden können.

Sonst konstruieren wir eine neue Formel $\phi'$, indem wir $y$ jeder Klausel hinzufügen. Ist also $\phi = \bigwedge_{i \in I} \bigvee_{j=1}^{d-1} \neg\lambda_{i,j}$, dann ist $\phi' = \bigwedge_{i \in I} \left( \bigvee_{j=1}^{d-1} \neg\lambda_{i,j} \vee \neg y \right)$.

Nun sehen wir, dass $\phi'$ durch eine Belegung $\mathfrak{I}'$ erfüllt wird, die jede Variable $x \in \mathsf{Var}(\phi)$ auf WAHR und $y$ auf FALSCH setzt, denn $y$ ist in jeder Klausel von $\phi'$ enthalten. Setzen wir die Änderungen $\delta = \{var^-_y\}$, dann erhalten wir mit der Abbildung

$$\begin{aligned} \psi \colon \Sigma^* \times \mathbb{N} &\to \Gamma^* \times \mathbb{N} \times \mathcal{P}(var^-) \times \{0,1\}^* \\ (\phi, k) &\mapsto (\phi', k+1, \delta, \mathfrak{I}'), \end{aligned}$$

eine *fpt*-Reduktion.

Falls $(\phi, k) \in$ MAX-WSAT$(\Gamma^-_{1,d-1})$, dann hat $\phi$ eine erfüllende Belegung $\mathfrak{I}$ mit einem Gewicht von mindestens $k$. Und da $y \notin \mathsf{Var}(\phi)$, kann eine solche Belegung $y$ nicht enthalten. Durch das Löschen der Variable $y$ ändert sich das Gewicht von $\mathfrak{I}$ also nicht.

Daher ist $\mathfrak{I}$ auch eine erfüllende Belegung von $\phi' \circ \delta = \phi$ mit Gewicht $k + 1 - |\delta| = k$, also

$$(\phi', k+1, \delta, \mathfrak{I}') \in \text{incr-MAX-WSAT}(\Gamma^-_{1,d}, var^-).$$

Umgekehrt gilt ebenfalls, wenn $(\phi', k+1, \delta, \mathfrak{I}') \in$ incr-MAX-WSAT$(\Gamma^-_{1,d}, var^-)$, dann ist $(\phi' \circ \delta, k+1-|\delta|) = (\phi, k) \in$ MAX-WSAT$(\Gamma^-_{1,d})$. Da $\phi'$ jedoch eine Formel mit $d$ Variablen je Klausel ist und $\phi$ aus $\phi'$ durch das Löschen der Variable $y$ hervorgeht, die in jeder Klausel von $\phi'$ und sonst nicht in $\phi$ enthalten ist, ist insbesondere $(\phi, k) \in$ MAX-WSAT$(\Gamma^-_{1,d-1})$.

Daher folgt zusammen mit Lemma 2.3.4

$$\text{MAX-WSAT}(\Gamma^-_{1,d-1}) \equiv^{fpt} \text{incr-MAX-WSAT}(\Gamma^-_{1,d}, var^-). \qquad \square$$

Aus diesem Resultat folgt auch die Schwierigkeit der inkrementellen Hypergraphprobleme.

**Korollar 4.3.4** ([MM17, Korollar 4]). incr-Hypergraph d-Independent-Set($v^-$) *ist für* $d > 2$ W[1]*-schwer.*

*Beweis.* Wir zeigen analog zu Korollar 4.3.1, dass

$$\text{incr-MAX-WSAT}(\Gamma^-_{1,d}, var^-) \leq^{fpt} \text{incr-Hypergraph d-Independent-Set}(v^-).$$

Wegen $d \neq 2$ ist incr-MAX-WSAT($\Gamma^-_{1,d}, var^-$) jedoch W[1]-vollständig, daher ist incr-Hypergraph d-Independent-Set($v^-$) W[1]-schwer. □

**Korollar 4.3.5** ([MM17, Korollar 4]). incr-Hypergraph d-Clique($v^-$) *ist für* $d > 2$ W[1]*-schwer.*

*Beweis.* Wir zeigen dazu

$$\text{incr-MAX-WSAT}(\Gamma^-_{1,d}, var^-) \leq^{fpt} \text{incr-Hypergraph d-Independent-Set}(v^-),$$

indem wir die Abbildung $\phi$ aus Lemma 3.3.1 nutzen, die wir zur Erinnerung noch einmal aufführen.

$$\begin{aligned} \varphi\colon \Sigma^* \times \mathbb{N} &\to \Gamma^* \times \mathbb{N} \\ (\phi, k) &\mapsto (G, k), \end{aligned}$$

wobei $V(G) = \mathsf{Var}(\phi)$ und $E(G) = \left( \bigcup_{i=1}^{d} \mathsf{Var}(\phi)^i \right) \setminus \mathrm{cl}(\phi)$.

Wir können nun eine weitere Abbildung $\psi$ konstruieren, die zusätzlich die Änderungen $\delta$ an $\phi$ und den Zeugen $S$ anpasst.

$$\begin{aligned} \psi\colon \Sigma^* \times \mathbb{N} \times \mathcal{P}(var^-) \times \{0,1\}^* &\to \Gamma^* \times \mathbb{N} \times \mathcal{P}(v^-) \times \{0,1\}^* \\ (\phi, k, \delta, S) &\mapsto (G, k, \delta', S), \end{aligned}$$

Dadurch erhalten wir

$$\begin{aligned} & (\phi, k, \delta, S) && \in \text{incr-MAX-WSAT}(\Gamma^-_{1,d}, var^-) \\ \iff & (\phi \circ \delta, k - |\delta|) && \in \text{MAX-WSAT}(\Gamma^-_{1,d}) \\ \iff & (G \circ \delta', k - |\delta|) && \in \text{Hypergraph d-Independent-Set} \\ \iff & (G, k, \delta', S) && \in \text{incr-Hypergraph d-Independent-Set}(v^-). \end{aligned} \tag{4.5}$$

Für 4.5 nutzen wir die Eigenschaft von $\varphi$ als *fpt*-Reduktion. □

Für den folgenden Beweis können wir eine ähnliche Konstruktion verwenden, wie im Beweis von Lemma 4.3.3, und müssen die Argumentation nur etwas anpassen.

**Lemma 4.3.6** ([MM17, Lemma 10]). incr-MIN-WSAT($\Gamma^+_{2,1}, var^-$) *ist* W[2]*-schwer.*

*Beweis.* Wir zeigen $\mathsf{MIN\text{-}WSAT}(\Gamma^+_{2,1}) \leq^{fpt} \mathsf{incr\text{-}MIN\text{-}WSAT}(\Gamma^+_{2,1}, var^-)$.

Es sei also $(\phi, k)$ eine Instanz von $\mathsf{MIN\text{-}WSAT}(\Gamma^+_{2,1})$ und $x$ eine neue Variable, die nicht in $\phi$ enthalten ist. Wir betrachten zunächst den allgemeinen Fall und behandeln dann die Spezialfälle.

*Fall $k \geq 2$:* Wir konstruieren wieder eine Formel $\phi'$, indem wir jeder Klausel von $\phi$ die Variable $y$ hinzufügen. Ist $\phi = \bigwedge_{i \in I_1} \bigvee_{j \in I_2} \lambda_{i,j}$, dann ist $\phi' = \bigwedge_{i \in I_1} \bigvee_{j \in I_2} (\lambda_{i,j} \vee y)$.

Nun sehen wir, dass $\phi'$ durch eine Belegung $\mathfrak{I}'$ erfüllt wird, die jede Variable $x \in \mathsf{Var}(\phi)$ auf Falsch und $y$ auf Wahr setzt, da $y$ in jeder Klausel von $\phi'$ enthalten ist. Vor allem hat $\mathfrak{I}'$ ein Gewicht von $1 < k$, ist also tatsächlich ein korrekter Zeuge für $(\phi', k-1) \in \mathsf{MIN\text{-}WSAT}(\Gamma^+_{2,1})$.

Setzen wir die Änderungen $\delta = \{var^-_v\}$, dann erhalten wir mit der Abbildung

$$\begin{aligned} \psi \colon \Sigma^* \times \mathbb{N} &\to \Gamma^* \times \mathbb{N} \times \mathcal{P}(v^-) \times \{0,1\}^* \\ (\phi, k) &\mapsto (\phi', k-1, \delta, \mathfrak{I}'), \end{aligned}$$

eine *fpt*-Reduktion.

Ist $(\phi, k) \in \mathsf{MAX\text{-}WSAT}(\Gamma^+_{2,1})$, dann hat $\phi$ eine erfüllende Belegung $\mathfrak{I}$ mit einem Gewicht von höchstens $k$. Damit ist $\mathfrak{I}$ jedoch auch eine erfüllende Belegung für $\phi' \circ \delta = \phi$ mit einem Gewicht von höchstens $k - 1 + |\delta| = k$. Daher ist

$$(\phi', k-1, \delta, \mathfrak{I}') \in \mathsf{incr\text{-}MIN\text{-}WSAT}(\Gamma^+_{2,1}, v^-).$$

Falls nun $(\phi', k-1, \delta, \mathfrak{I}') \in \mathsf{incr\text{-}MIN\text{-}WSAT}(\Gamma^+_{2,1}, v^-)$, dann hat $\phi = \phi' \circ \delta$ eine erfüllende Belegung mit einem Gewicht von höchstens $k - 1 + |\delta| = k$, also $(\phi, k) \in \mathsf{MAX\text{-}WSAT}(\Gamma^+_{2,1})$.

*Fall $k = 0$:* Wir können $(\phi, k)$ auf eine triviale Nein-Instanz abbilden, da $\phi$ nur positive Literale enthält und nicht durch eine Belegung mit Gewicht 0 erfüllt werden kann.

*Fall $k = 1$:* In diesem Fall existiert möglicherweise eine erfüllende Belegung für $\phi$, die genau eine Variable auf Wahr setzt. $(\phi, 1) \notin \mathsf{MAX\text{-}WSAT}(\Gamma^+_{2,1})$ gilt also nicht trivialerweise.

Wir können jedoch in angemessener Laufzeit für jede Variable $x \in \mathsf{Var}(\phi)$ überprüfen, ob sie zu einer erfüllenden Belegung $\mathfrak{I}$ mit Gewicht 1 gehört. Das Überprüfen einer Belegung ist in $|\phi|^{\mathcal{O}(1)}$ möglich, da das Erfüllbarkeitsproblem für aussagenlogische Formeln in NP ist, also polynomiell überprüfbar [Sip06, Theorem 7.37].

Da wir für jede Variable eine solche Belegung prüfen wollen und die Anzahl der Variablen durch die Formellänge beschränkt ist, ist dies also insgesamt auch in Polynomialzeit möglich und insbesondere in *fpt*-Laufzeit.

Falls diese Überprüfung ergibt, dass $(\phi, 1) \in \mathsf{MAX\text{-}WSAT}(\Gamma^+_{2,1})$, dann können wir auf die Instanz $(\phi', 1, \delta, S)$ abbilden, mit $\phi'$, $\delta$ und $S$ wie im Fall $k \geq 2$. Wegen $(\phi, 1) \in \mathsf{MAX\text{-}WSAT}(\Gamma^+_{2,1})$, gilt auch $(\phi, 2) \in \mathsf{MAX\text{-}WSAT}(\Gamma^+_{2,1})$, da $\phi$ nur positive Literale enthält. Deshalb gilt aber auch

$$(\phi', 1, \delta, \mathfrak{I}') \in \mathsf{incr\text{-}MIN\text{-}WSAT}(\Gamma^+_{2,1}, var^-).$$

Und falls $(\phi, 1) \notin \mathsf{MAX\text{-}WSAT}(\Gamma^+_{2,1})$, dann können wir wieder auf eine triviale Nein-Instanz abbilden.

Damit ist $\mathsf{incr\text{-}MIN\text{-}WSAT}(\Gamma^+_{2,1}, var^-)$ W[2]-schwer. Die W[2]-Vollständigkeit erhalten wir wieder durch Lemma 2.3.4. □

Wir sehen anhand der Argumentation, dass wir durch das Löschen von Variablen die Struktur unserer Instanz grundlegend verändern können und die Informationen aus dem Zeugen hinfällig werden.

So waren die jeweiligen Belegungen $\mathfrak{I}'$ aus den Beweisen für Lemmata 4.3.3 und 4.3.6 zwar für die (im Sinne des inkrementellen Problems) ursprüngliche Instanz $\phi'$ eine erfüllende Belegung, für die geänderte Instanz $\phi' \circ \delta$ jedoch keine erfüllende Belegung mehr.

Also haben wir in diesem Fall durch die vermeintlich zusätzlichen Informationen keinen wirklichen Vorteil. In der Tat bleibt es für die gesamte W-Hierarchie schwer, das Löschen von Variablen zu behandeln.

**Lemma 4.3.7** ([MM17, Lemma 11])**.** *Für jedes $t \in \mathbb{N}$ existiert ein* W[t]*-vollständiges Problem. Konkret gilt:*

1. $\mathsf{incr\text{-}MAX\text{-}WSAT}(\Gamma^-_{t,d}, var^-)$ *ist* W[$t$]*-vollständig für jedes ungerade $t \geq 1$.*
2. $\mathsf{incr\text{-}MAX\text{-}WSAT}(\Gamma^+_{t,1}, var^-)$ *ist* W[$t$]*-vollständig für jedes gerade $t \geq 2$.*

*Beweis.*

*1.* Ist $\phi \in \Gamma^-_{t,d}$, so hat $\phi$ die Form

$$\phi = \bigwedge_{I_t} \left( \bigvee_{I_{t-1}} \left( \cdots \left( \bigvee_{I_2} \phi_\alpha \right) \cdots \right) \right)$$

für $\alpha \in I_t \times I_{t-1} \times \cdots \times I_2$ und $\phi_\alpha = \bigwedge_{i \in I_\alpha} \bigvee_{j=1}^{d} \neg \ell_{\alpha,i,j} \in \Gamma^-_{1,d}$.

Wir können also für eine Variable $v$, die in keiner der $\phi_\alpha$ enthalten ist, die Konstruktion aus dem Beweis zu Lemma 4.3.3 auf jede der Teilformeln $\phi_\alpha$ anwenden. Ansonsten verläuft die Argumentation analog.

*2.* Für $\phi \in \Gamma^+_{t,1}$ gilt, dass

$$\phi = \bigwedge_{I_t} \left( \bigvee_{I_{t-1}} \left( \cdots \left( \bigvee_{I_3} \phi_\alpha \right) \cdots \right) \right)$$

für $\alpha \in I_t \times I_{t-1} \times \cdots \times I_3$ und $\phi_\alpha = \bigwedge_{i \in I_\alpha} \bigvee_{j \in I_{\alpha,i}} \ell_{\alpha,i,j} \in \Gamma^+_{2,1}$.

Auch hier können wir für eine Variable $v$, die in keiner der $\phi_\alpha$ enthalten ist, die Konstruktion aus dem Beweis zu Lemma 4.3.6 auf jede der Teilformeln $\phi_\alpha$ anwenden und ebenfalls analog argumentieren. □

Im Folgenden gehen wir davon aus, dass für eine Boole'sche Funktion $\circ \in \{\wedge, \vee\}$ und einen Schaltkreis $\mathcal{C}$, der $x \circ y$ berechnet, der Schaltkreis $\mathcal{C}'$, der aus $\mathcal{C}$ durch Löschen von $y$ hervorgeht, die Funktion $x$ berechnet. Wir sehen dann, dass das Löschen von Variablen auch für Boole'sche Schaltkreise schwer bleibt. Den Beweis dazu haben wir auch aus der Arbeit von Mans und Mathieson [MM17] übernommen, jedoch den Zeugen für die Instanz von incr-WSAT(CIRC) angepasst.

**Lemma 4.3.8** ([MM17, Lemma 12])**.** incr-WSAT(CIRC) *ist* W[P]*-vollständig.*

*Beweis.* Für die W[P]-Schwere zeigen wir WSAT(CIRC) $\leq^{fpt}$ incr-WSAT(CIRC). Sei also $\mathcal{C}$ ein Boole'scher Schaltkreis. Wir konstruieren den Schaltkreis $\mathcal{C}'$, indem wir ein $\vee$-Gatter mit $\mathcal{C}$ und einem neuen Eingabegatter $x$ verdrahten.

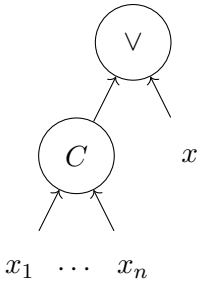



Abbildung (4.2): Schaltkreis $\mathcal{C}'$

Wir sehen, dass jede Belegung $\mathfrak{I}$ , die $x$ auf Wahr setzt, eine erfüllende Belegung für $\mathcal{C}$ ist. Insbesondere erhalten wir eine erfüllende Belegung $\mathfrak{I}_k$ für $\mathcal{C}'$ mit Gewicht $k$, indem wir die Variablen $x_1, \ldots, x_{k-1}$ und $x$ auf Wahr setzen

Dann ist folgende Abbildung $\varphi$ ist eine *fpt*-Reduktion.

$$\varphi \colon \Sigma^* \times \mathbb{N} \to \Gamma^* \times \mathbb{N} \times \mathcal{P}(v^-) \times \{0,1\}^*$$
$$(\mathcal{C}, k) \mapsto (\mathcal{C}', k-1, \{var_x^-\}, \mathfrak{I}_{k-1}),$$

Nach Definition ist genau dann $(\mathcal{C}', k-1, \{var_x^-\}, \mathfrak{I}_{k-1}) \in$ incr-WSAT(CIRC), wenn $\mathcal{C}' \circ var_x^-$ eine erfüllende Belegung $\mathfrak{I}$ mit Gewicht $k - 1 + |\delta| = k$ besitzt. Da jedoch $\mathcal{C}' \circ var_x^-$ äquivalent zu $\mathcal{C}$ ist, erhalten wir also insgesamt, dass genau dann $(\mathcal{C}', k-1, \{var_x^-\}, \mathfrak{I}_{k-1}) \in$ incr-WSAT(CIRC), wenn $(\mathcal{C}, k) \in$ WSAT(CIRC). □

## 4.4 incr-Dominating-Set

Die inkrementelle Variante von p-Dominating-Set ist insofern interessant, als nur eine bestimmte Klasse von Änderungen schwer zu behandeln ist. Alle anderen Klassen von Änderungen sind leicht zu handhaben.

incr-Dominating-Set($\Delta$)
**Eingabe:** Graph $G = (V, E)$, $k \in \mathbb{N}$, eine Menge von Änderungen $\delta \subsetneq \Delta$, $S$ ein Zeuge für $(G, k) \in$ p-Dominating-Set
**Parameter:** $k$
**Frage:** Existiert eine dominierende Menge $V' \subseteq V(G \circ \delta)$ mit $|V'| \geq k + |\delta|$?

Zunächst betrachten wir die leicht handhabbaren Fälle und wie sie konkret zu behandeln sind.

**Lemma 4.4.1** ([MM17, Lemma 7])**.** *Sei $G = (V, E)$ ein Graph, $k \in \mathbb{N}$ und $\delta \subsetneq \Delta$ eine Menge von Änderungen an $G$, die keinen Knoten löschen. Falls $G$ eine dominierende Knotenmenge der Größe $\leq k$ hat, dann hat $G \circ \delta$ eine dominierende Knotenmenge der Größe $\leq k + |\delta|$.*

*Beweis.* Es sei $V'$ eine dominierende Knotenmenge von $G$ der Größe $\leq k$. Wir beweisen die Aussage mittels Induktion über $|\delta|$. Für den Induktionsanfang betrachten wir $|\delta| = 1$.

$\delta = v_i^+$: $v_i$ wird als isolierter Knoten hinzugefügt und hat daher auch keine Verbindungskanten zu irgendeinem Knoten aus $V'$. Wir fügen $v_i$ also $V'$ hinzu und erhalten damit eine dominierende Knotenmenge für $G \circ \delta$ der Größe $|V' \cup \{v_i\}| \leq k + |\delta| = k + 1$.

$\delta = e_{ij}^+$: Falls wir eine Kante hinzufügen, bleiben alle Knoten außerhalb von $V'$ mit Knoten aus $V'$ verbunden, da wir nur eine weitere mögliche Verbindung angeben. $V'$ ist also auch für $G \circ \delta$ eine dominierende Knotenmenge mit höchstens $k \leq k + 1$ Knoten.

$\delta = e_{ij}^-$: Falls die Kante $\{v_i, v_j\} \in E$ gelöscht wird und o. B. d. A. $v_i \in V'$, dann füge auch $v_j$ zu $V'$ hinzu. $V' \cup \{v_j\}$ ist also eine dominierende Knotenmenge für $G \circ \delta$ mit höchstens $k + 1$ Knoten.

Falls $v_i, v_j \in V'$ oder $v_i, v_j \notin V'$, dann muss $V'$ nicht geändert werden.

Nun ist $\delta' = \delta \circ \omega$ für $\omega \in \Delta$. Nach Induktionsvorraussetzung hat $G \circ \delta$ eine dominierende Knotenmenge mit höchstens $k + |\delta|$ Knoten, falls $G$ bereits eine solche Menge der Größe höchstens $k$ hat. Für $G \circ \delta \circ \omega$ können wir aber die Argumentation aus dem Induktionsanfang nutzen und erhalten, dass auch $G \circ \delta \circ \omega$ eine dominierende Knotenmenge mit höchstens $k + |\delta| + 1$ Knoten haben muss. □

Wir haben bei dieser Betrachtung das Löschen von Knoten ausgeschlossen. In der Tat sehen wir im folgenden Lemma, dass incr-Dominating-Set sonst genauso schwierig ist wie seine statische Variante.

**Satz 4.4.2** ([MM17, Lemmata 6 und 7]). *Falls* $v^- \subset \Delta$, *dann ist* incr-Dominating-Set($\Delta$) W[2]*-vollständig, sonst in* FPT.

*Beweis.* Für den *Fall* $v^- \not\subset \Delta$ geben wir folgenden *fpt*-Algorithmus an, der incr-Dominating-Set($v^+, e^\pm$) entscheidet.

**Eingabe:** $G = (V, E), k \in \mathbb{N}, \delta \subsetneq \Delta, V'$
1: Prüfe, ob $(G, k) \in$ p-Dominating-Set für $V'$
2: Falls nicht, **return nein**
3: **for** $\omega \in \delta$ **do** ▷ $\mathcal{O}(|\delta|)$
4: **switch** $\omega$ **do**
5: **case** $\omega = v_i^+$: $V' \leftarrow V' \cup \{v_i\}$ ▷ $\mathcal{O}(k)$
6: **case** $\omega = e_{ij}^-$:
7: **if** $v_i \in V'$ **then** $V' \leftarrow V' \cup \{v_j\}$ ▷ $\mathcal{O}(k \cdot |V|)$
8: **else** $V' \leftarrow V' \cup \{v_i\}$ ▷ $\mathcal{O}(k \cdot |V|)$
9: **return ja**

Die Korrektheit der Anweisungen im **switch**-Block ergibt sich aus Lemma 4.4.1. Und da p-Dominating-Set $\in$ W[2], ist das Überprüfen des Zeugen in FPT-Laufzeit möglich.

Aus dieser Überprüfung folgt auch, dass $|V'| \leq k$, womit Operationen auf $V'$ in der Laufzeit durch $k + |\delta|$ beschränkt sind, aufgrund der höchstens $|\delta|$ Knoten, die noch hinzugefügt werden. Insgesamt hat der Algorithmus also eine Laufzeit von

$$\begin{aligned} & f(k+|\delta|) \cdot |G|^{\mathcal{O}(1)} + \mathcal{O}(|\delta|) \cdot \mathcal{O}((k+|\delta|) \cdot |V|) \\ \leq\ & f(k+|\delta|) \cdot |G|^{\mathcal{O}(1)} + \mathcal{O}(|\delta| \cdot (k+|\delta|)) \cdot \mathcal{O}(|V|) \\ \leq\ & f(k+|\delta|) \cdot |G|^{\mathcal{O}(1)} + \mathcal{O}((k+|\delta|)^2) \cdot \mathcal{O}(|V|) \\ \leq\ & \big(f(k+|\delta|) + \mathcal{O}((k+|\delta|)^2)\big) \cdot |G|^{\mathcal{O}(1)}, \end{aligned}$$

also eine für einen *fpt*-Algorithmus erlaubte Laufzeit.

Nun ist $v^- \subset \Delta$. Wir beweisen die folgende Aussage.

$$\text{p-Dominating-Set} \leq^{fpt} \text{incr-Dominating-Set}(\Delta)$$

Sei $(G, k)$ eine Instanz von p-Dominating-Set und $u$ ein Knoten, der nicht in $G$ enthalten ist. Dann konstruieren wir einen Graph $G'$ wie folgt. Wir setzen $V(G') = V(G) \cup \{u\}$ und $E(G') = E(G) \cup \{\, \{v, u\} \mid \forall v \in V(G) \,\}$. Die Menge $V' = \{u\}$ ist eine dominierende Knotenmenge der Größe $\leq k$ für $G'$, da jeder Knoten in $G'$ mit $u$ verbunden ist.

Dann ist folgende Abbildung eine *fpt*-Reduktion auf incr-Dominating-Set($\Delta$).

$$\begin{aligned} \varphi\colon \Sigma^* \times \mathbb{N} &\to \Sigma^* \times \mathbb{N} \times \mathcal{P}(\Delta) \times \{0,1\}^* \\ (G, k) &\mapsto (G', k-1, \{v_u^-\}, \{u\}) \end{aligned}$$

Es gilt also, dass genau dann $(G', k-1, \{v_u^-\}, \{u\}) \in$ incr-Dominating-Set($\Delta$), wenn $G = G' \circ v_u^-$ eine dominierende Knotenmenge mit höchstens $k - 1 + |\delta| = k$ Knoten hat. □

Auch hier sieht man am Ansatz der Reduktion, dass wir zwar für einen Graphen eine dominierende Knotenmenge finden können und entsprechend als Zeugen nutzen können. Durch das Löschen von Knoten kann sich die Struktur dieser Instanz jedoch verändern und die vorherige Lösung dadurch unbrauchbar werden.

## 4.5 incr-Vertex-Cover

Nachdem wir nun vorwiegend abstrakte Resultate zur Komplexität von inkrementellen Problemen betrachtet haben, konstruieren wir nun zu einem konkreten Algorithmus eine inkrementelle Variante und untersuchen, inwiefern wir die Laufzeit dadurch verbessern.

Dazu knüpfen wir an die Resultate aus Abschnitt 3.5 an und gelangen zu einer inkrementellen Variante des Algorithmus aus Satz 3.5.6. Die folgenden Resultate orientieren sich dabei an der Arbeit von Mans und Mathieson [MM17, Abschnitt 4].

Zunächst definieren wir das inkrementelle Problem, eine Knotenüberdeckung zu finden.

incr-Vertex-Cover($\Delta$)

| | |
|---|---|
| **Eingabe:** | Graph $G = (V, E)$, $k \in \mathbb{N}$, eine Menge von Änderungen $\delta \subsetneq \Delta$<br>$S$ ein Zeuge für $(G, k) \in$ p-Vertex-Cover |
| **Parameter:** | $k$ |
| **Frage:** | Hat $G$ eine Knotenüberdeckung der Größe $\leq k + \lvert\delta\rvert$? |

Nun erinnern wir uns an die Überlegungen, die wir zu Knotenüberdeckungen im statischen Fall gemacht haben. Sei $G = (V, E)$ ein Graph. Folgende Aussagen aus Abschnitt 3.5 sind äquivalent.

- $(G, k) \in$ p-Vertex-Cover.
- Für einen Knoten $v_i$ mit $\deg(v_i) > k$ sei $G \circ v_i^-$ der Graph, der durch Entfernen von $v_i$ und der inzidenten Kanten entsteht. Dann ist $(G \circ v_i^-, k-1) \in$ p-Vertex-Cover. (Lemma 3.5.1)
- Für einen Knoten $v_i \in V$ mit $\deg(v_i) = 0$ ist $(G, k)$ genau dann in p-Vertex-Cover, wenn $(G \circ v_i^-, k) \in$ p-Vertex-Cover. (Lemma 3.5.2)
- Sei $\deg(v) \leq k$ für alle $v \in V$. Dann ist $|V| \leq k(k+1)$. (Lemma 3.5.4)

Nun machen wir folgende Beobachtungen in Bezug auf Änderungen an einem Graphen und deren Einfluss auf eine mögliche Knotenüberdeckung. Das Lemma orientiert sich dabei an Aussagen aus Abschnitt 4 bei Mans und Mathieson [MM17].

**Lemma 4.5.1.** *Es sei $k \in \mathbb{N}$, $G = (V, E)$ ein Graph mit einer Knotenüberdeckung der Größe $\leq k$ und $\delta \subsetneq \Delta$ eine Menge von Änderungen an $G$. Dann hat $G \circ \delta$ eine Knotenüberdeckung der Größe $\leq k + \delta$.*

*Beweis.* Sei $U \subseteq V$ die Kontenüberdeckung von $G$ mit $|U| \leq k$. Wir beweisen die Aussage via Induktion über $|\delta|$.

Für den Induktionsanfang betachten wir $|\delta| = 1$.

$\delta = e_{ij}^+$: Falls $v_i, v_j \notin U$, dann müssen wir einen der beiden Knoten zu $U$ hinzufügen, um eine Knotenüberdeckung für $G \circ \delta$ zu erhalten. Sonst ist die hinzugefügte Kante bereits inzident zu $U$ und wir müssen $U$ nicht verändern. Insgesamt finden wir eine mögliche Knotenüberdeckung für $G \circ \delta$ höchstens der Größe $k + 1$.

$\delta = e_{ij}^-$: Falls $v_i, v_j \in U$, dann können beide Knoten immer noch inzident zu weiteren Kanten sein und $U$ bleibt damit eine Knotenüberdeckung.

Sonst ist o. B. d. A. nur $v_i \in U$. Falls $v_i$ keine Nachbarn außerhalb von $U$ hat, also $N(v_i) \subseteq U$, dann ist $U \setminus \{v_i\}$ ebenfalls eine Knotenüberdeckung der Größe $k - 1$.

$\delta = v_i^+$: Der neu hinzugefügte Knoten ist isoliert und muss deshalb nicht betrachtet werden. $U$ bleibt eine Knotenüberdeckung für $G \circ \delta$.

$\delta = v_i^-$: Falls $v_i \in U$, dann ist auch $U \setminus \{v_i\}$ eine Knotenüberdeckung für $G \circ \delta$ der Größe $\leq k - 1$, da wir insbesondere davon ausgehen, dass auch zu $v_i$ inzidente Kanten gelöscht werden.

Sonst ist $U \setminus \{v_i\} = U$ und weil $G \circ \delta$ ebenfalls alle Kanten aus $G$ abzüglich der zu $v_i$ Inzidenten enthält, bleibt $U$ auch zu allen Kanten von $G \circ \delta$ inzident und damit eine Kontenüberdeckung der Größe $\leq k$.

Nun ist $\delta' = \delta \circ \omega$ für $\omega \in \Delta$. Nach Induktionsvorraussetzung hat $G \circ \delta$ eine Knotenüberdeckung mit höchstens $k + |\delta|$ Knoten, falls $G$ bereits eine solche Menge der Größe höchstens $k$ hat. Nun können wir aber die Argumentation aus dem Induktionsanfang nutzen und erhalten, dass auch $G \circ \delta \circ \omega$ eine Knotenüberdeckung mit höchstens $k + |\delta| + 1$ Knoten haben muss. □

Das vorhergehende Lemma zeigt uns, wie wir im Allgemeinen auf Änderungen in einem Graphen reagieren. Im Folgenden präsentieren wir nun eine inkrementelle Variante von Buss' Kernelisation und insbesondere des Algorithmus aus Satz 3.5.6.

**Satz 4.5.2** ([MM17, Algorithmus 1, Theorem 6])**.** incr-Vertex-Cover$(v^\pm, e^\pm)$ *ist lösbar in Zeit $\mathcal{O}(2^{k^2} \cdot |\delta|)$. Insbesondere hängt die Laufzeit* nur *vom Parameter ab.*

*Beweis.* Sei $k \in \mathbb{N}$ und $G = (V, E)$ ein Graph. Der Zeuge enthalte die folgenden Informationen.

- Eine Knotenüberdeckung $U$ für $G$ der Größe $\leq k$.

- Die Knoten von $G$ seien danach markiert, ob sie im Kernel enthalten sind oder wegen hohen Knotengrades oder wegen Grad 1 gelöscht wurden.

Mit $K$ bezeichnen wir die Menge der Knoten im Kernel und mit $V^{(>k)}$ die Menge der Knoten $v$, die aufgrund von $\deg(v) > k$ gelöscht wurden, absteigend nach dem Knotengrad zum Zeitpunkt der Löschung sortiert.

Wir betrachten zunächst Unterprozeduren, die die jeweiligen Klassen von Änderungen behandeln.

*Fall $e_{ij}^+$:* Wir wissen aus Lemma 4.5.1, dass wir in diesem Fall $(G \circ e_{ij}^+, k+1)$ betrachten müssen. Deshalb kann es passieren, dass wir einige Knoten $v$ nicht mehr aus $V^{(>k)}$ löschen können, und zwar gerade die $v$ mit $\deg(v) = k+1$. Wir wollen zunächst eine obere Schranke für die Knoten finden, die wir zusätzlich betrachten müssen.

Bezeichne also $V_\ell^{(>k)}$ die ersten $\ell$ Knoten aus $V^{(>k)}$. Da $V^{(>k)}$ absteigend nach Knotengrad geordnet ist, können wir $\ell$ so wählen, dass $V_\ell^{(>k)}$ gerade die Knoten $v$ aus $V^{(>k)}$ enthält, sodass $\deg(v) > k+1$.

Damit wissen wir, dass die Knoten aus $V^{(>k)} \setminus V_\ell^{(>k)}$ höchstens Grad $k+1-\ell$ haben, da sonst die Wahl von $\ell$ falsch gewesen wäre, und ein Knoten $u$ mit $\deg(u) > k+1-\ell$ ebenfalls gelöscht werden müsste.

Im schlimmsten Fall ist $\ell = 0$, woraus jedoch folgt, dass für alle $v \in V^{(>k)}$ gilt, dass $\deg(v) \leq k+1$. Wir haben in diesem größeren Kernel also höchstens $2(k^2+k)$ Knoten.

Folgende Unterprozedur fasst diese Überlegungen zusammen und passt die Informationen aus $S$ an.

**Prozedur** EdgeAdded($G = (V, E), k \in \mathbb{N}, S = (U, K, V^{(>k)}, V^{(0)})$)

1: $k' \leftarrow k+1$
2: $V_\ell^{(>k)} \leftarrow \{\, v \in V^{(>k)} \mid \deg(v) > k+1 \,\}$ ▷ $\mathcal{O}(k)$
3: $K \leftarrow K \cup N[V^{(>k)} \setminus V_\ell^{(>k)}]$ ▷ $\mathcal{O}(k^2)$
4: $V^{(>k)} \leftarrow V^{(>k)} \setminus K$, $V^{(0)} \leftarrow V^{(0)} \setminus K$ ▷ $\mathcal{O}(k^2)$
5: $G' \leftarrow (K, E \cap K \times K)$
6: $U \leftarrow$ Search-VC$(G', k', \emptyset)$ ▷ $\mathcal{O}(2^k \cdot k^2)$
7: $S' \leftarrow (U, K, V^{(>k)}, V^{(0)})$
8: **return** $(k', S')$

*Fall $e_{ij}^-$:* Es werden auch weiterhin alle Knoten aus $V^{(>k)}$ mit genügend hohem Knotengrad gelöscht. Falls o. B. d. A. $v_i \in V^{(>k)}$ und $\deg(v_i) = k$ nach dem Löschen der Kante $\{v_i, v_j\}$, dann wird $v_i$ zwar nicht mehr aufgrund hohen Knotengrades gelöscht, weshalb wir $v_i$ also nur aus $V^{(>k)}$ in $K$ verschieben müssen.

$U$ bleibt jedoch zunächst eine gültige Knotenüberdeckung, da nur eine Kante gelöscht wird und alle übrigen Kanten auch weiterhin inzident zu einem $u \in U$ bleiben. Daher

können wir den ursprünglichen Kernel weiternutzen.

**Prozedur** EdgeDeleted($G = (V, E), k \in \mathbb{N}, S = (U, K, V^{(>k)}, V^{(0)})$)

1: **if** $\deg(v_i) = k$ **then** ▷ $\mathcal{O}(k)$
2: $\quad K \leftarrow K \cup \{v_i\}$
3: $\quad V^{(>k)} \leftarrow V^{(>k)} \setminus \{v_i\}$
4: **if** $\deg(v_j) = k$ **then** ▷ $\mathcal{O}(k)$
5: $\quad K \leftarrow K \cup \{v_j\}$
6: $\quad V^{(>k)} \leftarrow V^{(>k)} \setminus \{v_j\}$
7: $S' \leftarrow (U, K, V^{(>k)}, V^{(0)})$
8: **return** $(k', S')$

*Fall* $v_i^+$: Da $v_i$ als isolierter Knoten hinzugefügt wird, müssen wir ihn nicht weiter beachten.

*Fall* $v_i^-$: In jedem Fall können wir $U$ als Knotenüberdeckung weiterverwenden. Falls $v_i \in V^{(>k)}$, dann verhält sich das Löschen von $v_i$ ähnlich wie in Lemma 3.5.1 ohne $k$ zu verringern.

Falls $v_i \notin V^{(>k)}$ und adjazent zu einem $u \in V^{(>k)}$ ist, dann wird durch das Löschen der Grad von $u$ verringert. Ist $\deg(u) > k + 1$, war also der Grad von $u$ vor dem Löschen hinreichend groß, dann wird $u$ auch weiterhin unter Anwendung der Regel aus Lemma 3.5.1 gelöscht.

Ist $\deg(u) = k + 1$, dann wird $u$ zwar nicht mehr aufgrund hohen Knotengrades gelöscht, weshalb wir $u$ also nur aus $V^{(>k)}$ in $K$ verschieben müssen. $U$ bleibt jedoch zunächst eine gültige Knotenüberdeckung.

**Prozedur** VertexDeleted($G = (V, E), k \in \mathbb{N}, i \in \mathbb{N}, S = (U, K, V^{(>k)}, V^{(0)})$)

1: $U \leftarrow U \setminus \{v_i\}$, $K \leftarrow K \setminus \{v_i\}$ ▷ $\mathcal{O}(k)$
2: $V_\ell^{(>k)} \leftarrow \{\, v \in V^{(>k)} \mid \deg(v) > k \,\}$ ▷ $\mathcal{O}(k^2)$
3: $K \leftarrow K \cup N[V^{(>k)} \setminus V_\ell^{(>k)}]$ ▷ $\mathcal{O}(k^2)$

Folgender Algorithmus löst incr-Vertex-Cover.

**Eingabe:** $G = (V, E), k \in \mathbb{N}, \delta \subsetneq \Delta, S = (U, K, V^{(>k)}, V^{(0)})$

1: Prüfe, ob $(G, k) \in$ p-Vertex-Cover für $U$
2: Falls nicht, **return nein**
3: **for** $\omega \in \delta$ **do** ▷ $\mathcal{O}(|\delta|)$
4: **switch** $\omega$ **do**
5: **case** $e_{ij}^{+}$: $(k, S) \leftarrow \textsc{EdgeAdded}(G, k, S)$ ▷ $\mathcal{O}(2^k \cdot k^2)$
6: **case** $e_{ij}^{-}$: $(k, S) \leftarrow \textsc{EdgeDeleted}(G, k, S)$ ▷ $\mathcal{O}(k^2)$
7: **case** $v_i^{-}$: $(k, S) \leftarrow \textsc{VertexDeleted}(G, k, S)$ ▷ $\mathcal{O}(k^2)$
8: **return ja**

Dabei beträgt die Laufzeit dieses Algorithmus

$$\mathcal{O}(|\delta|) \cdot \mathcal{O}(2^k \cdot k^2) \subseteq \mathcal{O}(|\delta| \cdot 2^k \cdot k^2) \subseteq \mathcal{O}(|\delta| \cdot 2^{k^2}),$$

womit die Aussage gezeigt ist. □

Abschließend fassen wir die Resultate dieses Kapitels zusammen. Dabei haben sich alle Vollständigkeitsresultate aus *fpt*-Reduktionen ergeben.

Wir stellen fest, dass die zusätzlichen Informationen aus dem Zeugen tatsächlich hilfreich sein können. So lassen sich viele Graphenprobleme einfacher lösen, wie in Tabelle 4.1 dargestellt. Eine Ausnahme bildet hierbei incr-Dominating-Set$(v^-)$, dessen W[2]-Vollständigkeit sich aus dem Zusammenhang zwischen p-Dominating-Set und MIN-WSAT$(\Gamma_{2,1}^{+})$ ergibt.

Für die inkrementellen Varianten von Hypergraph d-Clique und Hypergraph d-Independent-Set haben wir nur das Löschen von Knoten als Folgerung aus der Schwierigkeit von incr-MAX-WSAT$(\Gamma_{1,d}^{-})$ betrachtet, da es in der betreffenden Literatur ebenfalls keine Resultate dazu gab [MM17].

| | Erlaubte Änderungen | | | |
|---|---|---|---|---|
| | $v^+$ | $v^-$ | $e^+$ | $e^-$ |
| incr-Clique | $\in$ FPT[4.1.2] | $\in$ FPT[4.1.2] | $\in$ FPT[4.1.2] | $\in$ FPT[4.1.2] |
| incr-Hypergraph d-Clique | — | W[1]-schwer[4.3.5] | — | — |
| incr-Independent-Set | $\in$ FPT[4.2.1] | $\in$ FPT[4.2.1] | $\in$ FPT[4.2.1] | $\in$ FPT[4.2.1] |
| incr-Hypergraph d-Independent-Set | — | W[1]-schwer[4.3.4] | — | — |
| incr-Dominating-Set | $\in$ FPT[4.4.2] | W[2]-vollständig[4.4.2] | $\in$ FPT[4.4.2] | $\in$ FPT[4.4.2] |
| incr-Vertex-Cover | $\in$ FPT[4.5.2] | $\in$ FPT[4.5.2] | $\in$ FPT[4.5.2] | $\in$ FPT[4.5.2] |

Tabelle (4.1): Auswirkungen der erlaubten Änderungen auf die Schwierigkeit inkrementeller Graphenprobleme. Die jeweiligen Resultate werden im oberen Index referenziert.

Für inkrementelle Erfüllbarkeitsprobleme stellen wir jedoch fest, dass nur das Löschen von Klauseln unabhängig von der Formelklasse einfach bleibt. Sobald wir das Löschen

von Variablen erlauben, wird das jeweilige inkrementelle Problem genauso schwer wie seine statische Variante. Die Resultate hierzu sind in Tabelle 4.2 zusammengefasst. Für incr-WSAT(CIRC) wurde dabei $cl^-$ nicht betrachtet, da das Löschen von Klauseln im Kontext von Boole'schen Schaltkreisen nicht sinnvoll ist.

| | Erlaubte Änderungen | |
|---|---|---|
| | $var^-$ | $cl^-$ |
| incr-MAX-WSAT($\Gamma^-_{1,2}$) | $\in$ FPT[4.3.1] | $\in$ FPT[4.3.2] |
| incr-MAX-WSAT($\Gamma^-_{1,d}$) | W[1]-vollständig[4.3.3] | $\in$ FPT[4.3.2] |
| incr-MIN-WSAT($\Gamma^+_{2,1}$) | W[2]-vollständig[4.3.6] | $\in$ FPT[4.3.2] |
| incr-MAX-WSAT($\Gamma^-_{t,d}$) | W[t]-vollständig[4.3.7] | $\in$ FPT[4.3.2] |
| incr-MIN-WSAT($\Gamma^+_{t,1}$) | W[t]-vollständig[4.3.7] | $\in$ FPT[4.3.2] |
| incr-WSAT(CIRC) | W[P]-vollständig[4.3.8] | — |

Tabelle (4.2): Auswirkungen der erlaubten Änderungen auf die Schwierigkeit inkrementeller Erfüllbarkeitsprobleme. Die jeweiligen Resultate werden im oberen Index referenziert.

# 5 Ausblick

Wie bereits in Kapitel 1 angesprochen, ist die inkrementelle Komplexitätstheorie zwar ein relativ junges Gebiet, bietet dafür jedoch bereits eine Fülle an Resultaten. Wir umreißen einige dieser Resultate, um einen Ansatzpunkt für zukünftige Forschung zu geben.

## 5.1 Inkrementelle Komplexitätsklassen nach Miltersen et al.

Miltersen et al. [Mil+94] führen *inkrementelle Komplexitätsklassen* auf Grundlage von *Random Access Machines (RAMs)* ein, also Maschinen, denen eine unbegrenzte Anzahl von Registern zur Verfügung stehen und die für den Zugriff auf diese Register konstante Zeit benötigen [BBJ07, Abschnitt 5.1].

Im Folgenden bringen wir exemplarisch die Definition für inkrementellen Zeitbedarf aus dem Aufsatz von Miltersen et al. [Mil+94]. Dabei haben wir die Benennung der einzelnen Komponenten an die Terminologie unserer inkrementellen Probleme aus Definition 2.3.3 angepasst.

**Definition 5.1.1** ([Mil+94, Definition 2.1])**.** Eine Sprache $L$ liegt in $\mathsf{incr\text{-}TIME}(f(n))$, falls zwei RAM-Programme $P_1$ und $P_2$ existieren, so dass die folgenden Bedingungen erfüllt sind.

1. Für eine initiale Instanz $x_0$ berechnet $P_1$ eine interne Datenstruktur $S_{x_0}$.
2. Für die momentane Instanz $x$, einer Menge von Änderungen $\delta$ an $x$ und der zugehörigen Datenstruktur $S_x$ aus der vorherigen Berechnung berechnet $P_2$ in Laufzeit $\mathcal{O}(|\delta| \cdot f(|x|))$, ob $x \circ \delta \overset{?}{\in} L$, und die zugehörige interne Datenstruktur $S_{x\circ\delta}$ für die neue Instanz.

Insbesondere wird deutlich, dass die Informationen aus einer vorherigen Berechung stammen müssen und entsprechend auch durch einen nicht notwendig inkrementellen Algorithmus berechnet wurden. Für die inkrementelle Betrachtung ist dann nur die Komplexität der tatsächlich inkrementellen Berechnung von Interesse, da nur an $P_2$ Bedingungen bezüglich der Laufzeit gestellt werden.

In ihrer Arbeit kommen Miltersen et al. [Mil+94] zu dem Schluss, dass Probleme mit geringem statischen Platzbedarf auch einen geringen inkrementellen Zeitbedarf haben [Mil+94, Theorem 2.4].

Außerdem führen sie einen inkrementellen Reduktionsbegriff ein und zeigen damit, dass die bekannten [GHR91; MSS90] P-vollständigen Probleme auch für die verwendeten inkrementellen Reduktionen P-vollständig sind [Mil+94, Theorem 3.2].

## 5.2 Inkrementelle Algorithmen für Voronoi-Diagramme

Für einen metrischen Raum $R$ und eine Menge von Objekten $O$ bezeichnen wir mit einem *Voronoi-Diagramm (VD)* im Wesentlichen eine Unterteilung des Raumes in Regionen, sodass jede Region aus allen Punkten des Raumes besteht, die zu einem Objekt $x \in O$ den geringsten Abstand haben [Kle05, Abschnitt 5.2].

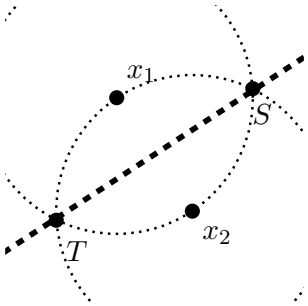
Für die Berechnung von VD ist der inkrementelle Ansatz laut Klein [Kle05, Abschnitt 6.2] eine der ältesten Methoden. Tatsächlich beschreiben Horton um 1917 [Hor17] und Kopec um 1963 [Kop63] inkrementelle Verfahren für die Ermittelung eines VD in der Ebene, die noch auf die Konstruktion mit Zirkel und Lineal ausgelegt sind. Das Verfahren sei im Folgenden grob skizziert.

Es sei eine Menge von Punkten $\{ x_1, \ldots, x_n \}$ in der Ebene $\mathbb{R}^2$ gegeben. Mit dem Abstand $|x - y|$ zwischen zwei Punkten $x$ und $y$ bezeichnen wir die euklidische Distanz.

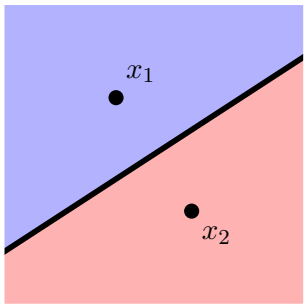
1. Wir ermitteln zunächst den *Bisektor* $B_{1,2}$ von $x_1$ und $x_2$, also die Menge der Punkte, die sowohl zu $x_1$ als auch $x_2$ denselben Abstand haben. Im Falle der euklidischen Distanz ist dies gerade die Mittelsenkrechte der Strecke zwischen $x_1$ nd $x_2$.

   Man erhält den Bisektor wie in Abbildung 5.1a, indem man jeweils um $x_1$ und $x_2$ einen Kreis mit Radius $|x_1 - x_2|$ zieht. Die Gerade, die durch die Schnittpunkte $S$ und $T$ verläuft, ist dann der Bisektor.

   Wir sehen, dass die gesamte Ebene nun in zwei Regionen $X_1$ und $X_2$ unterteilt wird, die jeweils die Punkte enthalten, die näher an $x_1$ respektive $x_2$ liegen.



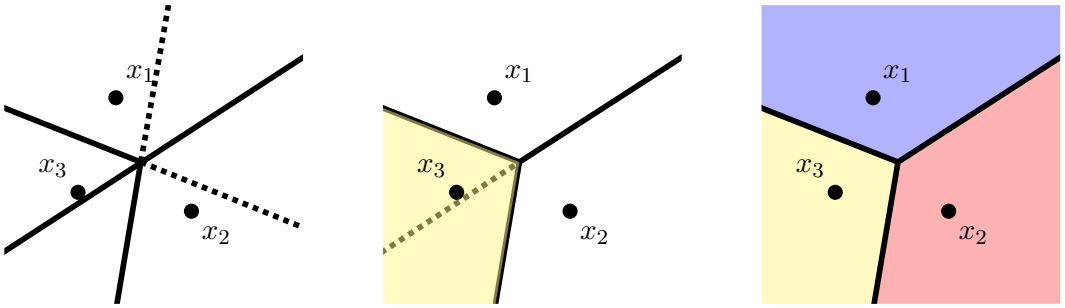

(a) Konstruktion des Bisektors




(b) Entstehende Unterteilung

Abbildung (5.1): Voronoi-Diagramm für zwei Punkte in der Ebene

2. Um nun den Punkt $x_i$ unserem Diagramm hinzuzufügen, ermitteln wir zunächst alle Bisektoren $B_{k,i}$ mit $1 \leq k < i$ nach dem Vorgehen aus Schritt 1. Um nun die entsprechenden Begrenzungen der Region $X_i$ zu erhalten, wählen wir die Abschnitte der Bisektoren $B_{k,i}$, die im Gebiet $X_k$ liegen. In Abbildung 5.2a sind diese Abschnitte der $B_{k,3}$ mit durchgezogenen Linien dargestellt. Sobald wir die Grenzen der Region $X_i$ ermittelt haben, müssen wir nur noch die nicht mehr benötigten Begrenzungen des vorherigen Vornoi-Diagrammes entfernen, die in der Region $X_i$ liegen, wie in Abbildung 5.2b.




(a) Bisektoren $B_{1,3}$ und $B_{2,3}$

(b) Ermittlung der Region $X_i$

(c) Entstehende Unterteilung

Abbildung (5.2): Hinzufügen eines dritten Punktes zum Diagramm aus Abbildung 5.1

Held und Huber [HH09] bauen auf vorangegangener Arbeit auf und stellen einen Algorithmus vor, der durch *zufälliges inkrementelles Einfügen* ein solches VD berechnet und insbesondere auch in der Lage ist, Kreisbögen als Objekte zu verarbeiten.

Ähnlich der zuvor beschriebenen Zirkel- und Linealkonstrukton ist dieser Algorithmus dadurch auch in der Lage, auf nachträgliche Änderungen der Eingabe, also an der Menge der Objekte $O$, zu reagieren, ist also ebenfalls ein inkrementeller Algorithmus im Sinne dieser Arbeit.

Für die Laufzeit ermitteln Held und Huber [HH09, Abschnitt 4] experimentell eine asymptotische Laufzeit von $\mathcal{O}(n \log n)$.

Man sieht, dass die Berechnung von VD in gewisser Weise ein natürliches inkrementelles Problem ist und in dieser Richtung bereits betrachtet wurde. Darauf aufbauend kann man dieses Problem auf typische Eigenschaften leicht lösbarer inkrementeller Probleme untersuchen.

## 5.3 Dynamische Komplexität

Der Begriff der dynamischen Komplexität stammt aus der deskriptiven Komplexitätstheorie. Dort werden Komplexitätsklassen anhand der Ausdrucksstärke der Logik charakterisiert, mit der die Probleme der entsprechenden Klasse formuliert werden können.

Das erste große Resultat hierzu ist der *Satz von Fagin*, der die Klasse NP mit SO∃ in Verbindung setzt [Imm99, Theorem 7.8]. Das bedeutet also, dass alle Probleme in NP durch Formeln der existenziellen Prädikatenlogik zweiter Stufe, d. h. prädikatenlogische Formeln mit einem Existenzquantor über Prädikate aus Sätzen der Prädikatenlogik erster Stufe [Imm99, Abschnitt 7.1] ausgedrückt werden können.

Dabei geht Immerman ebenfalls auf die Beobachtung ein, dass viele komplexitätstheoretische Probleme in der Praxis dynamischer Natur sind und führt daher im Rahmen der deskriptiven Theorie *dynamische Komplexitätsklassen* Dyn-$\mathcal{C}$ ein [PI97]. Der Fokus verschiebt sich also von der Frage, wie komplex ein Problem zu lösen ist, nachdem die gesamte Eingabe gelesen wurde, hin zu der Frage, welche Hilfsinformationen

gespeichert werden müssen, um Anfragen zur geänderten Eingabe möglichst schnell zu beantworten.

Die wesentliche Idee besteht darin, dass wir zu einem Problem $L$ zunächst mit einer initialen Struktur $\mathbf{A_0}$ beginnen. Dann erhalten wir eine Folge von Änderungsanfragen $\omega_1, \ldots, \omega_t$ und definieren induktiv $\mathbf{A_{i+1}} := \mathbf{A_i} \circ \omega_i$. Die jeweiligen $\mathbf{A_i}$ sind also mit der Eingabe $x$ und die $\omega_1, \ldots, \omega_t$ mit den Änderungen der inkrementellen Probleme vergleichbar.

Um nun die Frage zu beantworten, ob $\mathbf{A_i} \overset{?}{\in} L$, unterhalten wir eine dazugehörige Datenstruktur $\mathbf{B_i}$, wobei wir verlangen, dass

1. die initiale Struktur $\mathbf{B_0}$ effizient aus $\mathbf{A_0}$ berechnet werden kann
2. die $\mathbf{B_i}$ höchstens polynomiell größer sind als die jeweiligen $\mathbf{A_i}$
3. $\mathbf{B_{i+1}}$ effizient aus $\mathbf{B_i}$ und $\omega_i$ berechnet werden kann und
4. mithilfe von $\mathbf{B_i}$ effizient überprüft werden kann, ob $\mathbf{A_i} \overset{?}{\in} L$.

Die $\mathbf{B_i}$ entsprechen also dem Zeugen $S$ aus den inkrementellen Problemen.

Ähnlich zu den Resultaten bezüglich Independent-Set (Lemma 3.2.1 und Korollar 4.2.1) und Clique (Lemma 3.3.1 und Satz 4.1.2) in dieser Arbeit wurde nun gezeigt, dass viele Probleme im statischen Fall *nicht* mit Prädiktatenlogik erster Stufe (FO) ausgedrückt werden können, dies im dynamischen Fall jedoch durchaus möglich ist [PI97].

So ist es beispielsweise nicht möglich, das Problem Parity, also die Frage, ob ein Binärwort eine ungerade Anzahl von Einsen enthält, in FO auszudrücken [Imm99, Theorem 13.1]. Wird das Binärwort jedoch Bit für Bit konstruiert und speichert man mit einem zusätzlichen Bit die Päritat des momentanen Wortes, so lässt sich Parity mit FO-Formeln ausdrücken, liegt also in der Klasse Dyn-FO [Imm99, Proposition 14.16].

Das ist deshalb von Interesse, da wichtige Klassen der Schaltkreiskomplexität mit Varianten von FO charakterisiert wurden. Entsprechende Resultate hierzu finden sich im Buch von Immerman [Imm99, Theorem 5.22] und im Buch von Vollmer [Vol99, Kapitel 4.5.4]. Es bietet sich daher an, zunächst inkrementelle Probleme mit dynamischer Komplexität in Verbindung zu setzen.

Dann kann man Komplexitätsklassen der Schaltkreistheorie mit dynamischen Komplexitätsklassen charakterisieren, um damit Ansätze für inkrementelle Berechnung mit Schaltkreisen zu entwickeln, da dem üblichen Berechnungsmodell der Schaltkreise ebenfalls inhärent ist, dass die gesamte Eingabe vorliegt und einmalig verarbeitet wird.

## 5.4 Rekonfigurationsprobleme

Bei einem *Rekonfigurationsproblem* wird zu einer Instanz $x$ und zwei Lösungen $a$, $b$ die Frage gestellt, ob $a$ durch Änderungen in $b$ überführt werden kann. Einen Überblick hierzu findet sich bei van den Heuvel [Heu13].

Ein grundlegender Ansatz für Rekonfigurationsprobleme ist die Definition des sogenannten *Rekonfigurationsgraphen* zu einer gegebenen Instanz. Dieser Graph enthält alle möglichen Lösungen der Instanz als Knoten. Zwei Lösungen sind adjazent, falls sie durch eine einzelne Änderung ineinander überführt werden können.

Wir suchen hier also für zwei Lösungen $a$ und $b$ die Menge der Änderungen $\delta$, während bei inkrementellen Problemen $\delta$ vorgegeben ist und überprüft wird, ob die veränderte Instanz auch weiterhin eine Ja-Instanz ist. Außerdem wird der Zeuge $S$, der meistens eine Lösung für die ursprüngliche Instanz ist, in einen Zeugen für die veränderte Instanz überführt.

Die Beziehung zwischen Rekonfigurationsproblemen und inkrementellen Problemen scheint zunächst ähnlich wie die Beziehung zwischen polynomieller Lösbarkeit und Überprüfbarkeit, also zwischen $\mathsf{P}$ und $\mathsf{NP}$. Es stellt sich also die naheliegende Frage, welche Konsequenzen die Komplexität von Rekonfigurationsproblemen für inkrementelle Probleme hat und ob ein effizient lösbares Rekonfigurationsproblem auch ein effizient lösbares inkrementelles Problem impliziert.

# Abkürzungsverzeichnis

***DNF*** Disjunktive Normalform

***FPT*** fixed-parameter tractable

***KNF*** Konjunktive Normalform

***RAM*** Random Access Machine

***SAT*** Erfüllbarkeitsproblem (*Satisfiability*)

***TM*** Turingmaschine

***VD*** Voronoi-Diagramm

***WSAT*** Gewichtetes Erfüllbarkeitsproblem (*Weighted satisfiability*)

# Literatur


[BBJ07] George S. Boolos, John P. Burgess und Richard C. Jeffrey. 5. Aufl. Cambridge University Press, 2007. DOI: 10.1017/CBO9780511804076.006.

[Bie+09] A. Biere u. a. *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. NLD: IOS Press, 2009. ISBN: 1586039296.

[CKX06] Jianer Chen, Iyad A. Kanj und Ge Xia. „Improved Parameterized Upper Bounds for Vertex Cover". In: *Mathematical Foundations of Computer Science 2006*. Hrsg. von Rastislav Královič und Paweł Urzyczyn. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 238–249. ISBN: 978-3-540-37793-1.

[Cre+19] Nadia Creignou u. a. „Parameterised Enumeration for Modification Problems". In: *Algorithms* 12.9 (Sep. 2019), S. 189. ISSN: 1999-4893. DOI: 10.3390/a12090189.

[Des+05] Prasanna Desikan u. a. „Incremental Page Rank Computation on Evolving Graphs". In: *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*. WWW '05. Chiba, Japan: Association for Computing Machinery, 2005, S. 1094–1095. ISBN: 1595930515. DOI: 10.1145/1062745.1062885.

[DF13] Rodney G. Downey und Michael R. Fellows. *Fundamentals of Parameterized Complexity*. London: Springer London, 2013. ISBN: 978-1-4471-5559-1. DOI: 10.1007/978-1-4471-5559-1.

[DF99] Rodney G. Downey und Michael R. Fellows. *Parameterized Complexity*. New York, NY: Springer New York, 1999. ISBN: 978-1-4612-0515-9. DOI: 10.1007/978-1-4612-0515-9.

[FG06] Jörg Flum und Martin Grohe. *Parameterized Complexity Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. ISBN: 978-3-540-29953-0. DOI: 10.1007/3-540-29953-X.

[GHR91] Raymond Greenlaw, H James Hoovery und Walter L Ruzzoz. *A Compendium of Problems Complete for P*. Techn. Ber. 1991.

[Gol06] Oded Goldreich. „On Promise Problems: A Survey". In: *Theoretical Computer Science: Essays in Memory of Shimon Even*. Hrsg. von Oded Goldreich, Arnold L. Rosenberg und Alan L. Selman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 254–290. ISBN: 978-3-540-32881-0. DOI: 10.1007/11685654_12.

[Heu13] Jan van den Heuvel. „The complexity of change“. In: *Surveys in Combinatorics 2013*. Hrsg. von Simon R. Blackburn, Stefanie Gerke und MarkEditors Wildon. London Mathematical Society Lecture Note Series. Cambridge University Press, 2013, S. 127–160. DOI: 10.1017/CBO9781139506748.005.

[HH09] Martin Held und Stefan Huber. „Topology-oriented incremental computation of Voronoi diagrams of circular arcs and straight-line segments“. In: *Computer-Aided Design* 41.5 (2009). Voronoi Diagrams and their Applications, S. 327–338. ISSN: 0010-4485. DOI: 10.1016/j.cad.2008.08.004.

[Hor17] Robert Elmer Horton. „Rational study of rainfall data makes possible better estimates of water yield“. In: *Eng. News-Record* (1917), S. 211–213.

[Imm99] Neil Immerman. *Descriptive Complexity*. Springer New York, 1999. DOI: 10.1007/978-1-4612-0539-5.

[Kle05] Rolf Klein. *Algorithmische Geometrie: Grundlagen, Methoden, Anwendungen*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. ISBN: 978-3-540-27619-7. DOI: 10.1007/3-540-27619-X.

[Kop63] Richard J. Kopec. „AN ALTERNATIVE METHOD FOR THE CONSTRUCTION OF THIESSEN POLYGONS“. In: *The Professional Geographer* 15.5 (1963), S. 24–26. DOI: 10.1111/j.0033-0124.1963.024_r.x.

[Mil+94] Peter Bro Miltersen u. a. „Complexity models for incremental computation“. In: *Theoretical Computer Science* 130.1 (1994), S. 203–236. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)90159-7.

[MM17] Bernard Mans und Luke Mathieson. „Incremental problems in the parameterized complexity setting“. In: *Theory of Computing Systems* 60.1 (2017), S. 3–19.

[MSS90] Satoru Miyano, Shuji Shiraishi und Takayoshi Shoudai. „A list of p-complete problems“. In: *RIFIS technical report* 17 (1990).

[Pag+99] Lawrence Page u. a. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report 1999-66. Stanford InfoLab, Nov. 1999.

[PI97] Sushant Patnaik und Neil Immerman. „Dyn-FO: A Parallel, Dynamic Complexity Class“. In: *Journal of Computer and System Sciences* 55.2 (1997), S. 199–209. ISSN: 0022-0000. DOI: 10.1006/jcss.1997.1520.

[Ram96] G. Ramalingam, Hrsg. *Bounded Incremental Computation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996. ISBN: 978-3-540-68458-9. DOI: 10.1007/BFb0028290.

[RR93] G. Ramalingam und Thomas Reps. „A Categorized Bibliography on Incremental Computation“. In: *Proceedings of the 20th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '93. Charleston, South Carolina, USA: Association for Computing Machinery, 1993, S. 502–510. ISBN: 0897915607. DOI: 10.1145/158511.158710.

[Sip06] Michael Sipser. *Introduction to the Theory of Computation*. Boston, MA: International Thomson Publishing, 2006. ISBN: 9780619217648.

[Sto73] Larry Stockmeyer. „Planar 3-Colorability is Polynomial Complete“. In: *SIGACT News* 5.3 (Juli 1973), S. 19–25. ISSN: 0163-5700. DOI: 10.1145/1008293.1008294.

[Vol99] Heribert Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999. ISBN: 978-3-662-03927-4. DOI: 10.1007/978-3-662-03927-4.