25. April 2012

Diplomarbeit (Überarbeitung)

# Das Dominating Induced Matching Problem für azyklische Hypergraphen

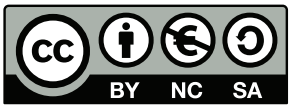
**Arne Leitert**
Diplom Informatik (6201646)
arne.leitert@uni-rostock.de

Prof. Dr. Andreas Brandstädt
Gutachter und Betreuer

Prof. Dr. Karsten Wolf
Zweitgutachter

**Lehrstuhl für Theoretische Informatik**
Universität Rostock
Fakultät für Informatik und Elektrotechnik

Universität Rostock Traditio et Innovatio



## Zusammenfassung

Diese Arbeit beschäftigt sich mit dem Dominating Induced Matching Problem für azyklische Hypergraphen. Dabei wird nach einer Kantenmenge gesucht, so dass jede Kante mit genau einer Kante aus dieser Menge einen gemeinsamen Knoten besitzt. Nach der Behandlung der theoretischen Grundlagen wird gezeigt, dass sich das Problem auf das Weighted Independent Set Problem für das Quadrat des Linegraphen reduzieren lässt. Am Ende der Arbeit werden dann eingie Algorithmen vorgestellt, um das Problem für verschiedene Klassen von azyklischen Hypergraphen in Polynomialzeit zu lösen.

## Abstract

This thesis deals with the Dominating Induced Matching problem for acyclic hypergraphs. This problem asks for a set of edges so that every edge shares a vertex with exactly one edge of this set. After dealing with the theoretical background this thesis will show that the problem can be reduced to the Weighted Independent Set problem on the square of the linegraph. At the end of the thesis some algorithms will be presentetd to solve the problem for several kinds of acyclic hypergraphs in polynomial time.

# Inhaltsverzeichnis

Kapitel 1

# Einleitung

Das *Dominating Induced Matching Problem* sucht nach einer Teilmenge der Kanten eines Graphen, so dass jede Kante des Graphen entweder in dieser Teilmenge ist oder mit genau einer solchen Kante benachbart ist. Es wird auch als *Efficient Edge Domination Problem* bezeichnet und ist im Allgemeinen NP-vollständig [29].

Die Besonderheit dieses Problems ist, dass es sowohl ein Pack- als auch ein Abdeckungsproblem darstellt. Es müssen genug Kanten gewählt werden, damit alle Kanten, die nicht zum Matching gehören mit einer Kante des Matchings verbunden sind. Zwischen den Kanten des Matchings müssen sich aber gleichzeitig immer mindestens zwei weitere Kanten befinden.

Das Problem wurde für einfache Graphen bereits umfangreich studiert (siehe beispielsweise [9, 11, 12, 35, 36]). Diese Arbeit untersucht das Problem nun für Hypergraphen. Dabei stehen azyklische Hypergraphen im Vordergrund.

Diese Arbeit ist dafür in sechs Kapitel unterteilt. Nach der Einleitung befassen sich das zweite und dritte Kapitel mit Graphen und Hypergraphen. Dabei werden verschiedene Klassen vorgestellt und miteinander in Verbindung gebracht. Das Thema des vierten Kapitels sind dominierende Knoten- und Kantenmengen sowie die dazugehörigen Probleme (zu denen auch das Dominating Induced Matching Problem gehört). Basierend auf den Ergebnissen der vorherigen Kapitel zeigt dann das fünfte Kapitel algorithmische Ansätze zur Lösung des Problems. Abschließend werden im letzten Kapitel die Ergebnisse der Arbeit noch einmal zusammengefasst.

Kapitel 2

# Graphen

Dieses Kapitel befasst sich mit den Grundlagen von Graphen. Es werden die in dieser Arbeit verwendeten Klassen definiert und in Verbindung gebracht.

## 2.1 Grundlagen

Dieser Abschnitt enthält einige grundlegende Definitionen.

**Definition 2.1 (Graph)**
Ein Graph $G$ ist ein 2-Tupel $G = (V, E)$. Dabei ist $V$ eine endliche Menge von Knoten und $E \subseteq \{\{u, v\} \mid u, v \in V \wedge u \neq v\}$ die Menge der Kanten.

Eine Kante $\{u, v\}$ wird abgekürzt durch $uv$.

**Definition 2.2 (Teilgraph)**
Es sei $T = (V_t, E_t)$ ein Graph. $T$ ist *Teilgraph* des Graphen $G = (V, E)$ genau dann, wenn $V_t \subseteq V$ und $E_t \subseteq E$. Gilt für alle $u, v \in V_t$ zusätzlich $uv \in E \Rightarrow uv \in E_t$, dann ist $T$ ein *induzierter Teilgraph*.

Falls $T$ ein induzierter Teilgraph von $G$ ist, so wird $T$ auch als $G[V_t]$ bezeichnet.

**Definition 2.3**
Es seien $G$ und $F$ Graphen. $G$ ist *$F$-frei* genau dann, wenn $F$ kein induzierter Teilgraph von $G$ ist.

**Definition 2.4 (Nachbarschaft eines Knotens)**
Gegeben sei ein Graph $G = (V, E)$. Dann seien $N(v) := \{u \mid uv \in E\}$ die *offene* und $N[v] := N(v) \cup \{v\}$ die *abgeschlossene Nachbarschaft* von $v$.

## 2.2 Graphisomorphie

Graphisomorphie bezeichnet vereinfacht gesagt, dass zwei Graphen gleich sind. Dies bedeutet bildlich gesprochen, dass man beide Graphen übereinander legen kann.

**Definition 2.5 (Graphisomorphie)**
Zwei Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ heißen *isomorph* genau dann, wenn eine bijektive Abbildung $\varphi : V_1 \to V_2$ existiert, so dass für alle $u, v \in V_1$ gilt:

$$uv \in E_1 \Leftrightarrow \varphi(u)\varphi(v) \in E_2$$

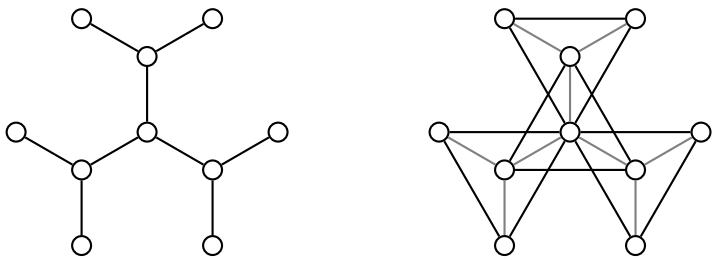
Die Isomorphie zweier Graphen wird wie folgt dargestellt: $G_1 \sim G_2$

## 2.3 Potenz eines Graphen

Bei der Potenzierung eines Graphen wird dessen Kantendichte erhöht. Dazu wird für $G^i$ eine Kante zwischen zwei Knoten $u$ und $v$ eingefügt, wenn der Abstand $\delta$ von $u$ und $v$ in $G$ kleiner oder gleich $i$ ist.

**Definition 2.6 (Potenz eines Graphen)**
Es seien $G = (V, E)$ ein Graph und $\delta$ eine Funktion, die den Abstand zweier Knoten zueinander angibt. Dann sei $G^i = (V, E^i)$ wie folgt definiert:

$$E^i := \{uv \mid u, v \in V;\ u \neq v;\ \delta(u, v) \leq i\}$$



**(a)** **(b)**

**Abbildung 2.1:** Beispiel für die Potenz eines Graphen: der Graph (a) und dessen Quadrat (b).

## 2.4 Spezielle Graphen

Bei den drei in diesem Abschnitt definierten Graphen handelt sich um individuelle Graphen. Die Abbildung 2.2 stellt alle drei dar.

**Definition 2.7 (House)**
Ein *House* ist ein Graph $(V, E)$, der wie folgt definiert ist (Index Arithmetik modulo k):

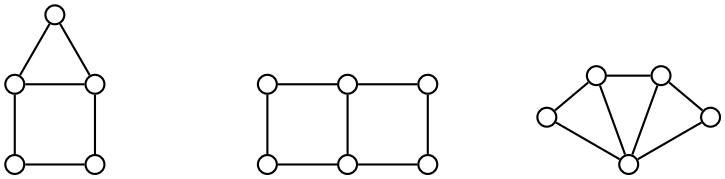
$$V = \{u, v_0, \ldots, v_3\}$$
$$E = \{v_i v_{i+1} \mid 0 \leq i \leq 3\} \cup \{uv_1, uv_2\}$$

**Definition 2.8 (Domino)**
Ein *Domino* ist ein Graph $(V, E)$, der wie folgt definiert ist:

$$V = \{u_i, v_i \mid 1 \leq i \leq 3\}$$
$$E = \{v_i v_{i+1} \mid 1 \leq i \leq 2\} \cup \{u_i v_i \mid 1 \leq i \leq 3\}$$

**Definition 2.9 (Gem)**
Ein *Gem* ist ein Graph $(V, E)$, der wie folgt definiert ist:

$$V = \{u, v_1, \ldots, v_4\} \text{ mit}$$
$$E = \{v_i v_{i+1} \mid 1 \leq i \leq 3\} \cup \{uv_i \mid 1 \leq i \leq 4\}$$



**(a)** House **(b)** Domino **(c)** Gem

**Abbildung 2.2:** Die Graphen House, Domino und Gem.

## 2.5 Einfache Klassen

Die Klassen in diesem Abschnitt sind sehr einfache Klassen. Graphen innerhalb dieser Klassen unterscheiden sich lediglich durch ihre Größe

voneinander. Die Abbildungen 2.3 und 2.4 zeigen für jede Klasse einen Beispielgraphen der Größe 5.

**Definition 2.10 (Kreis, $C_k$)**
Ein *sehnenloser Kreis* (engl. chordless cycle) der Länge $k$ ($k \geq 3$) ist ein Graph mit $k$ Knoten $v_1, \ldots, v_k$ und den Kanten $v_i v_{i+1}, 1 \leq i \leq k$ und $v_k v_1$.

Kreise der Länge $k \geq 5$ werden auch als *Hole* bezeichnet.

**Definition 2.11 (Clique, $K_i$)**
Ein Graph $G = (V, E)$ ist eine Clique der Größe $i$ genau dann, wenn $|V| = i$ und $E = \{uv \mid u, v \in V \wedge u \neq v\}$.

Man bezeichnet Cliquen auch als vollständige Graphen. Das Ermitteln einer größten Clique in einem Graphen (also eines größten Teilgraphen, der eine Clique darstellt) ist NP-vollständig [33].

**(a)** Ein Kreis der Länge 5 ($C_5$)

**(b)** Eine Clique der Größe 5 ($K_5$)

**Abbildung 2.3:** Die Graphen $C_5$ und $K_5$.

**Definition 2.12 (Sun, $S_k$)**
Gegeben seien die Knoten $V_u = \{u_0, \ldots, u_{k-1}\}$ und $V_v = \{v_0, \ldots, v_{k-1}\}$. Eine *complete Sun* $S_k = (V_s, E_s)$ der Größe $k$ ($k \geq 3$) sei dann wie folgt definiert (Index Arithmetik modulo $k$):

$$V_s := V_u \cup V_v$$
$$E_s := \{u_i v_i, u_i v_{i+1}, v_i v_j \mid 0 \leq i, j < k;\ i \neq j\}$$

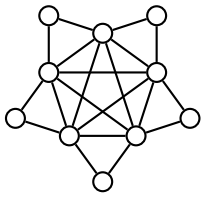


**Abbildung 2.4:** Der Graph $S_5$

## 2.6 Chordale Graphen

Unter chordalen Graphen versteht man Graphen, die keine Kreise der Größe 4 oder größer als induzierte Teilgraphen besitzen. Man bezeichnet sie in der (engl.) Literatur auch als *triangulated graphs*, *rigid-circuit graphs*, *monotone transitive graphs* und *perfect elemination graphs* [10].

**Definition 2.13 (chordaler Graph)**
Ein Graph $G$ ist *chordal* genau dann, wenn er $C_k$-frei ($k \geq 4$) ist.

Chordale Graphen besitzen einige hilfreiche Eigenschaften. Beispielsweise lassen sich einige NP-vollständige Probleme bei chordalen Graphen in Polynomial- oder gar Linearzeit lösen.

### 2.6.1 Perfekte Eliminationsordnungen

Bei einer *perfekten Eliminationsordnung* handelt es sich um eine spezielle Knotenreihenfolge. Dazu ist es jedoch nötig, *simpliziale Knoten* zu definieren.

**Definition 2.14 (simplizialer Knoten)**
Ein Knoten $v$ im Graphen $G$ ist *simplizial* genau dann, wenn $N(v)$ eine Clique in $G$ ist.

Jeder nichtleere, chordale Graph $G$ besitzt einen simplizialen Knoten; ist $G$ keine Clique, so gibt es sogar zwei simpliziale Knoten in $G$, die nicht miteinander verbunden sind [18]. Es ist somit möglich, die Knoten eines chordalen Graphen schrittweise zu eliminieren, indem man vom verbleibenden Graphen einen simplizialen Knoten entfernt. Diese Knotenreihenfolge wird als *perfekte Eliminationsordnung* bezeichnet.

**Definition 2.15 (perfekte Eliminationsordnung)**
Gegeben sei ein Graph $G = (V, E)$ mit $|V| = n$. Eine Folge $(v_1, \ldots, v_n)$ von Knoten ist eine *perfekte Eliminationsordnung* für $G$ genau dann, wenn für alle $i \in \{1, \ldots, n\}$ gilt: $v_i$ ist simplizial in $G[\{v_i, \ldots, v_n\}]$.

**Satz 2.1** [18] Ein Graph $G$ ist chordal genau dann, wenn $G$ eine perfekte Eliminationsordnung besitzt.

Es ist in Linearzeit möglich, die Chordalität eines Graph zu erkennen und eine perfekte Eliminationsordnung zu bilden, falls er chordal ist [39].

### 2.6.2 Strongly chordale Graphen

Eine (echte) Teilmenge der chordalen Graphen sind die strongly chordalen Graphen. Es gibt verschiedene (äquivalente) Definitionen für sie. Eine davon ist über das Verbot von Suns als induzierte Teilgraphen.

**Definition 2.16 (strongly chordaler Graph)**
Ein Graph $G$ ist *strongly chordal* genau dann, wenn $G$ chordal und $S_k$-frei ($k \geq 3$) ist.

Strongly chordale Graphen besitzen als chordale Graphen eine perfekte Eliminationsordnung. Es gibt allerdings auch andere Eliminationsordnungen für strongly chordale Graphen. Eine davon wird als *strong perfect elemination ordering* bezeichnet.

**Definition 2.17 (strong perfect elimination ordering, [24])**
Eine perfekte Eliminationsordnung $(v_1, \ldots, v_n)$ eines Graphen $G = (V, E)$ ist *strong perfect* genau dann, wenn für alle $i < j, k < l$ gilt:

$$\{v_i v_l, v_i v_k, v_j v_k\} \subseteq E \Rightarrow v_l v_j \in E$$

Eine strong perfect elimination ordering lässt sich mit einem Zeitaufwand von $\mathcal{O}(n^3)$ für einen gegebenen strongly chordalen Graphen finden [1].

Eine weitere Eliminationsordnung ist die *simple Eliminationsordnung*. Dabei müssen die entfernten Knoten sogenannte *simple Knoten* sein.

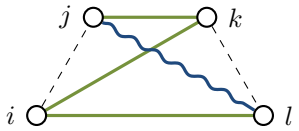



**Abbildung 2.5:** Skizze für Definition 2.17. Wenn die Kanten $il$, $ik$ und $jk$ (grün) vorhanden sind, muss auch die Kante $jl$ (blau gewellt) vorhanden sein. Über die Kanten $ij$ und $kl$ (gestrichelt) wird keine Aussage gemacht.

**Definition 2.18 (simpler Knoten, [24])**
Ein Knoten $v$ ist ein *simpler Knoten* im Graphen $G$ genau dann, wenn für alle seine Nachbarn $x, y \in N[v]$ gilt:

$$N[x] \subseteq N[y] \text{ oder } N[y] \subseteq N[x]$$

Aus Definition 2.18 folgt, dass es für die Nachbarn $u_1, \ldots, u_i$ eines simplen Knotens $v$ eine lineare Ordnung auf deren Nachbarschaft gibt:

$$N[u_1] \subseteq \ldots \subseteq N[u_i]$$

**Definition 2.19 (simple Eliminationsordnung, [24])**
Eine Eliminationsordnung $v_1, \ldots, v_n$ ist eine *simple Eliminationsordnung* genau dann, wenn für alle $i \in \{1, \ldots, n\}$ gilt: $v_i$ ist simpel in $G[\{v_i, \ldots, v_n\}]$.

**Lemma 2.1** [24] Ein Graph $G$ ist strongly chordal genau dann, wenn jeder induzierte Teilgraph von $G$ einen simplen Knoten besitzt.

Aus Lemma 2.1 folgt nun unmittelbar Satz 2.2:

**Satz 2.2** Ein Graph $G$ ist strongly chordal genau dann, wenn $G$ eine simple Eliminationsordnung hat.

### 2.6.3 Intervallgraphen

Die Idee bei Intervallgraphen ist es, die Überschneidung von verschiedenen Zeitintervallen darzustellen. Dazu wird für jedes Intervall ein Knoten

erstellt und diese dann miteinander verbunden, wenn sich die entsprechenden Intervalle überschneiden. Abbildung 2.6 zeigt dies an einem Beispiel.

**Definition 2.20 (Intervallgraph)**
Ein Graph $G = (V, E)$ ist ein Intervallgraph genau dann, wenn eine Menge von Intervallen $\mathcal{I} = \{I_1, \ldots, I_n\}$ existiert, sodass gilt:

$$V = \{v_1, \ldots, v_n\}$$
$$E = \{v_i v_j \mid I_i \cap I_j \neq \emptyset;\ i \neq j\}$$

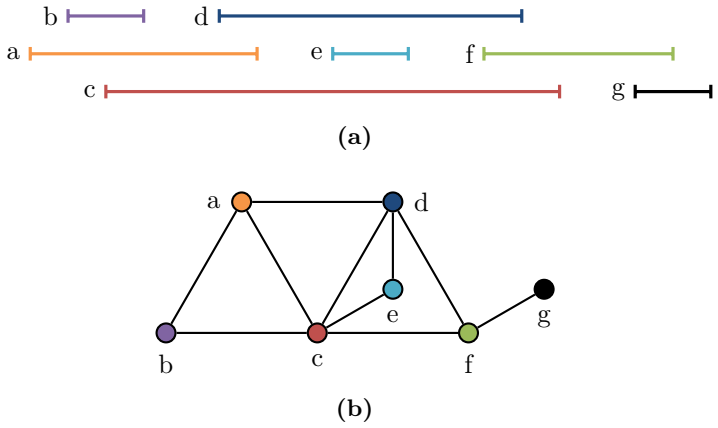



**Abbildung 2.6:** Eine Mengen von Intervallen (a) und der dazugehörige Intervallgraph (b) (Vorlage: [20])

Intervallgraphen sind strongly chordal [10, Abschnitt 4.4][22] und können in Linearzeit erkannt werden [6].

## 2.7 Distanzerbliche Graphen

Bei distanzerblichen Graphen handelt es sich um eine Klasse von Graphen, in denen der Abstand zwischen zwei Knoten in jedem zusammenhängenden induzierten Teilgraph gleich ist. Das heißt, dass (so lange es einen Pfad gibt) sich die Entfernung zwischen zwei Knoten nicht verändert, unabhängig davon, wie viele Knoten entfernt werden.

Ursprünglich wurden distanzerbliche Graphen in [31] vorgestellt. In [2] wurde gezeigt, dass sie sich auch über das Verbot von Teilgraphen

definieren lassen. Dabei handelt es sich um die Graphen House, Hole ($C_k$, $k \geq 5$), Domino und Gem.

**Definition 2.21 (distanzerblicher Graph, [2])**
Ein Graph $G$ ist *distanzerblich* (engl.: distance hereditary) genau dann, wenn $G$ (House, Hole, Domino, Gem)-frei ist.

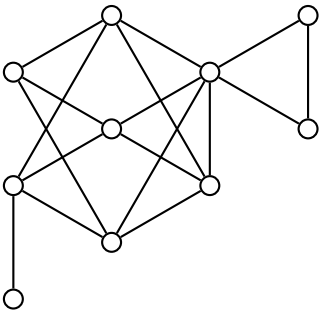


**Abbildung 2.7:** Beispiel für einen distanzerblichen Graphen (Vorlage: [19])

Zwar sind Holes (Kreise der Länge 5 oder größer) verboten, aber nicht Kreise der Länge 4. Somit sind distanzerbliche Graphen nicht notwendigerweise chordal. Umgekehrt beinhalten chordale Graphen bereits per Definition kein House, Hole oder Domino als induzierten Teilgraphen. Es gilt somit Lemma 2.2.

**Lemma 2.2** Ein chordaler Graph $G$ ist Gem-frei $\Rightarrow$ $G$ ist distanzerblich.

Da jede Sun ein Gem enthält, ist ein chordaler distanzerblicher Graph auch immer strongly chordal.

## 2.8 Dually chordale Graphen

Bei dually chordalen Graphen handelt es sich um eine Graphenklasse, die sich über eine Nachbarschaftsordnung ihrer Knoten definiert. Basis dafür ist ein sogenannter maximaler Nachbar.

**Definition 2.22 (maximaler Nachbar)**
Ein Knoten $u \in N[v]$ ist ein *maximaler Nachbar* von $v$ genau dann, wenn gilt:

$$\forall\, w \in N[v] : N[w] \subseteq N[u]$$

Kann nun ein Graph $G$ schrittweise reduziert werden, wobei jeder entfernte Knoten $v$ einen maximalen Nachbarn $u$ im Restgraphen hat, spricht man von einer maximalen Nachbarschaftsordnung. Dies schließt auch die Möglichkeit ein, dass $v$ sein eigener maximaler Nachbar ist ($v = u$).

**Definition 2.23 (maximale Nachbarschaftsordnung)**
Eine Ordnung von Knoten $(v_1, \ldots, v_n)$ ist eine *maximale Nachbarschaftsordnung* genau dann, wenn für alle $i \in \{1, \ldots, n\}$ jeder Knoten $v_i$ einen maximalen Nachbar in $G[\{v_i, \ldots, v_n\}]$ hat.

Dually chordale Graphen definieren sich nun durch eine maximale Nachbarschaftsordnung.

**Definition 2.24 (dually chordaler Graph)**
Ein Graph $G$ ist *dually chordal* genau dann, wenn $G$ eine maximale Nachbarschaftsordnung hat.

Im Laufe der Arbeit werden noch weitere Charakterisierungen für dually chordale Graphen genannt. Satz 3.9 (S. 30) fasst diese zusammen.

Jeder Graph $G$ lässt sich in einen dually chordalen Graphen $G'$ umwandeln, indem man einen Knoten $v$ einfügt, wobei $v$ mit allen Knoten in $G$ benachbart ist. Auf diese Weise ist $v$ ein maximaler Nachbar für alle anderen Knoten. Als Folge daraus kann ein dually chordaler Graph jeden Graphen als Teilgraphen haben. Somit sind dually chordale Graphen nicht notwendigerweise chordal.

Man kann die Definition der dually chordalen Graphen erweitern, indem auch jeder induzierte Teilgraph dually chordal sein muss. Die so entstehende Klasse nennt man *hereditary dually chordal*.

**Satz 2.3** [8] Ein Graph $G$ ist strongly chordal genau dann, wenn jeder induzierte Teilgraph von $G$ dually chordal ist.

Zwar sind die strongly chordalen Graphen eine Teilmenge der dually chordalen Graphen, aber nicht jeder Graph, der chordal und dually chordal ist, ist auch strongly chordal. Abbildung 2.8 stellt einen solchen Graphen dar.

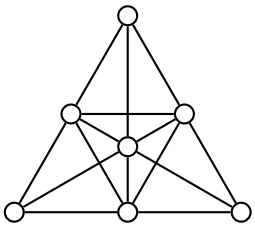


**Abbildung 2.8:** Ein Graph der chordal, dually chordal jedoch nicht strongly chordal ist.

## 2.9 Bäume

Bäume sind eine der bekanntesten Graphenklassen. Bei ihnen handelt es sich um zusammenhängende Graphen ohne Zyklen ($C_k$, $k \geq 3$). Nicht zusammenhängende Graphen, die frei von Zyklen sind, werden als Wald bezeichnet.

**Definition 2.25 (Baum)**
Ein zusammenhängender Graph ist ein Baum genau dann, wenn er keinen Zyklus besitzt.

Es gibt noch weitere Charakterisierungen für Bäume: Beispielsweise ist ein Graph $(V, E)$ ein Baum genau dann, wenn er zusammenhängend ist und $|V| = |E| + 1$ ist.

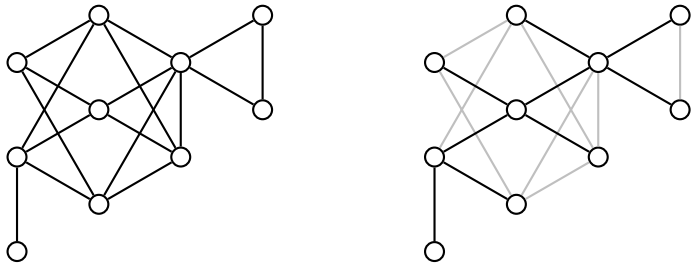
### 2.9.1 Spannbäume

Eine Anwendung von Bäumen ist das Konzept der Spannbäume. Hierbei handelt es sich um einen Baum $T$, der ein (in der Regel nicht induzierter) Teilgraph eines Graphen $G$ ist. Allerdings ist jeder Knoten von $G$ auch in $T$ vorhanden.

**Definition 2.26 (Spannbaum)**
Gegeben sei ein zusammenhängender Graph $G = (V_g, E_g)$ und ein Baum $T = (V_t, E_t)$. $T$ ist ein *Spannbaum* von $G$ genau dann, wenn die beiden nachfolgenden Bedingungen erfüllt sind:

$$V_t = V_g$$
$$E_t \subseteq E_g$$

Es ist leicht zu sehen, dass jeder zusammenhängende Graph auch einen Spannbaum besitzt. Dieser muss jedoch nicht eindeutig sein. Abbildung 2.9 stellt einen Graphen mit einem Spannbaum dar.



**(a)** **(b)**

**Abbildung 2.9:** Beispiel für einen Graphen (a) und einen Spannbaum (b) (Vorlage für (a): [19])

Spannbäume bieten eine weitere Möglichkeit, dually chordale Graphen zu charakterisieren.

**Satz 2.4** [8] Ein Graph $G$ ist dually chordal genau dann, wenn er einen Spannbaum $T$ besitzt, so dass jede maximale Clique in $G$ einen Teilbaum von $T$ induziert.

### 2.9.2 Blätter

Blätter sind eine spezielle Sorte von Knoten, die man üblicherweise mit Bäumen in Verbindung bringt. Dabei handelt es sich um Knoten, die nur einen Nachbarn haben.

**Definition 2.27 (Blatt)**
In einem Graphen ist ein Knoten $v$ ein *Blatt* genau dann, wenn $v$ genau einen Nachbarn hat ($|N(v)| = 1$).

Es ist leicht zu sehen, dass jeder Baum (mit zwei oder mehr Knoten) mindestens zwei Blätter besitzt. Trotzdem können auch in anderen Graphen Blätter vorkommen.

## 2.10 Cliquenweite

Bei Cliquenweite handelt es sich um ein Komplexitätsmaß für Graphen. Als Basis dienen vier Operationen, mit denen sich jeder Graph erzeugen lässt:

$\odot_i$ Es wird ein neuer Knoten erstellt und mit dem Label $i$ versehen.

$G_1 \oplus G_2$ Wurden bereits zwei Graphen $G_1$ und $G_2$ (mit disjunkten Knotenmengen) erzeugt, werden sie nun als ein Graph $G$ betrachtet. Es werden jedoch keine neuen Kanten erstellt. Somit ist $G$ nicht zusammenhängend.

$\eta_{i,j}(G)$ Bei dieser Operation werden zwei Knoten $u$ und $v$ miteinander verbunden, wenn $u$ das Label $i$ und $v$ das Label $j$ hat. Diese Operation betrifft alle Knoten im Graphen mit den entsprechenden Labels. Außerdem gilt, dass $i \neq j$ sein muss. Es ist die einzige Möglichkeit, um Kanten zu erzeugen.

$\rho_{i \to j}(G)$ Vorhandene Labels können umbenannt werden. Dabei erhalten alle Knoten mit dem Label $i$ das Label $j$.

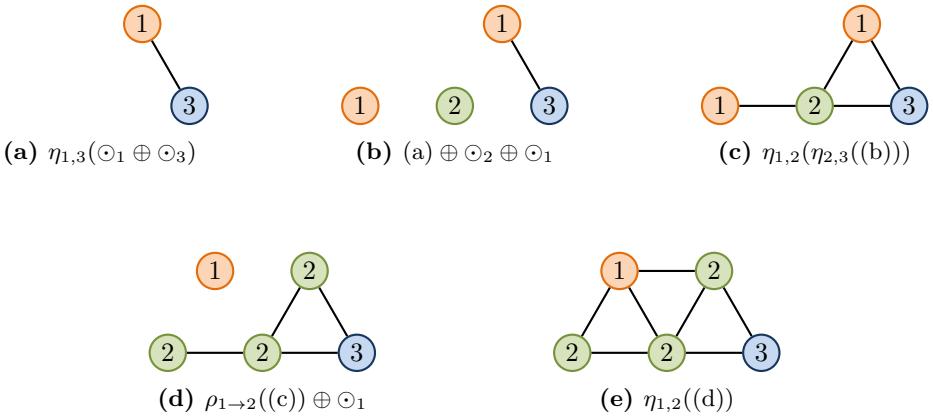
Die Cliquenweite eines Graphen definiert sich nun über diese vier Operationen.

**Definition 2.28 (Cliquenweite)**
Die Cliquenweite eines Graphen $G$ ist die minimal notwendige Anzahl an verschiedenen Labels, die nötig ist, um $G$ mit den Operationen $\odot_i$, $\oplus$, $\eta_{i,j}$ und $\rho_{i \to j}$ zu erzeugen.

Anhand eines Gems sei nachfolgend demonstriert, wie sich mit den oben genannten Operationen ein Graph erzeugen lässt. Abbildung 2.10 stellt dies graphisch dar.

(a) Begonnen wird mit dem Erstellen von zwei Knoten mit den Labels 1 und 3. Diese werden zu einem Graphen zusammengefügt und mit einer Kante verbunden.

$$G := \eta_{1,3}(\odot_1 \oplus \odot_3)$$




**(a)** $\eta_{1,3}(\odot_1 \oplus \odot_3)$ **(b)** (a) $\oplus \odot_2 \oplus \odot_1$ **(c)** $\eta_{1,2}(\eta_{2,3}((\text{b})))$

**(d)** $\rho_{1\to 2}((\text{c})) \oplus \odot_1$ **(e)** $\eta_{1,2}((\text{d}))$

**Abbildung 2.10:** Erstellen eines Gems

(b) Als nächstes wird der bisherige Graph um zwei weitere Knoten mit den Labels 1 und 2 erweitert.

$$G := G \oplus \odot_2 \oplus \odot_1$$

(c) Im nächsten Schritt verbindet man den Knoten mit Label 2 mit den Knoten, welche die Labels 1 und 3 haben.

$$G := \eta_{1,2}(\eta_{2,3}(G))$$

(d) Nun erhalten die Knoten mit Label 1 das Label 2. Außerdem wird ein neuer Knoten mit Label 1 eingefügt.

$$G := \rho_{1\to 2}(G) \oplus \odot_1$$

(e) Abschließend werden erneut die Knoten mit Label 1 mit den Knoten mit Label 2 verbunden.

$$G := \eta_{1,2}(G)$$

Der so erzeugte Graph lässt sich auch in einer einzigen Formel ausdrücken:

$$G := \eta_{1,2}(\rho_{1\to 2}(\eta_{1,2}(\eta_{2,3}(\eta_{1,3}(\odot_1 \oplus \odot_3) \oplus \odot_2 \oplus \odot_1))) \oplus \odot_1)$$

Eine solche Formel wird auch als Cliquenweite-Ausdruck bezeichnet.

Kapitel 3

# Hypergraphen

Dieses Kapitel befasst sich mit Hypergraphen. Der Fokus liegt dabei auf den azyklischen Hypergraphen sowie der Charakterisierung ihrer Linegraphen.

## 3.1 Einleitung

Hypergraphen sind eine Verallgemeinerung von Graphen. Kanten sind nun nicht mehr auf zwei Knoten beschränkt, sondern können beliebig viele (jedoch nicht null) Knoten beinhalten.

**Definition 3.1 (Hypergraph)**
Ein *Hypergraph* $H$ ist ein 2-Tupel $H = (V, \mathcal{E})$. Dabei ist $V$ eine endliche Menge von Knoten und $\mathcal{E} \subseteq \{e \mid e \subseteq V;\ e \neq \emptyset\}$ die Menge der Hyperkanten.

Da Hypergraphen nicht auf genau zwei Knoten pro Kante beschränkt sind, reicht die bisherige Definition für Graphisomorphie (Definition 2.5, S. 3) nicht mehr aus. Die ursprüngliche Definition wird deshalb so erweitert, dass Kanten nicht auf zwei Knoten beschränkt sind.

**Definition 3.2 (Isomorphie von Hypergraphen)**
Zwei Hypergraphen $H_1 = (V_1, \mathcal{E}_1)$ und $H_2 = (V_2, \mathcal{E}_2)$ heißen *isomorph* genau dann, wenn eine bijektive Abbildung $\varphi : V_1 \to V_2$ existiert, so dass für alle Hyperkanten $\{v_1, \ldots, v_i\} \in \mathcal{E}_1$ gilt:

$$\{v_1, \ldots, v_i\} \in \mathcal{E}_1 \Leftrightarrow \{\varphi(v_1), \ldots, \varphi(v_i)\} \in \mathcal{E}_2$$

Die Isomorphie zweier Hypergraphen wird wie folgt dargestellt: $H_1 \sim H_2$

## 3.2 Umwandlungsfunktionen

Es gibt verschiedene Möglichkeiten, Graphen und Hypergraphen in einander umzuformen. In diesem Abschnitt werden drei Varianten vorgestellt.

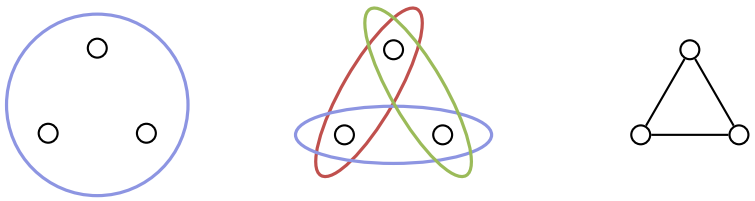
### 3.2.1 2-Section Graphen

Die 2-Section Graphen sind eine Möglichkeit, einen Hypergraphen $H$ als Graphen darzustellen. Dazu werden die Knoten von $H$ übernommen und miteinander verbunden, wenn sie in der gleichen Hyperkante liegen.

**Definition 3.3 (2-Section Graph)**
Es sei $H = (V, \mathcal{E})$ ein Hypergraph. Der *2-Section Graph* $2Sec(H) = (V_2, E_2)$ von $H$ ist dann wie folgt definiert:

$$V_2 = V$$
$$E_2 = \{uv \mid \exists\, e \in \mathcal{E}\ u, v \in e\}$$

Bei dieser Umwandlung gehen jedoch Informationen über den Hypergraphen verloren. Dadurch lässt sich ein 2-Section Graph nicht eindeutig einem Hypergraphen zuordnen. Abbildung 3.1 stellt zwei Hypergraphen und deren 2-Section Graphen dar.



**(a)** **(b)** **(c)**

**Abbildung 3.1:** Die Hypergraphen (a) und (b) sowie ihr 2-Section Graph (c)

### 3.2.2 Linegraphen

Die Idee der Linegraphen ist es, die Nachbarschaft von Hyperkanten in einem Hypergraphen zu betrachten und als Graphen darzustellen. Dazu werden die Hyperkanten eines gegebenen Hypergraphen $H$ als Knoten

betrachtet. Zwei Knoten sind dann durch eine Kante verbunden, wenn die entsprechenden Hyperkanten einen gemeinsamen Knoten in $H$ haben. Abbildung 3.2 stellt ein Beispiel für einen Hypergraphen und seinen Linegraphen dar.

**Definition 3.4 (Linegraph)**
Gegeben sei ein Hypergraph $H = (V, \mathcal{E})$. Der Graph $L(H) = (\mathcal{E}, E)$ mit

$$E = \{ef \mid e, f \in \mathcal{E}; e \neq f; e \cap f \neq \emptyset\}$$

ist dann der *Linegraph* von $H$.

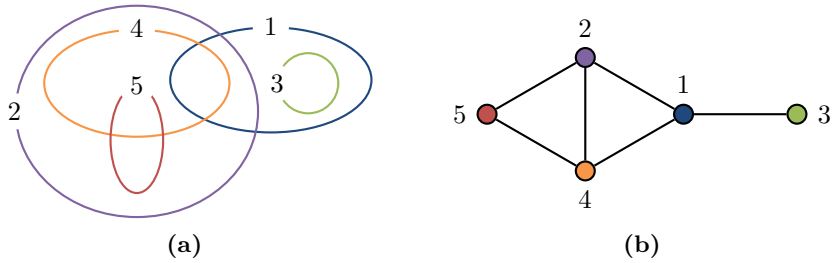



**Abbildung 3.2:** Beispiel für einen Hypergraphen (a) und seinen Linegraphen (b)

In Abschnitt 3.5 werden die Linegraphen von verschiedenen Hypergraphen charakterisiert.

### 3.2.3 Cliquen(hyper)graph

Eine Variante, einen Hypergraphen aus einem Graphen $G$ zu erzeugen, ist das Bilden des Cliquenhypergraphen. Dabei wird für jede maximale Clique in $G$ eine Hyperkante erzeugt.

**Definition 3.5 (Cliquenhypergraph)**
Es sei $G = (V, E)$ ein Graph. Dessen *Cliquenhypergraph* $\mathcal{C}(G) = (V, \mathcal{E})$ sei dann wie folgt definiert:

$$\mathcal{E} := \{c \mid c \text{ ist maximale Clique in } G\}$$

Erzeugt man für jede maximale Clique statt einer Hyperkante einen Knoten und verbindet diese, falls die entsprechenden Cliquen einen

gemeinsamen Knoten in $G$ besitzen, so wird der erzeugt Graph als Cliquengraph bezeichnet.

**Definition 3.6 (Cliquengraph)**
Für einen gegebenen Graphen $G = (V_g, E_g)$ sei sein *Cliquengraph* $K(G) = (V_k, E_k)$ wie folgt definiert:

$$V_k := \{c \mid c \text{ ist maximale Clique in } G\}$$
$$E_k := \{cd \mid c \cap d \neq \emptyset\}$$

Wendet man die Definition von Linegraphen auf die von Cliquenhypergraphen an, stellt man fest, dass der Cliquengraph genau der Linegraph eines Cliquenhypergraphen ist.

**Lemma 3.1** Für jeden Graphen $G$ gilt: $K(G) = L(\mathcal{C}(G))$.

Der Cliquengraph bietet auch eine weitere Möglichkeit, um dually chordale Graphen (siehe Abschnitt 2.8) zu definieren.

**Satz 3.1** [8] Ein Graph $G$ ist dually chordal genau dann, wenn $G$ der Cliquengraph eines chordalen Graphen ist.

## 3.3 Eigenschaften

Hypergraphen können Eigenschaften besitzen, die üblicherweise bei Graphen nicht betrachtet werden. So stellen Bäume die einzige Graphenklasse dar, welche die beiden nachfolgend vorgestellten Eigenschaften hat.

### 3.3.1 Helly-Eigenschaft

Die Helly-Eigenschaft trifft eine Aussage über gemeinsame Knoten von Kanten. Hat ein Hypergraph die Helly-Eigenschaft, dann existiert für jede Menge von Hyperkanten, die sich paarweise schneiden, ein gemeinsamer Knoten, der in jeder der Hyperkanten vorhanden ist.

**Definition 3.7 (Helly-Eigenschaft)**
Ein Hypergraph $H = (V, \mathcal{E})$ hat die Helly-Eigenschaft, wenn für alle $\mathcal{E}^* \subseteq \mathcal{E}$ gilt:

$$\Big(\forall\, e_1, e_2 \in \mathcal{E}^* : e_1 \cap e_2 \neq \emptyset\Big) \Rightarrow \bigcap_{e \in \mathcal{E}^*} e \neq \emptyset$$

Für den Linegraphen $\mathcal{L}$ eines Hypergraphen $H$ bedeutet die Helly-Eigenschaft, dass es für jede maximale Clique in $\mathcal{L}$ einen gemeinsamen Knoten der entsprechenden Hyperkanten in $H$ gibt.

### 3.3.2 Conformalität

Conformalität stellt eine zusätzliche Einschränkung für den 2-Section Graphen eines Hypergraphen dar. Ist ein Hypergraph conformal, so gibt es für jede (maximale) Clique in dessen 2-Section Graph auch eine Hyperkante, die alle Knoten der Clique enthält.

**Definition 3.8 (conformal)**
Ein Hypergraph $H = (V, \mathcal{E})$ ist *conformal*, wenn für jede Clique $K$ mit den Knoten $V_k$ in $2Sec(H)$ gilt:

$$\exists\, e \in \mathcal{E} \text{ mit } V_k \subseteq e$$

Ähnlich wie die Helly-Eigenschaft lässt sich Conformalität auch über die Knoten benachbarter Kanten beschreiben. Diese Gesetzmäßigkeit ist als *Gilmore Theorem* bekannt.

**Satz 3.2 (Gilmore Theorem)** [4] Ein Hypergraph $H = (V, \mathcal{E})$ ist *conformal* genau dann, wenn für alle 3-elementigen Knotenmengen $\{e_1, e_2, e_3\} \subseteq \mathcal{E}$ ein $e \in \mathcal{E}$ existiert mit $(e_1 \cap e_2) \cup (e_1 \cap e_3) \cup (e_2 \cap e_3) \subseteq e$.

Zwar treffen sowohl Conformalität als auch die Helly-Eigenschaft Aussagen über paarweise benachbarte Hyperkanten, jedoch sind sie nicht äquivalent und bedingen auch einander nicht. Abbildung 3.3 stellt zwei Hypergraphen dar, von denen einer conformal ist und einer die Helly-Eigenschaft hat.

(a) (b)

**Abbildung 3.3:** Unterschied von Conformalität und Helly-Eigenschaft: Der Hypergraph (a) ist conformal, aber erfüllt nicht die Helly-Eigenschaft. Der Hypergraph (b) hingegen erfüllt die Helly-Eigenschaft, aber ist nicht conformal.

## 3.4 Azyklische Hypergraphen

Dieser Abschnitt definiert drei Klassen von azyklischen Hypergraphen. Zusätzlich wird die Graham-Reduktion vorgestellt.

### 3.4.1 $\alpha$-azyklische Hypergraphen

Für $\alpha$-azyklische Hypergraphen, die auch als *dual hypertrees* bezeichnet werden, gibt es verschiedene Definitionen. Eine davon bezieht sich auf Conformalität und den 2-Section Graphen.

**Definition 3.9 ($\alpha$-azyklischer Hypergraph)**
Ein Hypergraph $H$ ist *$\alpha$-azyklisch* genau dann, wenn $H$ conformal und $2Sec(H)$ chordal ist.

Aus der Definition für $\alpha$-azyklische Hypergraphen und für Conformalität folgt nun unmittelbar, dass der Cliquenhypergraph eines chordalen Graphen $\alpha$-azyklisch ist.

**Lemma 3.2** [10, Corollary 1.3.2] Ein Graph $G$ ist chordal genau dann, wenn sein Cliquenhypergraph $\mathcal{C}(G)$ $\alpha$-azyklisch ist.

Ein nützlicher Aspekt von $\alpha$-azyklischen Hypergraphen ist die *Graham-Reduktion*. Dabei handelt es sich um zwei einfache Eliminationsregeln für einen gegeben Hypergraphen.

**Definition 3.10 (Graham-Reduktion)**
Unter der *Graham-Reduktion* versteht man das wiederholte Anwenden der beiden nachfolgenden Regeln auf einen Hypergraphen $H = (V, \mathcal{E})$.

(i) Ist ein Knoten $v$ in genau einer Hyperkante enthalten, dann kann $v$ entfernt werden.

(ii) Ist eine Hyperkante $e$ vollständig in einer anderen Hyperkante $f$ enthalten ($e \subseteq f$), kann $e$ entfernt werden.

Führt die Reduktion dazu, dass $H$ nur noch eine leere Kante besitzt ($\mathcal{E} = \{\emptyset\}$), so sagt man, die Reduktion war *erfolgreich*.

**Satz 3.3** [3] Ein Hypergraph $H$ ist $\alpha$-azyklisch genau dann, wenn die Graham-Reduktion erfolgreich ist für $H$.

Aufgrund von Satz 3.3 ist die Graham-Reduktion nicht nur eine Eliminationsregel für $\alpha$-azyklische Hypergraphen, sondern auch eine Konstruktionsregel.

### 3.4.2 $\beta$-azyklische Hypergraphen

Erweitert man die Bedingungen für einen $\alpha$-azyklischen Hypergraphen so, dass auch jede Teilmenge der Hyperkanten sie erfüllt, so erhält man die Klasse der $\beta$-azyklischen Hypergraphen.

**Definition 3.11 ($\beta$-azyklischer Hypergraph, [21])**
Ein Hypergraph $H = (V, \mathcal{E})$ ist *$\beta$-azyklisch* genau dann, wenn für alle $\mathcal{E}' \subseteq \mathcal{E}$ gilt: $\mathcal{E}'$ ist $\alpha$-azyklisch.

$\beta$-azyklische Hypergraphen werden auch als *totally balanced* bezeichnet. Sie erfüllen die Helly-Eigenschaft [4].

### 3.4.3 $\gamma$-azyklische Hypergraphen

Eine Teilmenge der $\beta$-azyklischen Hypergraphen sind die $\gamma$-azyklischen. Sie definieren sich über einen nicht erlaubten Teilgraphen, einen sogenannten $\gamma$-cycle.

**Definition 3.12 ($\gamma$-cycle, [21])**
Ein *$\gamma$-cycle* in einem Hypergraphen $H$ ist eine Folge $(v_1, e_1, \ldots, v_k, e_k)$ mit $k \geq 3$, welche die folgenden Bedingungen erfüllt:

(i) $v_1, \ldots, v_k$ sind paarweise verschiedene Knoten in $H$.

(ii) $e_1, \ldots, e_k$ sind paarweise verschiedene Hyperkanten in $H$ und $e_{k+1} = e_1$.

(iii) $\forall\, i\ (1 \leq i \leq k) : v_i \in e_i \cap e_{i+1}$

(iv) $\forall\, i\ (1 \leq i < k) : \forall\, j\ (j \neq i, i+1) : v_i \notin e_j$

Bedingung (iv) in Definition 3.12 bedeutet, dass alle Knoten $v_i$ nur in den Hyperkanten $e_i$ und $e_{i+1}$ sind, jedoch nicht der Knoten $v_k$. Dieser darf auch in den anderen Hyperkanten enthalten sein. Abbildung 3.4 gibt ein Beispiel für einen $\gamma$-cycle.

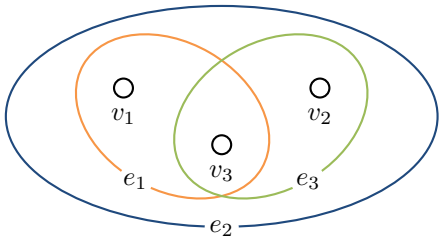



**Abbildung 3.4:** Beispiel für einen $\gamma$-cycle

**Definition 3.13 ($\gamma$-azyklischer Hypergraph, [21])**
Ein Hypergraph ist *$\gamma$-azyklisch* genau dann, wenn er keinen $\gamma$-cycle enthält.

Für azyklische Hypergraphen gilt die folgende Hierarchie:

**Satz 3.4** [21] $\gamma$-azyklisch $\subset$ $\beta$-azyklisch $\subset$ $\alpha$-azyklisch

## 3.5 Linegraphen von azyklischen Hypergraphen

Dieser Abschnitt beschäftigt sich mit der Charakterisierung der Linegraphen von $\alpha$-azyklischen Hypergraphen und deren Unterklassen ($\beta$- und $\gamma$-azyklisch).

Zuerst wird gezeigt, dass die Klasse der Linegraphen von $\alpha$-azyklischen Hypergraphen genau die Klasse der dually chordalen Graphen ist. Dazu seien die folgenden Aussagen wiederholt:

- $K(G) = L(\mathcal{C}(G))$ (Lemma 3.1)
- $G$ ist chordal $\Leftrightarrow$ $\mathcal{C}(G)$ ist $\alpha$-azyklisch (Lemma 3.2)
- $G$ ist dually chordal $\Leftrightarrow$ $G$ ist der Cliquengraph eines chordalen Graphen. (Satz 3.1)
- $H$ ist $\alpha$-azyklisch $\Leftrightarrow$ Die Graham-Reduktion ist erfolgreich für $H$. (Satz 3.3)

**Lemma 3.3** $H$ ist $\alpha$-azyklisch $\Rightarrow$ $L(H)$ ist dually chordal.

**Beweis**
$H = (V, \mathcal{E})$ ist $\alpha$-azyklisch. Man füge nun in jede Hyperkante $e \in \mathcal{E}$ einen Knoten $v_e$ so ein, dass $v_e$ nur in $e$ enthalten ist. Der so entstehende Hypergraph $H'$ ist weiterhin $\alpha$-azyklisch (Graham-Reduktion).

Es gilt, dass $L(H) \sim L(H')$, da das Einfügen der Knoten nichts an der Nachbarschaft der Hyperkanten geändert hat.

Aufgrund der eingefügten Knoten $v_e$ gibt es für jede Hyperkante $e$ eine maximale Clique in $2Sec(H')$. Angenommen es gäbe eine weitere maximale Clique $K$, dann wären ihre Knoten aufgrund der Confomalität von $H'$ auch in einer Hyperkante $e$ vorhanden. $K$ ist dann jedoch entweder nicht maximal, oder keine weitere maximale Clique. Somit gilt:
$\mathcal{C}(2Sec(H')) \sim H'$.

Es sei nun $G := 2Sec(H')$. $G$ ist chordal und $\mathcal{C}(G) \sim H'$. Daraus folgt, dass $L(H') \sim L(\mathcal{C}(G)) = K(G)$. Da $G$ chordal ist, ist somit $K(G)$ dually chordal. Also gilt $L(H) \sim L(H')$ ist dually chordal.

□

**Lemma 3.4** Für alle dually chordalen Graphen $G$ existiert ein $\alpha$-azyklischer Hypergraph $H$ mit $L(H) \sim G$.

**Beweis**
$G$ ist dually chordal. Das heißt, es existiert ein chordaler Graph $G'$, so dass $K(G') \sim G$. Es gilt $K(G') = L(\mathcal{C}(G'))$. Es sei nun $H := \mathcal{C}(G')$. $H$ ist

$\alpha$-azyklisch. Da $L(H) = L(\mathcal{C}(G')) = K(G')$ ist und $K(G') \sim G$, gilt auch $L(H) \sim G$.

□

Aus den Lemmata 3.3 und 3.4 folgt nun unmittelbar Satz 3.5:

**Satz 3.5** Ein Graph ist dually chordal genau dann, wenn er der Linegraph eines $\alpha$-azyklischen Hypergraphen ist.

In Verbindung mit Satz 2.3 (S. 11) ergibt sich nun für $\beta$-azyklische Hypergraphen, dass diese strongly chordal sind.

**Satz 3.6** Die Linegraphen von $\beta$-azyklischen Hypergraphen sind strongly chordal.

**Beweis**
Es seien $H = (V, \mathcal{E})$ ein $\beta$-azyklischer Hypergraph und $\mathcal{L} = L(H) = (\mathcal{E}, E)$ sein Linegraph.

Per Definition gilt, dass alle Teilmengen $\mathcal{E}'$ der Hyperkanten von $H$ ($\mathcal{E}' \subseteq \mathcal{E}$) einen $\alpha$-azyklischen Hypergraphen bilden. Übertragen auf den Linegraphen bedeutet dies, dass jeder induzierte Teilgraph $\mathcal{L}[\mathcal{E}']$ dually chordal ist.

Es gilt, dass ein Graph strongly chordal ist, wenn jeder induzierte Teilgraph von $G$ dually chordal ist (Satz 2.3). Somit ist $\mathcal{L}$ ebenfalls strongly chordal.

□

Als nächstes wird gezeigt, dass die Linegraphen von $\gamma$-azyklischen Hypergraphen distanzerblich chordal sind. Dazu werden folgende Aussagen verwendet:

- $H$ ist $\gamma$-azyklisch $\Rightarrow$ $H$ ist $\beta$-azyklisch. (Satz 3.4)
- $\beta$-azyklische Hypergraphen erfüllen die Helly-Eigenschaft. [4]
- $G$ ist chordal und Gem-frei $\Rightarrow$ $G$ ist distanzerblich. (Lemma 2.2, S. 10)

**Satz 3.7** Die Linegraphen von $\gamma$-azyklischen Hypergraphen sind distanzerblich chordal.

**Beweis**
Es seien $H$ ein $\gamma$-azyklischer Hypergraph und $\mathcal{L} = L(H)$ sein Linegraph.

Da jeder $\gamma$-azyklische Hypergraph auch $\beta$-azyklisch ist, ist $\mathcal{L}$ strongly chordal. $\mathcal{L}$ ist somit distanzerblich, wenn $\mathcal{L}$ Gem-frei ist.

Angenommen, $\mathcal{L}$ enthalte einen Gem $G$ mit den Knoten $e_1$, $e_2$ und $e_3$. Außerdem seien $K_1$, $K_2$ und $K_3$ die maximalen Cliquen in $G$.
Abbildung 3.5 stellt dies dar.

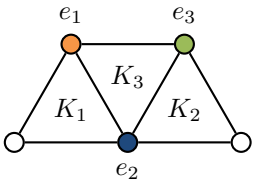



**Abbildung 3.5:** Der Gem $G$

Aufgrund der Helly-Eigenschaft gibt es für jede Clique $K_i$ in $G$ einen Knoten $k_i$ in $H$, der in den Hyperkanten enthalten ist, welche die entsprechende Clique in $\mathcal{L}$ bilden. Somit gilt: $k_1 \in e_1 \cap e_2$, $k_2 \in e_2 \cap e_3$ und $k_3 \in e_1 \cap e_2 \cap e_3$. Abbildung 3.6 stellt dies dar.

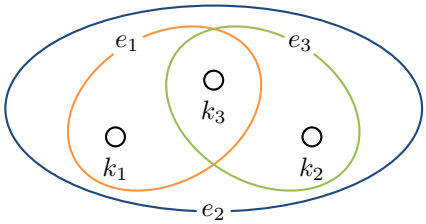



**Abbildung 3.6:** Der Gem $G$ in $H$ – Es wurden nur zu drei der Knoten aus $G$ die entsprechenden Hyperkanten dargestellt.

$(k_1, e_1, k_2, e_2, k_3, e_3)$ bildet nun einen $\gamma$-cycle. Somit kann $\mathcal{L}$ keinen Gem enthalten.

□

### 3.5.1 Charakterisierung mittels Graham-Reduktion

Eine weitere Möglichkeit, die Linegraphen von $\alpha$-azyklischen Hypergraphen zu charakterisieren, ergibt sich aus der Graham-Reduktion. Als Konstruktionsregel angewendet besteht die Graham-Reduktion aus zwei Regeln:

(i) Einfügen eines Knotens in genau eine Hyperkante

(ii) Einfügen einer Hyperkante in eine bereits bestehende

Regel (i) ist für den Linegraphen nicht relevant. Der hinzugefügte Knoten ist nur in genau einer Hyperkante enthalten. Somit haben sich die Nachbarschaften der Kanten durch das Einfügen nicht verändert.

Aus Regel (ii) hingegen ergibt sich für die Hyperkanten eine (gerichtete) Baumstruktur, welche die Reihenfolge darstellt, mit der die Hyperkanten eingefügt wurden. Die Wurzel ist dabei die erste Hyperkante und Blätter die zuletzt eingeführten Hyperkanten. Abbildung 3.7 stellt dies an einem Beispiel dar.

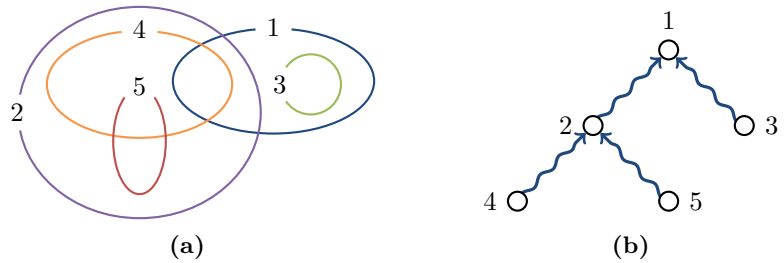



**Abbildung 3.7:** Beispiel für die Baumstruktur der Graham-Reduktion: Der Hypergraph (a) kann erzeugt werden, indem die Hyperkanten in der Reihenfolge ihrer Nummerierung ineinander eingefügt werden (2 in 1, 3 in 1, 4 in 2, 5 in 2). Der Baum (b) gibt diese Ordnung wieder.

Ein solcher Baum $T$ ist nun ein erster Ansatz für den gesuchten Linegraphen. Zwar ist $T$ ein Spannbaum des gesuchten Linegraphen, allerdings werden nicht alle möglichen Nachbarschaften der Hyperkanten wiedergegeben.

Zwei Knoten $a$ und $b$ sind benachbart in einem Linegraphen, wenn ihre entsprechenden Hyperkanten einen gemeinsamen Knoten $v$ besitzen. Die Graham-Reduktion erlaubt das Einfügen von Knoten jedoch nur in genau

eine Hyperkante. Alle anderen Knoten werden geerbt. Eine Hyperkante kann dabei Knoten nur von der Hyperkante erben, in die sie eingefügt wurde. Daraus ergeben sich nun die zwei folgenden Möglichkeiten, wenn $v$ sowohl in $a$ als auch in $b$ ist:

(i) $a$ wurde in $b$ eingefügt oder umgekehrt.

(ii) Es gibt einen gemeinsamen Elternknoten $e$ in $T$, wobei $v$ in die entsprechende Hyperkante eingefügt wurde.

Für den Fall (ii) bedeutet dies, dass auch alle Hyperkanten, deren Knoten auf dem Pfad (in $T$) von $e$ zu $a$ und von $e$ zu $b$ liegen, den Knoten $v$ besitzen. Andernfalls ließe sich $v$ nicht von $e$ auf $a$ und $b$ vererben. Somit sind $a$ und $b$ auch mit allen Knoten auf diesem Pfad benachbart.
Abbildung 3.8 stellt dies dar.

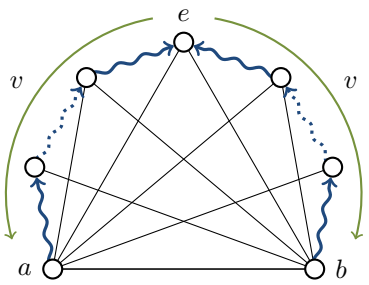



**Abbildung 3.8:** Die Nachbarschaft der Hyperkanten $a$ und $b$ entlang des Spannbaums (blau gewellt): Der Knoten $v$ wird entlang des Spannbaums von $e$ auf $a$ und $b$ vererbt (grün). Somit sind $a$ und $b$ auch mit allen Hyperkanten entlang dieser Pfade benachbart.

Entsprechend der obigen Argumentation ergibt sich nun Definition 3.14 für die Linegraphen von $\alpha$-azyklischen Hypergraphen.

**Definition 3.14 (Linegraph eines $\alpha$-azyklischen Hypergraphen)**
Es sei $P_T(u, v)$ die Menge der Knoten auf dem Pfad von $u$ nach $v$ in $T$ ($u, v \notin P_T(u, v)$).

Ein Graph $G = (V, E)$ ist der Linegraph eines $\alpha$-azyklischen Hypergraphen genau dann, wenn $G$ einen Spannbaum $T$ besitzt, so dass für alle Kanten $uv \in E$ gilt:

$$\forall\, w \in P_T(u, v) : uw, vw \in E$$

Es ist nun zu beweisen, dass die Graphen, welche die Definition 3.14 erfüllen, genau die Linegraphen der $\alpha$-azyklischen Hypergraphen sind. Jedoch ist bereits bekannt, dass es sich dabei um die dually chordalen Graphen handelt (Satz 3.5). Für dually chordale Graphen ist außerdem eine Definition bekannt, die auf einem Spannbaum beruht (Satz 2.4, S. 13). Deswegen wird an dieser Stelle lediglich gezeigt, dass beide Definitionen äquivalent sind.

**Satz 3.8** Es seien $G = (V, E)$ ein Graph sowie $T$ ein Spannbaum von $G$. Außerdem sei $P_{uv}$ die Menge der Knoten auf dem Pfad von $u$ nach $v$ in $T$ $(u, v \notin P_{uv})$.

Die folgenden Aussagen sind äquivalent:

(i) Jede maximale Clique in $G$ induziert einen Teilbaum von $T$.

(ii) Für alle Kanten $uv \in E$ gilt: $\forall\, w \in P_{uv} : uw, vw \in E$

**Beweis**
Es sei $\mathcal{K} \subseteq V$ eine maximale Clique in $G$ mit den Knoten $u$ und $v$.

**(i) $\Rightarrow$ (ii):** Die Clique $\mathcal{K}$ induziert einen Teilbaum von $T$. Somit gilt, dass $P_{uv} \subseteq \mathcal{K}$. Folglich sind auch alle Knoten $w \in P_{uv}$ mit $u$ und $v$ verbunden. Andernfalls wäre $\mathcal{K}$ keine Clique.

**(i) $\Leftarrow$ (ii):** Angenommen, $\mathcal{K}$ induziert keinen Teilbaum von $T$. Dann existiert ein Knoten $w$, der in $P_{uv}$ liegt, jedoch nicht in $\mathcal{K}$. Da $\mathcal{K}$ maximal ist, gibt es einen Knoten $k \in \mathcal{K}$, der nicht mit $w$ verbunden ist.

Die Knoten $u$ und $v$ sind mit $k$ verbunden (alle drei sind in $\mathcal{K}$). Für die entsprechenden Pfade $P_{uk}$ und $P_{vk}$ gilt nun, dass $w$ nicht auf diesen Pfaden liegt $(w \notin P_{uk} \cup P_{vk})$. Andernfalls wäre $k$ mit $w$ verbunden. Abbildung 3.9 stellt dies dar.

Es gibt nun in $T$ zwei mögliche Pfade von $u$ nach $v$: Zum einen $P_{uv}$ über $w$ und zum anderen $P_{uk}$ und $P_{vk}$. Somit ist $T$ kein Baum. Dies steht im Widerspruch zur Voraussetzung.

□

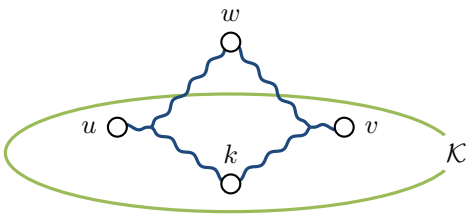



**Abbildung 3.9:** Skizze für den Beweis von Satz 3.8. Die Knoten $k$, $u$ und $v$ liegen in der Clique $\mathcal{K}$ (grün). Die blau gewellten Kanten sind Pfade im Spannbaum $T$.

### 3.5.2 Zusammenfassung der Definitionen

Es folgt nun eine Zusammenfassung der Charakterisierungen für die Linegraphen von $\alpha$-azyklischen Hypergraphen.

**Satz 3.9** Es seien $G = (V, E)$ ein Graph und $P_T(u, v)$ die Menge der Knoten auf dem Pfad von $u$ nach $v$ in $T$ $(u, v \notin P_T(u, v))$.

Die folgenden Aussagen sind äquivalent:

(i) $G$ ist dually chordal.

(ii) $G$ hat eine maximale Nachbarschaftsordnung.

(iii) $G$ ist der Cliquengraph eines chordalen Graphen.

(iv) $G$ ist der Linegraph eines $\alpha$-azyklischen Hypergraphen.

(v) $G$ besitzt einen Spannbaum $T$, so dass jede maximale Clique in $G$ einen Teilbaum von $T$ induziert.

(vi) $G$ besitzt einen Spannbaum $T$, so dass für alle Kanten $uv \in E$ gilt: $\forall\, w \in P_T(u, v) : uw, vw \in E$.

**Beweis**

(i) $\Leftrightarrow$ (ii): Definition 2.24 (S. 11)

(i) $\Leftrightarrow$ (iii): Satz 3.1 (S. 19)

(i) $\Leftrightarrow$ (iv): Satz 3.5 (S. 25)

(i) $\Leftrightarrow$ (v): Satz 2.4 (S. 13)

(v) $\Leftrightarrow$ (vi): Satz 3.8 (S. 29)

□

Kapitel 4

# Dominierende Mengen

Dieses Kapitel befasst sich mit dominierenden Knoten- und Kantenmengen in Graphen. Es werden verschiedene Varianten vorgestellt und Verbindungen zwischen ihnen aufgezeigt.

## 4.1 Einleitung

Angenommen, man hat eine Fläche gegeben, die in einzelne Teilflächen (Parzellen) eingeteilt ist. Ein entsprechendes Beispiel ist in Abbildung 4.1 gegeben. Diese in Parzellen unterteilte Fläche soll nun komplett abgedeckt werden. Denkbare Anwendungen hierfür sind beispielsweise Video-Überwachung, Bewässerung von Feldern oder das Anbieten von Funknetzen.

Um eine solche Abdeckung zu erreichen, werden nun einige Parzellen ausgewählt. Eine gewählte Parzelle deckt dabei sich selbst und ihre benachbarten Parzellen ab. Für das aktuelle Beispiel seien zwei Parzellen benachbart, wenn sie zumindest einen gemeinsamen Punkt haben. In Abbildung 4.1 wurde eine Parzelle gewählt und ihr Abdeckungsbereich grün eingefärbt.

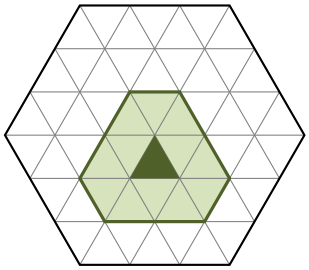


**Abbildung 4.1:** Beispiel für eine in Parzellen unterteilte Fläche. Die dunkelgrün eingefärbte Parzelle deckt sich selbst und ihre Nachbarn (hellgrün) ab.

Gesucht ist nun eine (möglichst kleine) Menge von Parzellen, damit die

gesamte Fläche abgedeckt wird. Eine solche Menge wird als *dominierende Menge* bezeichnet. Überträgt man das Konzept auf einen Graphen, so ergibt sich folgende Definition:

**Definition 4.1 (dominierende Menge)**
Es sei $G = (V, E)$ ein Graph. Eine Knotenmenge $D \subseteq V$ ist *dominierend* für $G$ genau dann, wenn gilt:

$$\bigcup_{d \in D} N[d] = V$$

Das dazugehörige Problem sucht nach einer kleinsten Teilmenge, die dominierend ist. Im gewichteten Fall wird dabei die Summe der Gewichte minimiert. Bereits ungewichtet ist das Problem NP-vollständig [27].

Zur Abgrenzung der in Abschnitt 4.3 vorgestellten Varianten wird eine dominierende Menge, die keinen weiteren Bedingungen unterliegt, in dieser Arbeit als *simple* bezeichnet.

## 4.2 Unabhängige Mengen

Bevor im nächsten Abschnitt verschiedene Varianten von dominierenden Mengen vorgestellt werden, definiert dieser Abschnitt kurz das Konzept von unabhängigen Mengen.

Eine unabhängige Menge (engl.: independent set) ist eine Knotenmenge, bei denen kein Knoten mit einem anderen benachbart ist. Das Ermitteln eines größten independent sets in einem Graphen ist NP-vollständig.

**Definition 4.2 (unabhängige Menge)**
In einem Graphen $G = (V, E)$ ist eine Teilmenge der Knoten $\mathcal{I} \subseteq V$ eine *unabhängige Knotenmenge* genau dann, wenn gilt:

$$\forall\, u, v \in \mathcal{I} : uv \notin E$$

## 4.3 Varianten dominierender Mengen

Die Anforderungen an eine dominierende Menge lassen sich erweitern. Dieser Abschnitt stellt drei Varianten für dominierende Mengen vor.

### 4.3.1 Independent Dominating Set

Ein mögliches Kriterium ist, dass die Knoten der dominierenden Menge ein independent set bilden. Man spricht dann von einem *independent dominating set*.

**Definition 4.3 (independent dominating set)**
In einem Graphen $G = (V, E)$ ist eine dominierende Menge $D \subseteq V$ *unabhängig* genau dann, wenn gilt:

$$\forall\, d, e \in D : de \notin E$$

### 4.3.2 Perfect Dominating Set

Eine weitere Variante wird als *perfect dominating set* bezeichnet. Dabei wird jeder Knoten, der nicht zur dominierenden Menge gehört, von genau einem Knoten dominiert.

**Definition 4.4 (perfect domination)**
Es sei $D \subseteq V$ eine dominierende Menge im Graphen $G = (V, E)$. $D$ ist *perfekt* genau dann, wenn für alle $v \in V \setminus D$ gilt:

$$\exists!\, d \in D : v \in N[d]$$

Bei einem perfect dominating set ist zu beachten, dass es sich nur auf die Knoten bezieht, die nicht zur dominierenden Menge gehören. Es muss sich somit nicht notwendigerweise auch um eine unabhängige dominierende Menge handeln.

### 4.3.3 Efficient Dominating Set

Ist eine dominierende Menge sowohl unabhängig als auch perfekt, dann spricht man von einem *efficient dominating set*. Dabei ist jeder Knoten in

der abgeschlossenen Nachbarschaft von genau einem Knoten der dominierenden Menge.

**Definition 4.5 (efficient dominating set)**
In einem Graphen $G = (V, E)$ ist eine Menge $D \subseteq V$ ein *efficient dominating set* genau dann, wenn gilt:

$$\forall\, v \in V : \exists!\, d \in D : v \in N[d]$$

Während sowohl eine unabhängige als auch eine perfekte dominierende Menge für jeden Graphen möglich sind, besitzt hingegen nicht jeder Graph ein efficient dominating set. Beispiele hierfür sind unter anderem die Graphen $C_4$ und $C_5$.

### 4.3.4 Bekannte Fälle

Für die in Kapitel 2 vorgestellten Graphenklassen ist für einige Varianten ein effizienter Algorithmus bekannt oder es konnte nachgewiesen werden, dass die Variante NP-vollständig ist. Die Tabelle 4.1 gibt dazu eine Übersicht.

| Klasse | Dominierungs-Variante | Komplexität |
|---|---|---|
| chordal | simple | NP-vollständig [5] |
| | independent | Linear [23] |
| | weighted independent | NP-vollständig [32] |
| | efficient | NP-vollständig [13] |
| dually chordal | simple | Linear [7] |
| strongly chordal[1] | simple | Linear [25] |
| | weighted independent | Linear [25] |
| Intervall- | efficient | Linear [14] |

**Tabelle 4.1:** Übersicht über die Komplexität verschiedener Dominierungsprobleme

[1] mit gegebener strong perfect elimination ordering

## 4.4 Dominierende Kantenmengen

Bei Kantenmengen sucht man häufig nach sogenannten Matchings. Dabei handelt es sich um eine Teilmenge der Kanten eines Graphen, wobei zwei Kanten keinen gemeinsamen Knoten besitzen.

**Definition 4.6 (Matching)**
Gegeben sei ein Hypergraph $H = (V, \mathcal{E})$. Eine Menge $M \subseteq \mathcal{E}$ heißt *Matching* genau dann, wenn für alle $e, e' \in M$ $(e \neq e')$ gilt: $e \cap e' = \emptyset$.

Ein solches Matching kann nun auch dominierend sein. Dabei sollen die anderen Kanten dominiert werden. Zusätzlich sei noch die Bedingung gegeben, dass zwischen zwei Kanten des Matchings mindestens zwei weitere Kanten liegen. Man spricht dann von einem *induced* Matching.

**Definition 4.7 (dominating induced Matching)**
Gegeben sei ein Hypergraph $H = (V, \mathcal{E})$ und ein Matching $M$. $M$ ist ein *dominating induced Matching* genau dann, wenn gilt:

$$\forall\, e \in \mathcal{E} : \exists!\, m \in M \text{ mit } m \cap e \neq \emptyset$$

Vergleicht man diese Definition nun mit der für ein efficient dominating set, so stellt man folgenden Zusammenhang fest:

**Satz 4.1** In einem Hypergraphen $H = (V, \mathcal{E})$ ist eine Kantenmenge $M \subseteq \mathcal{E}$ ein dominating induced Matching genau dann, wenn $M$ ein efficient dominating set im Linegraph $L(H)$ ist.

**Beweis**
Es sei $H = (V, \mathcal{E})$ ein Hypergraph, $L(H) = (\mathcal{E}, E) = (V', E)$ dessen Linegraph und $M$ ein dominating induced Matching in $H$.

Es gilt nun per Definition:

$$\forall\, e \in \mathcal{E} : \exists!\, m \in M : m \cap e \neq \emptyset$$

Bildet man nun den Linegraphen $L(H)$, kann man die Aussage wie folgt formulieren:

$$\forall\, e \in V' : \exists!\, m \in M : (me \in E \vee m = e)$$

In einem Graphen ist ein Knoten $m$ mit einem Knoten $e$ verbunden oder identisch ($me \in E \vee m = e$) genau dann, wenn $m$ in der abgeschlossenen Nachbarschaft von $e$ liegt ($m \in N[e]$). Somit lässt sich die obige Aussage erneut umformen.

$$\forall\, e \in V' : \exists!\, m \in M : e \in N[m]$$

Dies entspricht nun der Definition eines efficient dominating sets. Da alle durchgeführten Umformungen Äquivalenzumformungen waren, ist somit auch Satz 4.1 erfüllt.

□

Für einen Hypergraphen $H$ lässt sich somit ein dominating induced Matching ermitteln, indem man ein efficient dominating set in dessen Linegraph $L(H)$ sucht.

## 4.5 Reduktion von Efficient Domination

In diesem Abschnitt wird nun gezeigt, dass sich die Suche nach einem efficient dominating set in einem Graphen reduziert werden kann auf die Suche nach einer dominierenden Menge mit minimalem Gewicht und auf die Suche nach einer unabhängigen Menge mit maximalem Gewicht im Quadrat des Graphen.

Der Kern der Reduktionen ist die Gewichtung der Knoten. Dafür sei eine Gewichtsfunktion $\omega$ wie folgt definiert: Jeder Knoten $v$ erhält als Gewicht die Anzahl der Knoten in seiner abgeschlossenen Nachbarschaft.

$$\omega(v) := |N[v]|$$

Um eine kürzere Schreibweise zu ermöglichen, seien zusätzlich für eine Knotenmenge $D \subseteq V$ deren Gewicht $\omega(D)$ und deren Nachbarschaft $N[D]$ definiert.

$$\omega(D) := \sum_{v \in D} \omega(v) = \sum_{v \in D} |N[v]|$$
$$N[D] := \bigcup_{v \in D} N[v]$$

### 4.5.1 Reduktion auf Weighted Domination

Es seien $G = (V, E)$ ein Graph und $D \subseteq V$ eine Knotenmenge in $G$.

**Lemma 4.1** Wenn $D$ dominierend ist, dann gilt: $\omega(D) \geq |V|$.

**Beweis**
Angenommen $\omega(D) < |V|$. Das bedeutet, dass die Anzahl der Knoten in der Nachbarschaft $N[D]$ von $D$ kleiner ist, als die Anzahl der Knoten im Graphen. Somit muss es einen Knoten geben, der nicht in $N[D]$ liegt. Also kann $D$ keine dominierende Menge sein.

□

**Lemma 4.2** Ist $D$ ein efficient dominating set, dann gilt: $\omega(D) = |V|$.

**Beweis**
Aufgrund von Lemma 4.1 ist $\omega(D) \geq |V|$. Es sei nun angenommen, dass $\omega(D) > |V|$ ist. Somit wird ein Knoten $u$ für die Gewichtsfunktion mehrfach gezählt. Das bedeutet, dass $u$ in der Nachbarschaft von mehreren Knoten aus $D$ liegt. Also kann $D$ auch kein efficient dominating set sein.

□

**Satz 4.2** Eine Knotenmenge $D$ ist ein efficient dominating set genau dann, wenn $D$ ein minimum weight dominating set mit $\omega(D) = |V|$ ist.

**Beweis**
**$\Rightarrow$:** $D$ ist ein efficient dominating set. Aufgrund von Lemma 4.2 ist $\omega(D) = |V|$. Es sei nun angenommen, dass $D$ nicht minimal ist. Dann existiert eine dominierende Menge $D'$ mit $\omega(D') < |V|$. Dies steht im Widerspruch zu Lemma 4.1.

**$\Leftarrow$:** $D$ ist eine dominierende Menge mit minimalem Gewicht, wobei $\omega(D) = |V|$ ist. Es sei nun angenommen, dass $D$ kein efficient dominating set ist. Dann gibt es zwei Knoten $u$ und $v$ in $D$ ($u, v \in D$, $u \neq v$), die einen gemeinsamen Nachbarn $w$ haben ($w \in N[u] \cap N[v]$). Somit wird $w$ für das Gesamtgewicht von $D$ mehrfach gezählt. Also ist $\omega(w) > |V|$. Dies steht im Widerspruch zur Voraussetzung.

□

### 4.5.2 Reduktion auf Weighted Independent Set

Es seien $G = (V, E)$ ein Graph, $N[v]$ die abgeschlossene Nachbarschaft von $v$ in $G$ und $\mathcal{I}$ ein maximum weight independent set in $G^2$.

**Lemma 4.3** Für alle Knoten $i, j \in \mathcal{I}$ $(i \neq j)$ gilt: $N[i] \cap N[j] = \emptyset$.

**Beweis**
Angenommen, es existiert ein Knoten $v \in V \setminus \mathcal{I}$ und zwei Knoten $i, j \in \mathcal{I}$ $(i \neq j)$ mit $v \in N[i] \cap N[j]$. Dann ist der Abstand von $i$ zu $j$ höchstens 2. Somit ist $ij$ eine Kante in $G^2$. Also ist $\mathcal{I}$ kein independent set in $G^2$. Dies steht im Widerspruch zur Voraussetzung.

□

Aufgrund von Lemma 4.3 gilt nun:

$$\omega(\mathcal{I}) \leq |V|$$

**Satz 4.3** $\mathcal{I}$ ist ein efficient dominating set in $G$ genau dann, wenn $\omega(\mathcal{I}) = |V|$ ist.

**Beweis**
**⇒:** $\mathcal{I}$ ist ein efficient dominating set. Aufgrund von Lemma 4.2 ist $\omega(\mathcal{I}) = |V|$.

**⇐:** $\omega(\mathcal{I}) = |V|$

Angenommen, $\mathcal{I}$ ist nicht dominierend in $G$. Dann existiert ein Knoten $v \in V$, für den es keinen Knoten $i \in \mathcal{I}$ gibt mit $v \in N[i]$. Aufgrund von Lemma 4.3 ist somit $\omega(\mathcal{I}) < |V|$.

Daraus folgt, dass $\mathcal{I}$ dominierend in $G$ ist. Aufgrund von Satz 4.2 ist $\mathcal{I}$ somit auch ein efficient dominating set.

□

### 4.5.3 Zusammenfassung

Das oben Bewiesene lässt sich nun wie folgt zusammenfassen:

**Satz 4.4** Gegeben seien ein Hypergraph $H$, dessen Linegraph $\mathcal{L} = L(H) = (V, E)$ sowie eine Gewichtsfunktion $\omega(v \in V) := |N[v]|$, wobei $N[v]$ die abgeschlossene Nachbarschaft von $v$ in $\mathcal{L}$ ist.

Für eine Menge $M \subseteq V$ sind folgenden Aussagen äquivalent:

(i) $M$ ist dominating induced Matching in $H$

(ii) $M$ ist efficient dominating set in $\mathcal{L}$

(iii) $M$ ist minimum weight dominating set in $\mathcal{L}$ mit $\omega(M) = |V|$

(iv) $M$ ist maximum weight independent set in $\mathcal{L}^2$ mit $\omega(M) = |V|$

Kapitel 5

# Algorithmen

Dieses Kapitel stellt einige Algorithmen vor, mit denen sich das Dominating Induced Matching Problem für azyklische Hypergraphen lösen lässt. Dabei wird dieses jedoch nicht direkt gelöst, sondern wie in Abschnitt 4.5 gezeigt, auf die Suche nach einer dominierenden Menge bzw. einem independent set reduziert.

Das Kapitel beginnt mit der Überprüfung, ob ein Hypergraph $\alpha$-azyklisch ist. Die nachfolgenden Abschnitte befassen sich dann lediglich mit den entsprechenden Linegraphen. Es wird der Einfachheit halber davon ausgegangen, dass der Hypergraph $\beta$- bzw. $\gamma$-azyklisch war, wenn der Linegraph strongly chordal bzw. distanzerblich ist.

## 5.1 Überprüfung des Hypergraphen

Um zu überprüfen, ob ein Hypergraph $\alpha$-azyklisch ist, bieten sich verschiedene Möglichkeiten an. Nachfolgend werden drei Varianten vorgestellt.

### 5.1.1 Mittels Definition

Eine Variante ist es, schlicht zu überprüfen, ob ein gegebener Hypergraph die in Definition 3.9 (S. 21) gestellten Bedingungen erfüllt. In diesem Fall müsste der 2-Section-Graph chordal sein und jede (maximale) Clique des 2-Section-Graphen auch eine Hyperkante darstellen.

Wie bereits in Abschnitt 2.6 (S. 6) erwähnt, lässt sich in Linearzeit überprüfen, ob ein Graph chordal ist. In [39] wird dazu ein Verfahren vorgestellt, bei dem auch eine perfekte Eliminationsordnung ermittelt wird (siehe auch Abschnitt 5.4.3). Mit dieser lassen sich dann auch die maximalen Cliquen ermitteln und mit den Hyperkanten vergleichen.

### 5.1.2 Mittels Graham-Reduktion

Eine weitere Möglichkeit bietet die Graham-Reduktion (siehe Definition 3.10, S. 22). Aus ihrer Definition ergibt sich direkt ein Algorithmus, um zu testen, ob ein Hypergraph $\alpha$-azyklisch ist. Dazu entfernt man so lange entsprechende Knoten und Hyperkanten, bis nur noch eine leere Kante übrig bleibt, oder weder Knoten noch Kanten entfernt werden können.

### 5.1.3 Mittels des Algorithmus von Tarjan und Yannakakis

Ein Möglichkeit, die sich auch in Linearzeit implementieren lässt, wird in [42] vorgestellt.

Der erste Schritt des Verfahren ist eine von $i = 1$ bis $i = k$ aufsteigende Nummerierung der Knoten und Hyperkanten, wobei sich der Wert von $k$ erst während der Nummerierung ergibt. Die dabei vergebenen Nummern seien der $\beta$-Wert der Hyperkante bzw. des Knotens. Allerdings muss nicht jede Hyperkante $e$ eine Nummer erhalten. In diesem Fall ist $\beta(e)$ nicht definiert.

Für die Nummerierung wird zunächst eine Hyperkante $e$ gewählt. Nun werden sowohl $e$ als auch alle unnummerierten Knoten $v \in e$ mit $i$ nummeriert ($\beta(e) = \beta(v) = i$). Anschließend wird $i$ um eins erhöht ($i := i + 1$) und die nächste Hyperkante gewählt. Als nächste Hyperkante wird dabei eine Hyperkante gewählt, die noch unnummerierte Knoten enthält und deren Anzahl an nummerierten Knoten maximal ist. Dies wird nun so lange wiederholt, bis alle Knoten nummeriert wurden.

Neben dem $\beta$-Wert erhält eine Hyperkante $e$ auch einen $\gamma$-Wert ($\gamma(e)$). Dieser ist wie folgt definiert:

$$\gamma(e) = \begin{cases} \max\{\beta(v) \mid v \in e\} & \beta(e) \text{ ist nicht definiert} \\ \text{nicht definiert} & \forall\, v \in e : \beta(v) = \beta(e) \\ \max\{\beta(v) \mid v \in e, \beta(v) < \beta(e)\} & \text{sonst} \end{cases}$$

In Abbildung 5.1 wurden für zwei Beispiel-Hypergraphen die $\beta$- und $\gamma$-Werte mit dem oben beschriebenen Verfahren vergeben.

Mit Hilfe der $\beta$- und $\gamma$-Werte lässt sich nun überprüfen, ob ein Hypergraph

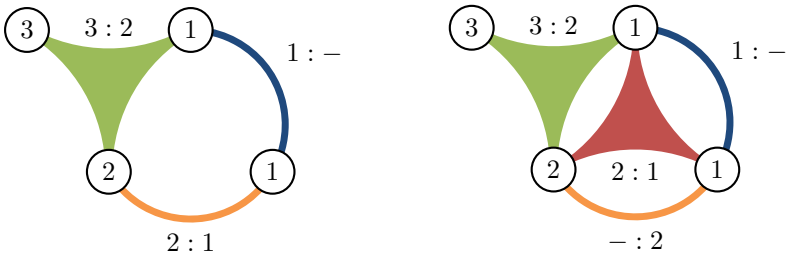



**Abbildung 5.1:** Beispiel für die $\beta$- und $\gamma$-Werte in Hypergraphen: Die Zahlen in den Knoten sind deren $\beta$-Werte. Die erste Zahl einer Hyperkante ist ihr $\beta$- und die zweite ihr $\gamma$-Wert. (Beispiel entnommen aus [42])

$\alpha$-azyklisch ist.

**Satz 5.1** [42] Es seien $H = (V, \mathcal{E})$ ein Hypergraph und $e_i \in \mathcal{E}$ die Hyperkante mit dem $\beta$-Wert $i$ ($\beta(e_i) = i$).

$H$ ist $\alpha$-azyklisch genau dann, wenn gilt:

$$\forall\, i \in \{1, \ldots, k\} : \forall\, e \in \mathcal{E}\,(\gamma(e) = i) : e \cap \{v \mid \beta(v) < i\} \subseteq e_i$$

Aus Satz 5.1 leitet sich nun direkt ein Algorithmus ab. Dieser lässt sich so modifizieren, dass er in Linearzeit läuft.

## 5.2 $\gamma$-azyklische Hypergraphen

Die Linegraphen $\gamma$-azyklischer Hypergraphen sind distanzerblich chordal (Satz 3.7, S. 26). Für diesen Abschnitt ist dabei die Distanzerblichkeit die interessante Eigenschaft. In [17] wird ein Algorithmus angegeben, mit dem sich in Linearzeit überprüfen lässt, ob ein gegebener Graph distanzerblich ist.

Interessant ist die Distanzerblichkeit eines Graphen deswegen, weil sie die Cliquenweite beschränkt.

**Satz 5.2** [28] Die Cliquenweite von distanzerblichen Graphen ist höchstens 3. Der dazugehörige Cliquenweite-Ausdruck lässt sich in Linearzeit ermitteln.

Ist die Cliquenweite einer Graphenklasse beschränkt und ist zusätzlich für einen Graphen der entsprechende Cliquenweite-Ausdruck gegeben, so lässt sich ein Problem für diesen Graphen in Linearzeit lösen, wenn es in monadischer Prädikatenlogik zweiter Stufe (engl.: monadic second-order logic) definiert werden kann [15]. Prädikatenlogik zweiter Stufe erweitert die Prädikatenlogik erster Stufe um die Möglichkeit, dass auch Prädikate an Quantoren gebunden werden können. Beschränkt man sich zusätzlich auf einstellige Prädikate, dann spricht man von monadischer Prädikatenlogik zweiter Stufe. Eine ausführliche Betrachtung des Themas wird es voraussichtlich in [16] geben.

Gelingt es also, die Suche nach einem efficient dominating set in monadischer Prädikatenlogik zweiter Stufe zu formulieren, dann lässt sich diese Suche auch in Linearzeit für distanzerbliche Graphen abschließen.

## 5.3 $\beta$-azyklische Hypergraphen

Ist der gegebene Hypergraph $\beta$-azyklisch, so ist sein Linegraph strongly chordal. Dies schließt auch die Möglichkeit ein, dass der Linegraph ein Intervallgraph ist.

### 5.3.1 Intervallgraphen

Das Überprüfen, ob ein Graph ein Intervallgraph ist, ist in Linearzeit möglich. In [6], [30] und [34] sind dazu Algorithmen angegeben. Außerdem wird in [14] ein Linearzeit-Algorithmus vorgestellt, mit dem sich ein effcient dominating set für einen Intervallgraphen ermitteln lässt.

### 5.3.2 Strongly chordale Graphen

Im Fall der strongly chordalen Graphen ist bisher kein Algorithmus bekannt, der eine Erkennung in Linearzeit ermöglicht. Zwei der schnellsten Algorithmen laufen in $\mathcal{O}(|E| \log |V|)$ [37] bzw. $\mathcal{O}(|V|^2)$ [41]. Der nächste Schritt ist das Ermitteln einer strong perfect elimination ordering. Der zeitliche Aufwand hierfür liegt bei $\mathcal{O}(|V|^3)$ [1]. Mit Hilfe einer strong perfect elimination ordering lässt sich nun ein minimum weight independent dominating set in Linearzeit berechnen [25]. Aufgrund von Satz 4.4 lässt sich somit auch überprüfen, ob der gegebene Graph ein efficient dominating set besitzt.

## 5.4 $\alpha$-azyklische Hypergraphen

Ist für einen Hypergraphen nur bekannt, dass er $\alpha$-azyklisch ist, so lässt sich der Linegraph auch lediglich auf die Klasse der dually chordalen Graphen eingrenzen.

### 5.4.1 Erkennen von dually chordalen Graphen

Prinzipiell können auch Hypergraphen, die nicht $\alpha$-azyklisch sind, einen dually chordalen Graphen als Linegraphen besitzen. Somit wäre es sinnvoll die Klasse des Linegraphen statt des Hypergraphen zu ermitteln.

In [8] wird gezeigt, dass in Linearzeit getestet werden kann, ob ein Graph dually chordal ist. Allerdings wird dabei der Graph in einen Hypergraphen umgewandelt und getestet, ob der Hypergraph $\alpha$-azyklisch ist. Bei dieser Umwandlung wird der sogenannte *Nachbarschaftshypergraph* gebildet.

**Definition 5.1 (Nachbarschaftshypergraph)**
Gegeben sei ein Graph $G = (V, E)$. Dessen *Nachbarschaftshypergraph* $\mathcal{N}(G) = (V, \mathcal{E})$ ist wie folgt definiert:

$$\mathcal{E} = \{N[v] \mid v \in V\}$$

**Satz 5.3** [8] Ein Graph $G$ ist dually chordal genau dann, wenn sein Nachbarschaftshypergraph $\mathcal{N}(G)$ $\alpha$-azyklisch ist.

Da die Größe des Nachbarschaftshypergraphen linear mit der Größe des ursprünglichen Graphen wächst, lässt sich somit in Linearzeit testen, ob ein Graph dually chordal ist.

### 5.4.2 Minimum Weight Dominating Set

Um nun ein efficient dominating set in einem dually chordalen Graphen $G$ zu finden, bietet Satz 4.4 prinzipiell zwei Möglichkeiten. Eine davon ist das Finden einer dominierenden Menge mit minimalem Gewicht in $G$. Diese Variante erweist sich allerdings als ungeeignet.

**Satz 5.4** Das Ermitteln eines minimum weight dominating set ist NP-vollständig für dually chordalen Graphen.

**Beweis**
Gegeben sei ein beliebiger Graph $G = (V, E)$. Es wird nun ein Graph $G^* = (V^*, E^*)$ wie folgt erstellt:

$$V^* = \{v^*\} \cup V$$
$$E^* = \{v^* v \mid v \in V\} \cup E$$

Offensichtlich ist $G^*$ dually chordal ($v^*$ ist maximaler Nachbar für alle Knoten). Eine Gewichtsfunktion $\omega$ sei nun wie folgt definiert:

$$\omega(v) = \begin{cases} |V| + 1 & \text{wenn } v = v^* \\ 1 & \text{sonst} \end{cases}$$

Es ist nun jede dominierende Menge $D \subseteq V$ für $G$ auch eine dominierende Menge in $G^*$. Außerdem ist das Gesamtgewicht von $D$ immer kleiner als das des Knotens $v^*$. Somit ist $D$ die kleinste dominierende Menge in $G$ genau dann, wenn $D$ die dominierende Menge mit dem kleinsten Gewicht in $G^*$ ist.

□

### 5.4.3 Quadrat dually chordaler Graphen

Neben der Suche nach einer dominierenden Menge mit minimalem Gewicht bietet Satz 4.4 noch die Option, ein independent set mit maximalem Gewicht im Quadrat eines Graphen zu suchen.

**Satz 5.5** [7] Das Quadrat eines dually chordalen Graphen $G$ ist chordal. Außerdem ist eine maximale Nachbarschaftsordnung für $G$ eine perfekte Eliminationsordnung für $G^2$.

Da das Quadrat des vorliegenden Linegraphen chordal ist, ist es nun möglich, ein independent set zu ermitteln. In [26] wird dafür ein Linearzeit-Algorithmus vorgestellt. Voraussetzung für diesen ist eine perfekte Eliminationsordnung.

### Ermitteln einer perfekten Eliminationsordnung

Es gibt verschiedene Möglichkeiten, eine perfekte Eliminationsordnung für chordale Graphen zu ermitteln. Einige Algorithmen mit linearer Laufzeit werden in [38], [39] und [42] vorgestellt. Ein weiteres Verfahren, mit dem sich jede perfekte Eliminationsordnung finden lässt, ist in [40] beschrieben.

Auch Satz 5.5 bietet einen Ansatz zum Finden einer perfekten Eliminationsordnung. Da es sich bei dem gegebenen chordalen Graphen um das Quadrat eines dually chordalen Graphen $G$ handelt, kann auch eine maximale Nachbarschaftsordnung für $G$ ermittelt werden. In [7] wird dafür der folgende Algorithmus vorgestellt.

**Algorithmus 5.1 (MNO, [7])**
**Eingabe:** Ein dually chordaler Graph $G = (V, E)$.
**Ausgabe:** Eine maximale Nachbarschaftsordnung $(v_1, \ldots, v_n)$ von $G$.

(1) Initialisiere alle Knoten $v \in V$ als unnummeriert und unmarkiert.

(2) Wähle einen beliebigen Knoten $v \in V$. Nummeriere $v$ mit $n$ (also $v_n = v$) und setze $mn(v) := v$.

(3) **Repeat**

  (a) Wähle aus allen unmarkierten Knoten einen nummerierten Knoten $u$, so dass $N[u]$ eine maximale Anzahl an nummerierten Knoten enthält.

  (b) Nummeriere alle unnummerierten Knoten $x$ aus $N[u]$ fortlaufend mit der höchstmöglichen Nummer zwischen 1 und $n - 1$, die noch frei ist. Setze außerdem $mn(x) := u$.

  (c) Markiere $u$.

**Until** Alle Knoten sind nummeriert.

Die so ermittelte maximale Nachbarschaftsordnung für $G$ ist nun auch eine perfekte Eliminationsordnung für $G^2$. Außerdem weist der Algorithmus jedem Knoten einen maximalen Nachbarn zu. Beides wird sich später noch als hilfreich erweisen (siehe Abschnitt 5.4.4). Abbildung 5.2 zeigt die Anwendung des Algorithmus an einen Beispielgraphen.

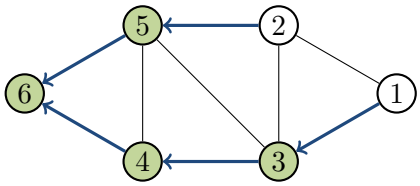



**Abbildung 5.2:** Beispiel für Algorithmus 5.1. Die Nummerierung der Knoten ist die vom Algorithmus vergebene Nummerierung. Markierte Knoten sind grün hinterlegt. Die blauen Pfeile zeigen jeweils auf den maximalen Nachbarn eines Knotens (ausgenommen Knoten 6). Der Algorithmus beginnt mit Knoten 6 und wählt die Knoten danach in folgender Reihenfolge aus: $(6, 4, 5, 3)$. Es sind allerdings auch andere Reihenfolgen möglich. (Graph entnommen aus [26])

### Ermitteln eines Maximum Weight Independent Sets

Da nun eine perfekte Eliminationsordnung gegeben ist, lässt sich nun auch ein maximum weight independent set mittels des in [26] vorgestellten Algorithmus berechnen. Ein Beispiel für die Anwendung des Algorithmus ist in Abbildung 5.3 gegeben.

**Algorithmus 5.2 (mwIS, [26])**
**Eingabe:** Ein chordaler Graph $G = (V, E)$ mit perfekter Eliminationsordnung $(v_1, \ldots, v_n)$ und einer Gewichtsfunktion $\omega$.
**Ausgabe:** Ein maximum weight independent set $\mathcal{I}$.

(1) $\mathcal{I} := \emptyset$

(2) **For** $i := 1$ **To** $n$

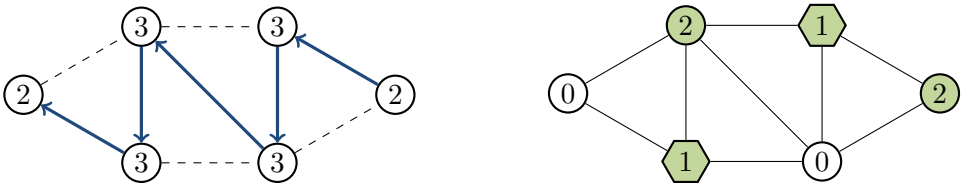
Wenn $\omega(v_i) > 0$ ist, dann markiere $v_i$ und setze $\omega(u) := \max(\omega(u) - \omega(v_i), 0)$ für alle Knoten $u \in N(v_i)$.

(3) **For** $i := n$ **DownTo** $1$

Wenn $v_i$ markiert ist, dann setze $\mathcal{I} := \mathcal{I} \cup \{v_i\}$ und entferne von allen Knoten $u \in N(v_i)$ die Markierung.

## 5.4.4 Vermeiden des Quadrats

Ein Problem beim Bilden des Quadrats ist es, dass sich damit auch die Anzahl der Kanten quadriert. Gleiches gilt somit auch für die Laufzeit. Ziel dieses Abschnitts ist es nun, auf die Bildung des Quadrats zu verzichten,




**(a)** Der Graph $G$ mit Gewichten, Nachbarschaftsordnung (blaue Pfeile) und den Kanten in $G^2$ (gestrichelt)

**(b)** $G^2$ nach dem Algorithmus mit markierten Knoten (grün), gewählten Knoten (sechseckig) und relevante Gewichte.

**Abbildung 5.3:** Beispiel für Algorithmus 5.2. Der Algorithmus wird auf den Graphen $G^2$ angewendet (blaue und gestrichelte Kanten in (a)). In (b) ist $G^2$ nach dem Algorithmus dargestellt. Die Beschriftung der Knoten ist das Gewicht der Knoten, wenn es überprüft wird (negative Gewichte sind als 0 dargestellt). Vom Algorithmus in der ersten Schleife markierte Knoten sind grün hinterlegt. In der zweiten Schleife gewählte Knoten sind zusätzlich sechseckig.

um eine Laufzeit zu erreichen, die linear zur Größe des Linegraphen ist.

### Reihenfolge der Knoten

Ein Algorithmus, der auf das Quadrat des Graphen verzichtet, kann durch Modifikation von Algorithmus 5.2 erreicht werden. Der erste Punkt dabei ist die perfekte Eliminationsordnung. Wie bereits erwähnt, ist hierfür Satz 5.5 hilfreich. Eine maximale Nachbarschaftsordnung für einen dually chordalen Graphen $G$ ist auch eine perfekte Eliminationsordnung für dessen Quadrat. Sie kann in Linearzeit mit Algorithmus 5.1 ermittelt werden.

### Verrechnung der Gewichte

Der nächste Punkt ist das Subtrahieren des Gewichts des aktuellen Knotens $v_i$ von dessen Nachbarn $N(v_i)$. Das Problem dabei ist, dass die Nachbarschaften in $G$ und $G^2$ unterschiedlich sind. Eine einfache Lösung wäre es, nicht nur die direkten Nachbarn von $v_i$ zu ermitteln sondern auch die Nachbarn der Nachbarn. Dies führt jedoch wieder zu einer quadratischen Laufzeit. Es kann allerdings ausgenutzt werden, dass $v_i$ immer einen maximalen Nachbarn $m_i$ besitzt.

**Satz 5.6** Es seien $G = (V, E)$ ein Graph, $G^2 = (V, E^2)$ dessen Quadrat und $(v_1, \ldots, v_n)$ eine maximale Nachbarschaftsordnung für $G$, wobei $m_i$ der maximale Nachbar von $v_i$ ist. Außerdem seien $1 \leq i < j \leq n$ und $v_i \neq m_i$. Für alle Knoten $v_j$ gilt nun:

$$v_i v_j \in E^2 \Leftrightarrow m_i v_j \in E$$

**Beweis**
**⇐:** $v_j$ ist mit dem maximalen Nachbarn von $v_i$ verbunden ($m_i v_j \in E$). Somit ist der Abstand von $v_i$ zu $v_j$ in $G$ höchstens 2. Also sind $v_i$ und $v_j$ auch in $G^2$ miteinander verbunden ($v_i v_j \in E^2$).

**⇒:** $v_i$ und $v_j$ sind miteinander in $G^2$ verbunden ($v_i v_j \in E^2$). Es kann nun o. B. d. A. angenommen werden, dass $v_i v_j \notin E$ ist. Folglich existiert ein Knoten $v_k$, der mit $v_i$ und $v_j$ verbunden ist ($v_i v_k, v_k v_j \in E$). Nun gilt es zwei Fälle zu unterscheiden:

(i) $i < k$. Per Definition ist der maximale Nachbar $m_i$ mit allen Nachbarn von $v_k$ verbunden. Somit auch mit $v_j$.

(ii) $k < i$. Da $v_k$ vor $v_i$ in der Nachbarschaftsordnung liegt, besitzt $v_k$ einen maximalen Nachbarn $m_k$, der sowohl mit $v_i$ also auch $v_j$ verbunden ist. Somit kann die Fallunterscheidung für $m_k$ wiederholt werden.

Per Voraussetzung ist ein Knoten nie sein maximaler Nachbar (außer $v_n$). Daraus folgt, dass sich Fall (ii) nun so lange wiederholt, bis $i < k$ ist und Fall (i) eintritt.

□

Anmerkung: Damit Satz 5.6 gültig ist, darf (bis auf den letzten Knoten) kein Knoten sein eigener maximaler Nachbar sein. Allerdings erfüllt eine von Algorithmus 5.1 berechnete Nachbarschaftsordnung diese Bedingung.

Satz 5.6 erlaubt es nun, Algorithmus 5.2 so zu ändern, dass die Gewichte auch in Linearzeit korrekt berechnet werden. Dazu erhält jeder Knoten $v$ neben seinem eigenen Gewicht $\omega$ ein zusätzliches Gewicht $\omega_p$, welches mit dem Wert 0 initialisiert wird. Dieses Zusatzgewicht dient nun als Zwischenspeicher.

Das Gewicht eines Knotens wird nun nicht mehr direkt abgezogen, sondern erst in dessen maximalen Nachbarn zwischengespeichert. Von dort wird es dann zu einem späteren Zeitpunkt weiterverteilt. Dies stellt sicher, dass über jede Kante höchstens zweimal ein Gewicht „wandert“ (je einmal $\omega$ und $\omega_p$).

#### Sperrung von Nachbarn

Der letzte Punkt beim Modifizieren von Algorithmus 5.2 ist das Sperren von Knoten, die zu nahe an einem gewählten Knoten liegen. In Algorithmus 5.2 werden dazu (in der ersten Schleife) Knoten als Kandidaten markiert. Wird nun in der zweiten Schleife ein Knoten gewählt, werden anschließend die Markierungen sämtlicher Nachbarn entfernt. Sie fallen damit als Kandidaten für die gesuchte Knotenmenge weg.

Auch in diesem Fall würde es zu quadratischer Laufzeit führen, wenn man die Nachbarn der Nachbarn ermittelt. Das Problem lässt sich dadurch lösen, dass man eine zusätzliche Markierung einführt. Für die Markierung eines Knotens $v$ gibt es dann drei Möglichkeiten: $v$ ist unmarkiert, $v$ ist Kandidat oder $v$ ist gesperrt.

Ein Knoten $v$ wird nun zur Lösungsmenge hinzugefügt, wenn er als Kandidat markiert ist und weder $v$ noch einer seiner Nachbarn gesperrt wurden. Außerdem werden in diesem Fall alle Nachbarn als gesperrt markiert.

### 5.4.5 Zusammenfassung

Das oben Beschriebene führt nun zu folgendem Algorithmus, der ein efficient dominating set für dually chordale Graphen berechnet (falls es existiert):

**Algorithmus 5.3 (dcED)**
**Eingabe:** Ein dually chordaler Graph $G = (V, E)$.
**Ausgabe:** Ein efficient dominating set $D$ (falls existent).

(1) $D := \emptyset$

(2) **For All** $v \in V$

$\omega(v) := |N[v]|$

$\omega_p(v) := 0$

(3) Ermittle eine maximale Nachbarschaftsordnung $(v_1, \ldots, v_n)$ sowie die dazugehörigen maximalen Nachbarn $(m_1, \ldots, m_n)$ mittels Algorithmus 5.1.

(4) **For** $i := 1$ **To** $n$

(a) Ziehe das Gewicht $\omega_p(u)$ aller Knoten $u \in N[v_i]$ vom Gewicht $\omega(v_i)$ ab $(\forall\, u \in N[v_i] : \omega(v_i) := \omega(v_i) - \omega_p(u))$.

(b) Wenn $\omega(v_i) > 0$ ist, dann markiere $v_i$ als Kandidat und setze $\omega_p(m_i) := \omega_p(m_i) + \omega(v_i)$.

(5) **For** $i := n$ **DownTo** 1

Wenn $v_i$ als Kandidat markiert ist und kein Knoten $u \in N[v_i]$ gesperrt ist, dann setze $D := D \cup \{v_i\}$ und sperre alle Knoten $u \in N(v_i)$.

(6) $D$ ist ein efficient dominating set genau dann, wenn $\sum_{v \in D} |N[v]| = |V|$.

**Satz 5.7** Algorithmus 5.3 findet ein efficient dominating set für dually chordale Graphen in Linearzeit.

Die Korrektheit ergibt sich aus der oberen Argumentation (besonders Satz 5.6). Daher wird an dieser Stelle auf einen Beweis verzichtet.

### 5.4.6 Der gewichtete Fall

Mit Algorithmus 5.3 ist es nun möglich, ein efficient dominating set für einen dually chordalen Graphen zu finden. Somit auch ein dominating induced Matching für $\alpha$-azyklische Hypergraphen. Das Problem ist damit jedoch nur für den ungewichteten Fall gelöst.

Angenommen, für die Kanten eines Hypergraphen bzw. die Knoten des Linegraphen sei auch eine Gewichtsfunktion $\alpha$ gegeben. Es soll nun ein dominating induced Matching mit maximalem Gewicht ermittelt werden. Da für das ursprüngliche Problem bereits eine Knotenmenge mit maximalem Gewicht gesucht wird, müssen die Gewichte nur angepasst

werden. Nachfolgend werden zwei Varianten dafür vorgestellt.

### Multiplikator-Variante

Eine Möglichkeit, um sowohl die Anzahl der Nachbarn als auch das gegebene Gewicht einer Hyperkante (bzw. Knotens $v$ im Linegraphen) zu berücksichtigen, ist es, die Summe von beiden zu bilden. Dabei wird die Anzahl der Nachbarn allerdings vorher mit einem hinreichend großen Multiplikator $m$ verrechnet.

$$\omega(v) = m \cdot N[v] + \alpha(v)$$

Die Variante hat zwei Nachteile. Zum einen muss ein Multiplikator gewählt werden, der groß genug ist, damit die Anzahl der Nachbarn die dominante Größe bleibt. Zum anderen können negative Gewichte dazu führen, dass sich am Gesamtgewicht der gefundenen Knotenmenge nicht mehr direkt ablesen lässt, ob sie dominierend ist.

### Vektor-Variante

Alternativ zu einem Gesamtgewicht, kann man die Gewichte als Vektoren betrachten. Ein solches Vektorgewicht besteht aus dem ursprünglichen Gewicht und der Anzahl der Nachbarn.

$$\omega(v) = \begin{pmatrix} N[v] \\ \alpha(v) \end{pmatrix}$$

Allerdings ist es notwendig, zu definieren, wann ein Vektor größer ist als ein anderer.

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \Leftrightarrow a = x \wedge b = y$$
$$\begin{pmatrix} a \\ b \end{pmatrix} > \begin{pmatrix} x \\ y \end{pmatrix} \Leftrightarrow a > x \vee (a = x \wedge b > y)$$

Der Vorteil besteht darin, dass die Anzahl der Nachbarn immer vom ursprünglichen Gewicht getrennt ist. Somit muss kein Multiplikator errechnet werden. Außerdem lässt sich am Gesamtgewicht sofort ablesen, ob es sich um eine dominierende Menge handelt.

$$D \subseteq V \text{ ist ein efficient dominating set } \Leftrightarrow \sum_{d \in D} \omega(d) > \begin{pmatrix} |V| \\ -\infty \end{pmatrix}$$

Kapitel 6

# Abschluss

Diese Arbeit untersuchte das Dominating Induced Matching Problem. Dabei stellte sich heraus, dass es sich (unabhängig von der Graphenklasse) auf die gewichtete Suche nach einer dominierenden Menge im Linegraphen bzw. auf die Suche nach einer gewichteten unabhängigen Menge im Quadrat des Linegraphen reduzieren lässt (Satz 4.4, S. 39).

Darauf basierend konnte gezeigt werden, dass sich das Problem für die Klasse der $\alpha$-azyklischen Hypergraphen in Polynomialzeit lösen lässt. Dazu wurde ein Algorithmus entwickelt, der in Linearzeit ein efficient dominating set für dually chordale Graphen findet (Algorithmus 5.3, S. 50), welches die Linegraphen der $\alpha$-azyklischen Hypergraphen sind (Satz 3.5, S. 25).

Offen bleibt die Frage, ob es in Linearzeit möglich ist, ein dominating induced Matching für $\alpha$-azyklischen Hypergraphen zu finden. Die im vorherigen Kapitel vorgestellte Lösung kann quadratische Laufzeit erreichen, da der Linegraph des Hypergraphen gebildet werden muss. Für eine Lösung mit linearem Aufwand muss das vermieden werden.

Da Satz 4.4 unabhängig von einer Graphenklasse ist, stellen sich außerdem die Fragen: Gibt es weitere Graphen- oder Hypergraphenklassen, für die Satz 4.4 zu einer Lösung führt? Gibt es Klassen, für welche die Suche nach einer dominierenden Menge im Linegraphen eine bessere Lösung bringt als die Suche nach einer unabhängigen Menge im Quadrat des Linegraphen?

# Literaturverzeichnis


[1] Anstee, R.P ; Farber, Martin: Characterizations of totally balanced matrices. In: *Journal of Algorithms* 5 (1984), Nr. 2, S. 215–230

[2] Bandelt, Hans-Jürgen ; Mulder, Henry M.: Distance-hereditary graphs. In: *Journal of Combinatorial Theory, Series B* 41 (1986), Nr. 2, S. 182–208

[3] Beeri, Catriel ; Fagin, Ronald ; Maier, David ; Yannakakis, Mihalis: On the Desirability of Acyclic Database Schemes. In: *Journal of the ACM* 30 (1983), S. 479–513

[4] Berge, Claude: *North-Holland mathematical library*. Bd. 45: *Hypergraphs: combinatorics of finite sets*. North Holland, 1989

[5] Booth, Kellogg S. ; Johnson, J. H.: Dominating Sets in Chordal Graphs. In: *SIAM Journal on Computing* 11 (1982), Nr. 1, S. 191–199

[6] Booth, Kellogg S. ; Lueker, George S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. In: *Journal of Computer and System Sciences* 13 (1976), Nr. 3, S. 335–379

[7] Brandstädt, Andreas ; Chepoi, Victor D. ; Dragan, Feodor F.: The algorithmic use of hypertree structure and maximum neighbourhood orderings. In: *Discrete Applied Mathematics* 82 (1998), Nr. 1–3, S. 43–77

[8] Brandstädt, Andreas ; Dragan, Feodor ; Chepoi, Victor ; Voloshin, Vitaly: Dually Chordal Graphs. In: *SIAM Journal on Discrete Mathematics* 11 (1998), Nr. 3, S. 437–455

[9] Brandstädt, Andreas ; Hundt, Christian ; Nevries, Ragnar: Efficient Edge Domination on Hole-Free Graphs in Polynomial Time.

In: López-Ortiz, Alejandro (Hrsg.): *LATIN 2010: Theoretical Informatics* Bd. 6034. Springer Berlin / Heidelberg, 2010, S. 650–661

[10] Brandstädt, Andreas ; Le, Van B. ; Spinrad, Jeremy P.: *Graph classes: a survey.* Society for Industrial and Applied Mathematics, 1999 (SIAM monographs on discrete mathematics and applications)

[11] Brandstädt, Andreas ; Mosca, Raffaele: Dominating Induced Matchings for $P_7$-free Graphs in Linear Time. In: Asano, Takao (Hrsg.) ; Nakano, Shin-ichi (Hrsg.) ; Okamoto, Yoshio (Hrsg.) ; Watanabe, Osamu (Hrsg.): *Algorithms and Computation* Bd. 7074. Springer Berlin / Heidelberg, 2011, S. 100–109

[12] Cardoso, Domingos M. ; Korpelainen, Nicholas ; Lozin, Vadim V.: On the complexity of the dominating induced matching problem in hereditary classes of graphs. In: *Discrete Applied Mathematics* 159 (2011), S. 521–531

[13] Chain-Chin, Yen ; Lee, R.C.T.: The weighted perfect domination problem and its variants. In: *Discrete Applied Mathematics* 66 (1996), Nr. 2, S. 147–160

[14] Chang, Gerard ; Pandu Rangan, C. ; Coorg, Satyan: Weighted independent perfect domination on cocomparability graphs. In: Ng, K. (Hrsg.) ; Raghavan, P. (Hrsg.) ; Balasubramanian, N. (Hrsg.) ; Chin, F. (Hrsg.): *Algorithms and Computation* Bd. 762. Springer Berlin / Heidelberg, 1993, S. 506–514

[15] Courcelle, B. ; Makowsky, J. A. ; Rotics, U.: Linear Time Solvable Optimization Problems on Graphs of Bounded Clique-Width. In: *Theory of Computing Systems* 33 (2000), S. 125–150

[16] Courcelle, Bruno ; Engelfriet, Joost: *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach.* Cambridge University Press, 2012 (Encyclopedia of Mathematics and its Applications 138). – noch nicht erschienen

[17] Damiand, Guillaume ; Habib, Michel ; Paul, Christophe: A simple paradigm for graph recognition: application to cographs and distance hereditary graphs. In: *Theoretical Computer Science* 263 (2001), Nr. 1–2, S. 99–111

[18] Dirac, G.: On rigid circuit graphs. In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 25 (1961), S. 71–76

[19] Eppstein, David: *A distance-hereditary graph.* http://commons.wikimedia.org/w/index.php?oldid=26354539. Version: Mai 2010. – [Online; Stand 3 Januar 2012]

[20] Eppstein, David: *An interval graph.* http://commons.wikimedia.org/w/index.php?oldid=36410506. Version: 2010. – [Online; Stand 27 Dezember 2011]

[21] Fagin, Ronald: Degrees of acyclicity for hypergraphs and relational database schemes. In: *Journal of the ACM* 30 (1983), Juli, S. 514–550

[22] Farber, Martin: *Applications of linear programming duality to problems involving independence and domination*, Rutgers University, Diss., 1981

[23] Farber, Martin: Independent domination in chordal graphs. In: *Operations Research Letters* 1 (1982), Nr. 4, S. 134–138

[24] Farber, Martin: Characterizations of strongly chordal graphs. In: *Discrete Mathematics* 43 (1983), Nr. 2-3, S. 173–189

[25] Farber, Martin: Domination, independent domination, and duality in strongly chordal graphs. In: *Discrete Applied Mathematics* 7 (1984), Nr. 2, S. 115–130

[26] Frank, Andras: Some polynomial algorithms for certain graphs and hypergraphs. In: *Proceedings of the Fifth British Combinatorial Conference 1975*. Utilitas Mathematica Publishing, 1976 (Congressus Numerantium XV), S. 211–226

[27] Garey, M.R. ; Johnson, D.S.: *Computers and Intractability: a Guide to the Theory of NP-completeness*. W. H. Freeman, 1979 (Series of books in the mathematical sciences)

[28] Golumbic, Martin ; Rotics, Udi: On the Clique-Width of Perfect Graph Classes. In: Widmayer, Peter (Hrsg.) ; Neyer, Gabriele (Hrsg.) ; Eidenbenz, Stephan (Hrsg.): *Graph-Theoretic Concepts in Computer Science* Bd. 1665. Springer Berlin / Heidelberg, 1999, S. 135–147

[29] Grinstead, Dana L. ; Slater, Peter J. ; Sherwani, Naveed A. ; Holmes, Nancy D.: Efficient Edge Domination Problems in Graphs. In: *Information Processing Letters* 48 (1993), Nr. 5, S. 221–228

[30] Habib, Michel ; McConnell, Ross ; Paul, Christophe ; Viennot, Laurent: Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. In: *Theoretical Computer Science* 234 (2000), Nr. 1–2, S. 59–84

[31] Howorka, Edward: A characterization of distance-hereditary graphs. In: *The Quarterly Journal of Mathematics* 28 (1977), Nr. 4, S. 417–420

[32] J., Gerard ; Chang: The weighted independent domination problem is NP-complete for chordal graphs. In: *Discrete Applied Mathematics* 143 (2004), Nr. 1-3, S. 351–352

[33] Karp, R.: Reducibility among combinatorial problems. In: Miller, R. (Hrsg.) ; Thatcher, J. (Hrsg.): *Complexity of Computer Computations*. Plenum Press, 1972, S. 85–103

[34] Korte, Norbert ; Möhring, Rolf H.: An Incremental Linear-Time Algorithm for Recognizing Interval Graphs. In: *SIAM Journal on Computing* 18 (1989), Nr. 1, S. 68–81

[35] Lu, Chin L. ; Ko, Ming-Tat ; Tang, Chuan Y.: Perfect edge domination and efficient edge domination in graphs. In: *Discrete Applied Mathematics* 119 (2002), Nr. 3, S. 227–250

[36] Lu, Chin L. ; Tang, Chuan Y.: Solving the weighted efficient edge domination problem on bipartite permutation graphs. In: *Discrete Applied Mathematics* 87 (1998), Nr. 1–3, S. 203–211

[37] Paige, Robert ; Tarjan, Robert E.: Three Partition Refinement Algorithms. In: *SIAM Journal on Computing* 16 (1987), Nr. 6, S. 973–989

[38] Panda, B.S.: New linear time algorithms for generating perfect elimination orderings of chordal graphs. In: *Information Processing Letters* 58 (1996), Nr. 3, S. 111–115

[39] Rose, Donald J. ; Tarjan, R.Endre ; Lueker, George S.: Algorithmic Aspects of Vertex Elimination on Graphs. In: *SIAM Journal on Computing* 5 (1976), S. 266–283

[40] Shier, D.R.: Some aspects of perfect elimination orderings in chordal graphs. In: *Discrete Applied Mathematics* 7 (1984), Nr. 3, S. 325–331

[41] Spinrad, Jeremy P.: Doubly lexical ordering of dense 0–1 matrices. In: *Information Processing Letters* 45 (1993), Nr. 5, S. 229–235

[42] Tarjan, Robert E. ; Yannakakis, Mihalis: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. In: *SIAM Journal on Computing* 13 (1984), S. 566–579

# Abbildungsverzeichnis

# Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Hilfsmittel verfasst habe.

Rostock, den 25. April 2012