Karlsruhe Institute of Technology

Minimum-Contamination-Probleme mit Bezug zu Graphclustern

Bachelorarbeit
von

Tobias Fleck

An der Fakultät für Informatik
Institut für Theoretische Informatik (ITI)
Lehrstuhl Algorithmen I

Erstgutachter:	Prof. Dr. rer. nat. Dorothea Wagner
Zweitgutachter:	Prof. Dr. rer. nat. Peter Sanders
Betreuender Mitarbeiter:	Dipl.-Inform. Andrea Kappes

Bearbeitungszeit: 11. Juli 2013 – 10. November 2013

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, 10. November 2013

.....

(**Tobias Fleck**)

Zusammenfassung

Minimum-Contamination-Probleme treten im Bereich der Schadsoftwarebekämpfung in Netzwerken auf. Wird ein Netzwerkknoten kontaminiert, das heißt mit einem Virus oder ähnlichem infiziert, soll durch Blockieren von Kanten die durchschnittliche Ausbreitung des Virus minimiert werden. Minimum-Contamination-Probleme lassen sich als Graph-Probleme beschreiben und treten auch als Teilprobleme im Bereich des Clusterns von Graphen auf. Das Clustern von Graphen beschäftigt sich damit stark zusammenhängende Substrukturen, so genannte Cluster eines Graphen, zu definieren und zu finden. Ein bekannter Ansatz besteht darin eine Zielfunktion zu optimieren, die jeder möglichen Clusterung eines Graphen einen Wert zuweist. Das wohl populärste Beispiel für eine solche Zielfunktion ist die von Newman und Girvan vorgestellte Zielfunktion Modularity[NG04]. Eine zu Modularity alternative Zielfunktion stellt das von Aldecoa und Marín vorgestellte Maß Surprise dar, insbesondere, da Surprise-optimale Clusterungen Vorteile gegenüber Clusterungen haben, die durch Modularity-Optimierung entstehen [AM11, AM13].

Diese Arbeit widmet sich der theoretischen Analyse von Surprise-optimalen Clusterungen und der Lösung von Minimum-Contamination-Problemen auf verschiedenen Klassen von Graphen. Für Pfade werden Algorithmen mit polynomieller Laufzeit, sowohl zur Lösung des Wertproblems zu Surprise, als auch zur Lösung von Minimum-(Average)-Contamination-Problemen angegeben. Nachdem Verbindungen zwischen den Bereichen der Minimum-Contamination-Probleme und dem Graphclustern durch Modularity- bzw. Surprise-Optimierung aufgezeigt werden, wird ein dynamischer Algorithmus angegeben, der Surprise auf Bäumen durch das Lösen des gewichteten Minimum-Average-Contamination Problems optimiert. Abschließend wird die $\mathcal{NP}$*-Schwere* von Minimum-Average-Contamination-Problemen für planare Graphen gezeigt.

Danksagung

Mein Dank gilt in erster Hinsicht Andrea Kappes für ihre Geduld und ihr enormes zeitliches Engagement. Sie stand mir sowohl bei unseren wöchentlichen Treffen, als auch außerhalb dieser Zeiten mit Ideen, guten Ratschlägen und Korrekturen zur Seite.

Außerdem möchte ich Frau Prof. Dr. Wagner für die Möglichkeit danken, meine Bachelorarbeit an ihrem Lehrstuhl zu verfassen. Für die Bewertung dieser Arbeit bedanke ich mich ebenfalls bei Frau Prof. Dr. Wagner, sowie bei Prof. Dr. Peter Sanders.

Ein letzter Dank gilt meinen Eltern, die mich zu jedem Zeitpunkt meines Studiums unterstützt haben.

Inhaltsverzeichnis

1. Einleitung

Minimum-Contamination-Probleme treten im Bereich der Schadsoftwarebekämpfung in Netzwerken auf [LT11, KSM08]. Ist ein Netzwerkknoten mit einem Virus infiziert worden, stellt sich die Frage, wie eine großräumige Ausbreitung des besagten Virus, mittels Abtrennen von Verbindungen zwischen Netzwerkknoten, verhindert werden kann. Unabhängig von dieser praktischen Annäherung können Minimum-Contamination-Probleme als Probleme der Graphentheorie aufgefasst werden. Bei den in dieser Bachelorarbeit betrachteten *Minimum-Average-Contamination-Problemen* geht es darum, die Summe der Quadrate der Gewichte entstehender Zusammenhangskomponenten zu minimieren, wenn eine feste Anzahl an Kanten eines gegebenen Graphen gelöscht wird. Hauptaugenmerk dieser Arbeit ist die theoretischen Analyse von Minimum-Contamination-Problemen, wenn für den gegebenen Graphen Einschränkungen gelten, er beispielsweise planar ist oder Baumstruktur hat.

Graph Clustering befasst sich mit dem Finden von stark in Beziehung stehenden Substrukturen, so genannten Clustern, eines Graphen. Es besitzt viele Anwendungsgebiete, die vom Extrahieren von Communities aus sozialen Netzwerken [NG04, AMM05], bis hin zum Aufteilen von Programmcode auf Hardwareressourcen reichen [KM77, ACSG01]. Motiviert durch zahlreiche Anwendungsgebiete existieren eine Vielzahl von Ansätzen, um stark zusammenhängende Teilbereiche eines Graphen zu charakterisieren und zu finden. Fortunato bietet eine Übersicht über mögliche Verfahren und Ansätze [For10]. Eines der bekannteren Beispiele für eine Definition von Clustern ist das, von Newman und Girvan eingeführte, Clustern mittels Modularity-Optimierung [NG04]. Hierbei stellen Cluster disjunkte Teilmengen der Knotenmenge eines Graphen dar, für die es gilt eine Zielfunktion Modularity zu optimieren, die jeder möglichen Clusterung einen reellen Wert zuweist. Über die Optimierung von Modularity gibt es viele Arbeiten. Diese decken sowohl den Bereich der theoretischen Analyse [NG04, DT, GSW11, BDG$^+$07], als auch praktische Umsetzungen [CNM04, BGLL08] mittels Heuristiken ab.

Ebenfalls disjunkte Cluster von Knoten zugrunde legend, bietet die von Aldecoa und Marín in [AM11] eingeführte Zielfunktion Surprise eine Alternative zu Modularity. Im Gegensatz zu Modularity existieren für Surprise wenig theoretische Analysen, was Komplexität und optimale Lösungen auf speziellen Klassen von Graphen anbelangt. Aldecoa und Marín legen dar, dass das Optimieren von Surprise in vielen Fällen bessere Resultate liefert als das Suchen von Clustern mittels Modularity-Optimierung [AM11, AM13]. Obgleich diese Aussage auf der Verwendung von Heuristiken und Metaheuristiken beruht,

macht sie eine theoretische Analyse des Clusterns von Graphen mittels SURPRISE interessant. Kappes et. al. machen diesbezüglich erste Schritte und zeigen neben weiteren wichtigen Eigenschaften der Funktion, dass SURPRISE-Optimierung auf allgemeinen Graphen *$\mathcal{NP}$-vollständig* ist [FKW14]. Außerdem geben sie exakte Lösungen für kleine Graphen, zum Beispiel Bäume, an. Hierbei kommen Verbindungen zum Bereich von Minimum-Contamination-Problemen zum Tragen, da SURPRISE-Optimierung auf Bäumen durch Lösen des ungewichteten Minimum-Contamination-Problems erreicht werden kann.

1.1 Ergebnisse dieser Arbeit

In dieser Bachelorarbeit wird sowohl das gewichtete Minimum-Average-Contamination-Problem, als auch SURPRISE auf Pfaden und einfachen Kreisen untersucht. Es wird ein polynomieller Algorithmus angegeben, der das gewichtete Minimum-Average-Contamination-Problem optimal auf Pfaden und Kreisen löst, sowie eine Teilcharakterisierung einer optimalen SURPRISE-Clusterung erreicht. Darüber hinaus wird zur Lösung des gewichteten Minimum-Average-Contamination-Problems auf Bäumen ein dynamischer Algorithmus entwickelt, der im allgemeinen Fall pseudopolynomielle Laufzeit, für den Spezialfall des ungewichteten Minimum-Average-Contamination-Problems aber polynomielle Laufzeit besitzt. Dieser Algorithmus wird anschließend verwendet, um einen Algorithmus anzugeben, der das Wertproblem zu SURPRISE auf Bäumen in $O(n^5)$ Zeit löst. Der Algorithmus zur SURPRISE-Optimierung auf Bäumen erscheint zusätzlich in der Arbeit [FKW14]. Durch die Bezüge zwischen SURPRISE-Optimierung und dem ungewichteten Minimum-Average-Contamination-Problem auf Bäumen inspiriert, werden Verbindungen zwischen dem Clustern von Graphen mittel MODULARITY- und SURPRISE-Optimierung, sowie *Minimum-Average-Contamination-Problemen* erläutert und anschließend gezeigt, dass sowohl das ungewichtete, als auch das gewichtete Minimum-Average-Contamination-Problem auf planaren Graphen *$\mathcal{NP}$-schwer* ist.

1.2 Verwandte Probleme und Abgrenzung

Auf Bäumen bestehen Minimum-Average-Contamination-Probleme darin, eine feste Anzahl an Kanten des Baumes zu löschen, sodass die Summe der Quadrate der Komponentengewichte, im durch das Löschen der Kanten entstehenden Waldes, minimal wird. Ein sehr ähnliches, obgleich mit weniger Aufwand zu lösendes Problem ist das so genannte *Tree-Partitioning-Problem*. Es besteht darin aus einem gegebenen Baum eine möglichst kleine Anzahl an Kanten zu löschen, sodass alle Zusammenhangskomponenten im entstehenden Wald ein Maximalgewicht nicht überschreiten. Erstaunlich ist, dass sich das Tree-Partitioning-Problem in linearer Laufzeit [KM77] lösen lässt, wohingegen es schwierig erscheint einen Algorithmus zu finden, der das ungewichtete Minimum-Average-Contamination-Problem mit besserer Laufzeit als $O(n^5)$ löst.

Ein Problem, das ebenso wie Minimum-Contamination-Probleme aus dem Bereich der Schadsoftwarebekämpfung stammt, ist das so genannte *Sum-Of-Squares-Partition-Problem*. Der Unterschied besteht darin, anstatt Kanten ganze Knoten aus einem Netzwerk zu löschen, sodass die Summe der Quadrate der entstehenden Clustergrößen minimiert wird [ACY06]. Trotz der ähnlichen Struktur des Problems, gibt es, was die Lösbarkeit betrifft, keine offensichtlichen Verbindungen zu Minimum-Contamination-Problemen.

Ebenso ähnlich ist das *Graph-Partitionierungs-Problem(GPP)*, das darin besteht, die Knotenmenge einen Graphen in k disjunkte Teilmengen fester Größe zu zerlegen, sodass die Summe aller Kantengewichte von Kanten, die zwischen je zwei Teilmengen verlaufen, minimiert wird. Analog zu Minimum-Average-Contamination-Problemen, liegt der Schwerpunkt im Graph-Partitionierungs-Problem darin, die Knoten eines Graphen gleichmäßig

auf disjunkte Teilmengen zu verteilen. Während GPP die gleichmäßige Verteilung der Knoten als harte Vorbedingung enthält und die Anzahl, bzw. das Gewicht der zu schneidenden Kanten zwischen Clustern variabel lässt, wählen Minimum-Average-Contamination-Probleme eine feste Anzahl an Kanten, die maximal geschnitten werden dürfen, sodass die resultierenden Cluster möglichst gleichmäßige Größe haben. Das Graph-Partitionierungs-Problem tritt unter anderem im Bereich des Parallelrechnens beim Aufteilen von Ressourcen auf [For10].

Correlation Clustering stellt ebenfalls eine ähnliche Problemstellung zu Minimum-Average-Contamination-Problemen dar. Gegeben ein vollständiger Graph mit Kantenbeschriftungen " + " und " − " soll eine Clusterung des Graphen gefunden werden, sodass möglichst viele mit " + " beschriftete Kanten innerhalb von Clustern liegen und möglichst Kanten, die mit " − " beschriftet sind, zwischen Clustern verlaufen [BBC04]. Zusätzlich sollen mögliche Fehlzuweisungen, also " + "-Kanten außerhalb von Clustern und " − "-Kanten innerhalb von Clustern minimiert werden. Ebenso wie beim Problem GPP offenbart sich keine direkte Beziehung zu Minimum-Average-Contamination-Problemen.

1.3 Gliederung

In Kapitel 2 werden allgemeine Grundlagen, Notation und Definitionen eingeführt, die für diese Arbeit von Bedeutung sind. Darauf aufbauend wird anschließend die Funktion Surprise eingeführt, um dann einige wichtige Eigenschaften von Surprise zu erläutern, die nützlich sind, um Surprise-optimale Clusterungen zu finden. Kapitel 3 führt diverse Optimierungsprobleme ein. Hierunter sind sowohl Minimum-Average-Contamination-Probleme, die direkten Bezug zum Kontext von Surprise haben, als auch *Minimum-Sum-Of-Squares-of-Component-Volumes* Probleme, die dem Kontext von Modularity Clustering entsprechen. Neben deren Definition wird ein kurzer Überblick über die Komplexität der einzelnen Probleme auf verschiedenen Klassen von Graphen gegeben und versucht, die Probleme miteinander in Beziehung zu setzen. Kapitel 4 widmet sich Pfaden und betrachtet sowohl das in Kapitel 3 eingeführte gewichtete Minimum-Average-Contamination-Problem w-MACP, als auch die Optimierung von Surprise. Gegenstand von Kapitel 5 sind Minimum-Average-Contamination-Probleme eingeschränkt auf Bäume. Es wird ein dynamischer Algorithmus angegeben und analysiert, mit dem besagte Probleme in polynomieller bzw. pseudopolynomieller Laufzeit gelöst werden können. Des Weiteren wird ein Algorithmus angegeben, der durch Lösung des ungewichteten Minimum-Average-Contamination-Problems den Wert einer bezüglich Surprise optimalen Clusterung auf Bäumen berechnet. Anschließend wird in Kapitel 6 die Komplexität von Minimum-Average-Contamination-Problemen, eingeschränkt auf planare Graphen, untersucht. Zunächst wird auf einen bestehenden Beweis [LT11] zur $\mathcal{NP}$*-Schwere* des Minimum-Average-Contamination-Problems eingegangen und ein Gegenbeispiel für die Gültigkeit auf planaren Graphen angegeben. Inspiriert durch den betrachteten Beweis wird ein neuer Beweis zur $\mathcal{NP}$*-Schwere*, sowohl für das ungewichtete, als auch für das gewichtete Minimum-Average-Contamination-Problem, geführt.

2. Grundlagen und Notation

In diesem Kapitel sollen Grundlagen und Notation eingeführt werden. Zunächst werden allgemeine Definitionen erläutert, die in der gesamten Arbeit Verwendung finden, um anschließend die Funktion SURPRISE zu definieren und wichtige Eigenschaften von SURPRISE darzulegen.

2.1 Allgemeine Grundlagen und Notation

Definition 1 (Graph, knotengewichteter Graph, kantengewichteter Graph)**.** *Ein* Graph $G = (V, E)$ *besteht aus einer endlichen Knotenmenge* V *und einer ungerichteten Kantenmenge* $E \subseteq \big\{\{u, v\} \mid u, v \in V\big\}$*. Die Anzahl von Knoten sei* $n := |V|$ *und die Anzahl von Kanten sei* $m := |E|$*.*

Besitzt G *eine Gewichtsfunktion* $w : V \to \mathbb{N}$*, die jedem Knoten* $v \in V$ *ein Gewicht* $w(v)$ *zuordnet, so heißt* G knotengewichteter *oder auch nur* gewichteter Graph*.*

Für eine Funktion $c : E \to \mathbb{N}$*, die jeder Kante* $e \in E$ *ein Gewicht zuordnet, heißt* G kantengewichteter Graph*.*

Zu einem Graphen sind verschiedene Aufteilungen desselben in Substrukturen denkbar. Die folgende Definition gibt an, was in dieser Arbeit unter einem *Cluster* verstanden werden soll.

Definition 2 (Clusterung eines Graphen)**.** *Sei* $G = (V, E)$ *ein Graph und* $\zeta \subset 2^V$ *eine Partition von* G*, d.h.* $V = \bigcup_{C \in \zeta} C$ *und* $C_i \cap C_j = \emptyset$ *für alle* $C_i, C_j \in \zeta$ *mit* $i \neq j$*. Dann heißt* ζ Clusterung *von* G*.*

Bei einer Clusterung eines Graphen $G = (V, E)$ handelt es sich um eine Partitionierung der Knotenmenge, sodass jeder Knoten aus G genau einem Cluster $C \in \zeta$ zugeordnet ist. Zu einer gegebenen Clusterung lassen sich die folgenden Kennzahlen definieren.

Definition 3 (Intraclusterkanten, Intraclusterpaare, Interclusterkanten,)**.** *Sei* $G = (V, E)$ *ein Graph,* ζ *eine Clusterung von* G *und* $C \in \zeta$ *ein einzelner Cluster. Dann heißt*

- $m(C) = \big|\big\{e = \{u, v\} \in E \mid u, v \in C\big\}\big|$ *die Anzahl der der* Intraclusterkanten *in Cluster* C*.*
- $p(C) = \binom{|C|}{2} = \frac{|C|(|C|-1)}{2}$ *die Anzahl der* Intraclusterpaare *in Cluster* C*.*

- $i_e(\zeta) = \sum_{C \in \zeta} m(C)$ *die Anzahl der* Intraclusterkanten *der Clusterung* ζ.
- $i_p(\zeta) = \sum_{C \in \zeta} p(C)$ *die Anzahl der* Intraclusterpaare *der Clusterung* ζ.
- $p = \binom{n}{2} = \frac{n(n-1)}{2}$ *die Anzahl der Paare von Knoten in* G.
- $m - i_e(\zeta)$ *die Anzahl der* Interclusterkanten *der Clusterung* ζ.

Wichtige Kennzahlen im Zusammenhang mit SURPRISE sind die Anzahl der Intraclusterkanten $i_e(\zeta)$ und die Anzahl der Intraclusterpaare $i_p(\zeta)$.

Definition 4 (Komponentengewicht, Gewicht von Teilmengen)**.** *Sei* $G = (V, E)$ *ein Graph mit Knotengewichtsfunktion* $w : V \to \mathbb{N}$. *Dann ist das* Gewicht einer Teilmenge von Knoten $C \subseteq V$ *definiert als* $w(C) = \sum_{v \in C} w(v)$. *Ist* ζ *eine Clusterung, so heißt* $w(C)$ *das* Komponentengewicht *von* $C \in \zeta$.

Außerdem werden in einem Graphen die folgenden Begriffe für Nachbarschaftsbeziehungen zwischen Knoten und Kanten verwendet.

Definition 5 (Adjazenz, Inzidenz)**.** *Sei* $G = (V, E)$ *ein Graph. Zwei Knoten* $u, v \in V$ *heißen* adjazent*, wenn* $\{u, v\} \in E$ *gilt. Ein Knoten* $v \in V$ *heißt* inzident *zu Kante* $e \in E$*, wenn* $e = \{u, v\}, u \in V$ *gilt. Zwei Kanten* $e, e' \in E$ *heißen* inzident*, wenn sie einen gemeinsamen Knoten besitzen.*

Für einen Knoten eines Graphen wird außerdem die Begrifflichkeit des Knotengrades geprägt. Dieser stellt ein Maß für die Verbundenheit des Knoten zum restlichen Graphen dar.

Definition 6 (gewichteter und ungewichteter Knotengrad)**.** *Sei* $G = (V, E)$ *ein kantengewichteter Graph mit Gewichtsfunktion* $c : E \to \mathbb{N}$. *Dann bezeichnet* $grad(v) = \sum_{\{u,v\} \in E} c(\{u, v\})$ *den* gewichteten Knotengrad *von Knoten* v. *Ist* $c(e) = 1$ *für alle* $e \in E$*, so ist* $grad(v) = \left|\left\{\{u, v\} \in E\right\}\right|$ *der* Knotengrad *oder* Grad *von* v.

Werden aus einem Graphen Kanten entfernt, so zerfällt der Graph möglicherweise in mehrere Teilgraphen, die untereinander nicht über Kanten erreichbar sind. Derartige Teilgraphen werden auch als Zusammenhangskomponenten bezeichnet.

Definition 7 (Schnitt, Wert eines Schnittes)**.** *Sei* $G = (V, E)$ *ein Graph mit Kantengewichtsfunktion* $c : E \to \mathbb{N}$. *Dann heißt eine Teilmenge* $S \subseteq E$*, für die* $G' = (V, E \setminus S)$ *in mehrere Zusammenhangskomponenten zerfällt,* Schnitt *in* G. *Die* Größe *oder* Wert des Schnittes *ist gerade* $\sum_{e \in S} c(e)$.

Werden die Kanten eines Schnittes aus einem Graphen gelöscht, so zerfällt der Graph in mindestens zwei Zusammenhangskomponenten bzw. Cluster. Im Folgenden werden einige Klassen von Graphen vorgestellt, die eine spezielle Struktur haben.

Definition 8 (Pfad)**.** *Ein Graph* $P = (V, E)$ *mit* $V = \{v_1, \ldots, v_n\}$ *und* $E = \left\{\{v_i, v_{i+1}\} \mid i = 1, \ldots, n-1\right\}$ *heißt* Pfad.

Werden in einem Pfad die beiden Endknoten v_1 und v_n durch eine Kante verbunden, entsteht ein einfacher Kreis.

Definition 9 (einfacher Kreis)**.** *Ein Graph* $G = (V, E)$ *mit* $v = \{v_1, \ldots, v_n\}$ *und* $E = \left\{\{v_i, v_{i+1}\} \mid 1 \leq i \leq n-1\right\} \cup \left\{\{v_n, v_1\}\right\}$ *heißt* einfacher Kreis.

Eine Verallgemeinerung von Pfaden stellen Bäume dar.

Definition 10 (Baum). *Sei $T = (V, E)$ ein ungerichteter, zusammenhängender Graph mit $|E| = |V| - 1$. Dann heißt T Baum.*

Pfade, Kreise und Bäume stellen wiederum Spezialfälle für eine größere Klasse von Graphen, so genannte planare Graphen, dar.

Definition 11 (planarer Graph). *Ein Graph $G = (V, E)$, der in der Ebene gezeichnet werden kann, ohne dass Kreuzungen von Kanten entstehen, heißt* planar.

Eine weitere Klasse von Graphen stellen so genannte *bipartite Graphen* dar, deren Knotenmenge sich in zwei disjunkte Teilmengen aufteilen lässt, sodass Kanten nur von einer in die andere Menge verlaufen.

Definition 12 (bipartiter Graph). *Sei $G = (V, E)$ ein Graph, sodass $V = V_1 \cup V_2$ mit $V_1 \cap V_2 = \emptyset$ gilt und für jede Kante $e = \{v_1, v_2\} \in E$ gilt: $v_1 \in V_1, v_2 \in V_2$.*

2.2 Surprise

Im Folgenden wird die Problemstellung des Graphclusterns durch SURPRISE-Optimierung definiert und es werden Grundlagen für die weitere Arbeit mit SURPRISE geschaffen. SURPRISE ist eine Funktion, die jeder möglichen Clusterung eines Graphen einen reellen Wert zuordnet. Die folgende Definition von SURPRISE entstammt [AMM05].

Definition 13 (SURPRISE). *Sei $G = (V, E)$ ein Graph mit $n := |V|, m := |E|$ und ζ eine Clusterung von G. Dann heißt*

$$\mathcal{S}(\zeta) = S\big(i_e(\zeta), i_p(\zeta)\big) = \sum_{i=i_e(\zeta)}^{m} \frac{\binom{i_p(\zeta)}{i} \cdot \binom{p-i_p(\zeta)}{m-i}}{\binom{p}{m}} \tag{2.1}$$

SURPRISE-Wert *der Clusterung ζ.*

Eine anschauliche Erklärung wie die Funktion SURPRISE zu verstehen ist, liefert [FKW14] mit einer kombinatorischen Betrachtung. Jeder Summand von SURPRISE gibt die mögliche Anzahl an Graphen mit n Knoten und m Kanten an, die bezüglich Clusterung ζ genau i Intraclusterkanten und $m - i$ Interclusterkanten besitzen. Somit stellt SURPRISE insgesamt die Anzahl der Graphen mit n Knoten und m Kanten dar, die mindestens so viele Intraclusterkanten in ζ haben, wie der betrachtete Graph. Je kleiner der Wert von SURPRISE, desto besser ist eine gegebene Clusterung.

Bemerkung 1. SURPRISE *kann bei gegebenem Graphen als eine Funktion $S\big(i_e(\zeta), i_p(\zeta)\big)$ von zwei Parametern aufgefasst werden. Sie hängt ab von der Anzahl der Intraclusterpaare $i_p(\zeta)$ und Anzahl der Intraclusterkanten $i_e(\zeta)$ einer Clusterung ζ. Eine ausführlichere Erläuterung zu dieser bikriteriellen Sichtweise ist in [FKW14] enthalten.*

Basierend auf der Funktion SURPRISE lässt sich das folgende Optimierungsproblem definieren.

Optimierungsproblem 1 (Optimierungsproblem zu SURPRISE). *Sei $G = (V, E)$ ein Graph. Finde eine Clusterung ζ, sodass $\mathcal{S}(\zeta) \leq \mathcal{S}(\zeta')$ für alle Clusterungen $\zeta' \neq \zeta$ von G.*

Um das Verständnis und die Arbeit mit der Funktion SURPRISE zu erleichtern, ist es von Nutzen zu betrachten, wie sich SURPRISE bezüglich Optimalität bei Änderung einer der beiden Parameter $i_e(\zeta)$, bzw. $i_p(\zeta)$ verhält.

Lemma 1. *[FKW14]*
Sei G ein beliebiger Graph und ζ eine Clusterung von G. Dann gilt $\mathcal{S}\big(i_e(\zeta)+1, i_p(\zeta)\big) < \mathcal{S}\big(i_e(\zeta), i_p(\zeta)\big)$.

Das Beibehalten der Anzahl der Intraclusterpaare $i_p(\zeta)$, bei Erhöhen der Anzahl der Intraclusterkanten $i_e(\zeta)$, führt also zu einer besseren Clusterung des Graphen bezüglich SURPRISE. Ebenso führt das Verringern der Anzahl der Intraclusterpaare $i_p(\zeta)$ bei Festhalten der Anzahl der Intraclusterkanten $i_e(\zeta)$ zu einer Verbesserung der SURPRISE-Funktion, wie das folgende Lemma zeigt.

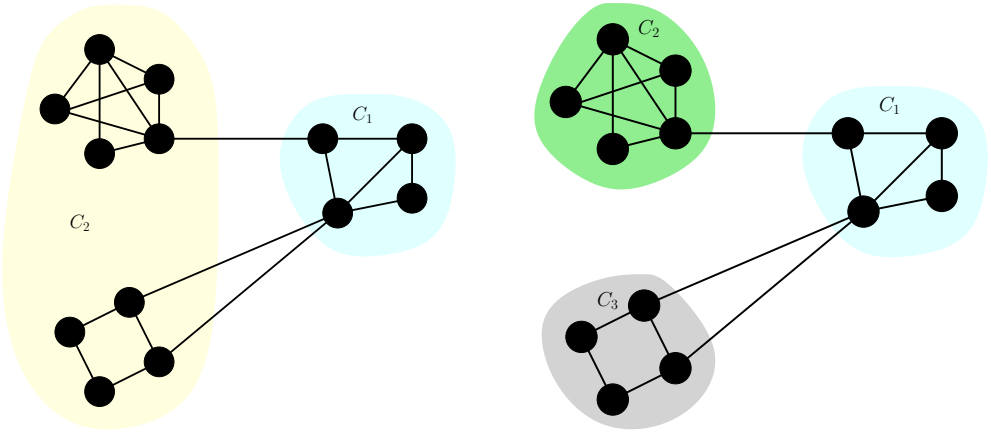
Lemma 2. *[FKW14]*
Sei G ein Graph, ζ eine Clusterung von G und $i_e(\zeta) > 0$. Dann gilt: $\mathcal{S}\big(i_e(\zeta), i_p(\zeta)-1\big) < \mathcal{S}\big(i_e(\zeta), i_p(\zeta)\big)$.

Eine wichtige Charakterisierung einer SURPRISE-optimalen Clusterung eines Graphen ist, dass alle Cluster Zusammenhangskomponenten bilden, d.h. in einem Cluster jeder Knoten von jedem anderen Knoten, über einen Pfad von paarweise adjazenten Knoten, erreichbar ist.

Satz 1 (Zusammenhang von SURPRISE-optimalen Clusterungen)**.** *Sei G ein Graph und ζ eine Clusterung, die bezüglich* SURPRISE *optimal ist. Dann bestehen alle Cluster $C \in \zeta$ jeweils aus einer Zusammenhangskomponente.*

Beweis. Sei G ein Graph und ζ eine bezüglich SURPRISE minimale Clusterung von G. Annahme: Es existiert ein Cluster $C \in \zeta$, der aus Zusammenhangskomponenten $C_1, \ldots, C_k$ besteht. Betrachte eine neue Clusterung, die durch Aufteilung der Zusammenhangskomponenten $C_1, \ldots, C_k$ von Cluster C entsteht. Während die Anzahl der Intraclusterkanten $m(C) = \sum_{1 \leq i \leq k} m(C_i)$ und somit auch die Anzahl der Intraclusterkanten $i_e(\zeta)$ in ζ gleich bleibt, sinkt die Summe der Intraclusterpaare $i_p(\zeta)$, was nach Lemma 2 zu einer besseren Clusterung führt, ein Widerspruch. Folglich kann eine SURPRISE-minimale Clusterung, in der ein Cluster aus mehr als einer Zusammenhangskomponente besteht, nicht existieren. □

Abbildung 2.1 stellt ein Beispiel zur Veranschaulichung des Beweises zu Satz 1 dar. Die Anzahl der Intraclusterkanten $i_e(\zeta)$ des Graphen bleibt gleich, während die Anzahl an Intraclusterpaaren $i_p(\zeta)$ durch Aufspalten des Clusters C_2 in zwei Cluster C_2 und C_3 sinkt.



(a) $i_e(\zeta) = 17, i_p(\zeta) = 97$ (b) $i_e(\zeta) = 17, i_p(\zeta) = 57$

Abbildung 2.1: Beispiel zum Beweis von Satz 1

3. Minimum-Average-Contamination-Probleme im Zusammenhang mit Surprise und Modularity

Dieses Kapitel befasst sich mit der Definition von Problemen, die eng im Zusammenhang mit dem Clustern von Graphen durch SURPRISE- und MODULARITY-Optimierung stehen. Zu diesen Problemen gehören unter anderem Minimum-Average-Contamination-Probleme. Darüber hinaus soll eine Übersicht über die Lösbarkeit und Komplexität der einzelnen Probleme auf verschiedenen Klassen von Graphen gegeben und erläutert werden, wie die Probleme aus den verschiedenen Kontexten zu einander in Verbindung stehen.

Optimierungsproblem 2 (MINIP)**.** *Sei $G = (V, E)$ ein Graph und $k \in \mathbb{N} > 0$ eine natürliche Zahl. Finde zu k eine Clusterung ζ von G mit $i_e(\zeta) = k$, sodass $i_p(\zeta)$ minimal ist.*

Das Problem MINIP besteht darin die Anzahl der Intraclusterpaare $i_p(\zeta)$ zu minimieren, während die Anzahl der Intraclusterkanten $i_e(\zeta)$ einen konstanten Wert k behält. Es stellt ein wichtiges Teilproblem von SURPRISE dar. Ist $G = (V, E)$ ein beliebiger Graph mit $|V| =: n$, $|E| =: m$ und A ein Algorithmus, der das Problem MINIP auf G optimal löst, so kann unter Verwendung von A, der Wert einer SURPRISE-optimalen Clusterung berechnet werden. Löse hierzu für jede Anzahl von Intraclusterkanten $0 \leq k \leq m \leq n^2$ das Problem MINIP, sodass $A(G, k)$ der optimale Wert von MINIP bezüglich k in G ist. Dann ist die Anzahl der Intraclusterkanten durch k und die Anzahl der Intraclusterpaare durch $A(G, k)$ gegeben. Der Wert einer SURPRISE-optimalen Clusterung ζ^* ergibt sich somit wie folgt:

$$\mathcal{S}(\zeta^*) = \min_{0 \leq k \leq m} \mathcal{S}(k, A(G, k)) \tag{3.1}$$

Besitzt Algorithmus A polynomielle Laufzeit, bleibt auch die Laufzeit zur SURPRISE-Optimierung polynomiell, da Algorithmus A höchstens n^2 mal ausgeführt wird. Aus dieser Beobachtung und der $\mathcal{NP}$*-Schwere* des Optimierungsproblems zu SURPRISE folgt das folgende Lemma.

Lemma 3. *Sei $\mathcal{P} \neq \mathcal{NP}$. Dann existiert kein Algorithmus, der zu einer gegebenen Instanz von* MINIP *eine Lösung in polynomieller Zeit berechnet.*

Die folgenden Minimum-Average-Contamination-Probleme W-MACP und U-MACP stammen aus dem Bereich der Bekämpfung von Schadsoftware in Netzwerken und werden in

[KSM08] vorgestellt. Als Motivation für beide Probleme kann das folgende Szenario betrachtet werden. Eine Netzwerkkomponente wird zufällig mit einem Virus oder ähnlichem infiziert, das heißt zufällig ein Knoten im Netzwerk ausgewählt. Das entstehende Problem liegt nun darin die durchschnittliche Ausbreitung des Virus über das Netzwerk, gegebenenfalls durch das Entfernen von Leitungen, zu verhindern. Eine detailliertere Beschreibung des Szenarios, sowie die formale Definition der Probleme liegt in [LT11, KSM08] vor.

Optimierungsproblem 3 (U-MACP (ungewichtetes Minimum-Average-Contamination–Problem)). *Sei $G = (V, E)$ ein Graph und $K \in \mathbb{N}$ eine positive Zahl. Finde eine Teilmenge $E' \subseteq E$ mit $|E'| = K$, sodass $\sum_{C \in \zeta} |C|^2$ für die Clusterung ζ, die durch Zusammenhangskomponenten in $G' = (V, E \setminus E')$ induziert wird, minimiert wird.*

Optimierungsproblem 4 (W-MACP (gewichtetes Minimum-Average-Contamination–Problem)). *Sei $G = (V, E)$ ein Graph, $K \in \mathbb{N}$ eine positive Zahl und $w : V \to \mathbb{N}$ eine Knotengewichtsfunktion. Finde eine Teilmenge $E' \subseteq E$ mit $|E'| = K$, sodass $\sum_{C \in \zeta} w(C)^2$ für die Clusterung ζ, die durch Zusammenhangskomponenten in $G' = (V, E \setminus E')$ induziert wird, minimiert wird.*

Die hier gegebene Definition von U-MACP und W-MACP unterscheidet sich zur Definition in [LT11] durch Weglassen eines konstanten Faktors $1/|V|$. Für die Suche nach optimalen Lösungen der Probleme ist dieser konstante Faktor allerdings nicht von Belang, weshalb an dieser Stelle darauf verzichtet wird. Das zugehörige Entscheidungsproblem zu W-MACP hat folgende Gestalt.

Entscheidungsproblem 1 (DECISION-W-MACP(Entscheidungsproblem W-MACP)). *Sei $G = (V, E)$ ein Graph mit Knotengewichtsfunktion $w : V \to R$ und $K, B \in \mathbb{N}$. Gibt es eine Clusterung ζ von G, die durch Löschen von K Kanten entsteht, sodass*

$$\sum_{C \in \zeta} w(C)^2 \leq B$$

gilt?

Anschaulich gesehen, sollen zur W-MACP-Optimierung K Kanten aus einem gegebenen Graphen G gelöscht werden, wodurch der Graph in mehrere Zusammenhangskomponenten zerfällt, die eine Clusterung ζ induzieren. Hierbei soll die Summe über die Quadrate der Gewichte der entstehenden Cluster $\sum_{C \in \zeta} w(C)^2$ in G minimiert werden. Die Probleme U-MACP und MINIP sind bezüglich Optimalität äquivalent, was im folgenden Lemma festgehalten ist.

Lemma 4. *Sei $G = (V, E)$ ein Graph. Die Probleme* MINIP *und* U-MACP *sind bezüglich Optimalität äquivalent, d.h. eine Clusterung ζ ist bezüglich* MINIP *optimal, genau dann wenn ζ auch für* U-MACP *optimal ist.*

Beweis. Sei ζ eine Clusterung eines Graphen $G = (V, E)$. Für die Anzahl der Intraclusterpaare gilt:

$$\begin{aligned} i_p(\zeta) &= \sum_{C \in \zeta} \frac{|C|\,(|C| - 1)}{2} \\ &= \frac{1}{2} \sum_{C \in \zeta} |C|^2 - \frac{1}{2} \sum_{C \in \zeta} |C| \\ &= \frac{1}{2} \underbrace{\sum_{C \in \zeta} |C|^2}_{\text{Wert von U-MACP auf G}} - \frac{1}{2} |V| \end{aligned}$$

Es folgt die Behauptung. □

Wie Li und Tang zeigen, ist das gewichtete Minimum-Average-Contamination-Problem(w-MACP) $\mathcal{NP}$-*schwer* auf bipartiten Graphen [LT11], insbesondere also auch auf allgemeinen Graphen. Auch im ungewichteten Fall bleibt das Problem auf bipartiten Graphen $\mathcal{NP}$-*schwer* [LT11]. Eigentlich zeigen Li und Tang, dass DECISION-w-MACP $\mathcal{NP}$-*vollständig* auf bipartiten, planaren Graphen ist. Was die $\mathcal{NP}$-*vollständigkeit* auf planaren Graphen angeht, ist der Beweis allerdings fehlerhaft, was in Kapitel 6 näher erläutert wird. Darüber hinaus wird in Kapitel 6 die Polynomialzeitreduktion aus [LT11] leicht verändert und anschließend gezeigt, dass die Probleme DECISION-w-MACP und DECISION-u-MACP auf planaren Graphen $\mathcal{NP}$-*vollständig* sind.

Für die praktische Anwendung ist die Approximierbarkeit von Minimum-Contamination-Problemen von Interesse. Für w-MACP lässt sich ein bikriterieller Approximationsalgorithmus angeben, der Güte $\big(1+\epsilon, O(\frac{1+\epsilon}{\epsilon}\log n)\big)$ hat [LT11]. Der Algorithmus berechnet also zu einem beliebigen Graphen G und einem $\epsilon > 0$ eine Clusterung, die durch Löschen von $O(\frac{1+\epsilon}{\epsilon}\log n)$ Kanten entsteht und Wert $(1+\epsilon)Opt(I)$ hat, wobei $Opt(I)$ der Wert einer optimalen Lösung von w-MACP auf G ist, wenn K Kanten gelöscht werden. Darüber hinaus ist das Problem, w-MACP mit einer Güte von $\frac{5}{3}-\epsilon$ zu approximieren $\mathcal{NP}$-*schwer* [LT11]. Die Ergebnisse zur Approximierbarkeit von w-MACP fußen auf der bereits beschriebenen alternativen Definition, die den optimalen Wert einer Lösung von w-MACP mit einem Faktor von $1/|V|$ multipliziert. Für Approximationsalgorithmen mit relativer Güte bedeutet dies keine Änderung, da der Faktor $1/|V|$ sowohl in der optimalen Lösung, als auch in der approximierten Lösung enthalten ist und somit das Verhältnis beider Lösungen gleich bleibt.

Ein zu Surprise ähnlicher Ansatz zur Clusterung von Graphen ist das Optimieren der Zielfunktion Modularity, die sowohl für ungewichtete [NG04], als auch für gewichtete Graphen [New04] definiert ist.

Definition 14 (Modularity)**.** *Sei $G=(V,E)$ mit $|E| =: m$ ein Graph mit Kantengewichtsfunktion $w : E \to \mathbb{N}$. Dann heißt für eine Clusterung ζ von G*

$$Q(\zeta) = \sum_{C\in\zeta}\left(\frac{w\big(E(C)\big)}{W} - \left(\frac{\sum_{v\in C} grad(v)}{2W}\right)^2\right) \tag{3.2}$$

Modularity-*Wert der Clusterung ζ, wobei $w(E(C)) = \sum_{u,v\in C,\{u,v\}\in E} w\big(\{u,v\}\big)$ und $W = \sum_{e\in E} w(e)$ ist.*

Die Funktion Modularity basiert, wie Surprise, auf so genannten *Nullmodellen* und setzt sich zum einen aus der *Coverage*, der Anzahl der Intraclusterkanten der Clusterung ζ, zum anderen aus dem Erwartungswert der *Coverage*, also der erwarteten Anzahl der Intraclusterkanten in einem Zufalls-Graphen zusammen [For10]. Je höher der Modularity-Wert einer gegebene Clusterung, desto besser ist die Clusterung. Die zu Modularity zugehörigen Optimierungsprobleme haben folgende Gestalt.

Optimierungsproblem 5 (u-Modularity)**.** *Sei $G=(V,E)$ ein ungewichteter Graph, es gelte also $w(v)=1$ für alle $v\in V$. Finde eine Clusterung ζ von G, sodass für alle Clusterungen $\zeta' \neq \zeta$ von G gilt: $Q(\zeta) \geq Q(\zeta')$.*

Optimierungsproblem 6 (w-Modularity)**.** *Sei $G=(V,E)$ ein Graph mit Kantengewichtsfunktion $w : E \to \mathbb{N}$. Finde eine Clusterung ζ von G, sodass für alle Clusterungen $\zeta' \neq \zeta$ gilt: $Q(\zeta) \geq Q(\zeta')$.*

Das Problem u-MODULARITY stellt einen Spezialfall des Problems w-MODULARITY dar. Da u-MODULARITY *$\mathcal{NP}$-schwer* ist [BDG+07], folgt die *$\mathcal{NP}$-Schwere* von w-MODULARITY unmittelbar. Analog zu MINIP aus dem Kontext von SURPRISE lassen sich die folgenden zwei Teilprobleme von u-MODULARITY und w-MODULARITY definieren, die entstehen, wenn eine feste Anzahl an Interclusterkanten betrachtet wird.

Optimierungsproblem 7 (k-MSSV (ungewichtetes Minimum Sum-of-Squares of Component Volumes))**.** *Sei $G = (V, E)$ ein Graph. Finde eine Teilmenge $E' \subseteq E$ von k Kanten, sodass deren Wegnahme aus G die Summe der Quadrate der Komponenten-Volumina $\sum_{C \in \zeta} (vol(C))^2$ minimiert. Hierbei seien $\zeta = \{C_1, \ldots, C_l\}$ die entstehenden Zusammenhangskomponenten in $G' = (V, E \setminus E')$ und $vol(C) = \sum_{v \in C} grad(v)$.*

Optimierungsproblem 8 (w-k-MSSV (gewichtetes Minimum Sum-of-Squares of Component Volumes))**.** *Sei $G = (V, E)$ ein Graph mit Kantengewichtsfunktion $w : E \to \mathbb{R}_{\geq 0}$. Finde eine Teilmenge $E' \subseteq E$ von Kanten mit $\sum_{e \in E'} w(e) = k$, sodass deren Wegnahme aus G die Summe der Quadrate der Komponenten-Volumina $\sum_{C \in \zeta} (vol(C))^2$ minimiert. Hierbei seien $\zeta = \{C_1, \ldots, C_l\}$ die entstehenden Zusammenhangskomponenten in $G' = (V, E \setminus E')$ und $vol(C) = \sum_{v \in C} grad(v)$.*

Wird das Problem k-MSSV für alle möglichen Anzahlen von Interclusterkanten $0 \leq k \leq m$ mit Wert $v^*(k)$ optimal gelöst, so kann der Wert einer u-MODULARITY-optimalen Clusterung ζ^* wie folgt berechnet werden:

$$Q(\zeta^*) = \max_{0 \leq k \leq m} \frac{1}{2m} \left((m - k) - \frac{1}{2m} v^*(k) \right) \tag{3.3}$$

Damit folgt direkt, dass das Problem k-MSSV *$\mathcal{NP}$-schwer* ist, woraus wiederum folgt, dass auch w-k-MSSV *$\mathcal{NP}$-schwer* ist, da es eine Verallgemeinerung von k-MSSV darstellt. Auf Bäumen lässt sich das Problem k-MSSV mit Laufzeit $O(n^5)$ lösen, woraus, wie im Fall von SURPRISE ein Algorithmus zur u-MODULARITY-Optimierung resultiert [DT]. Für die Verallgemeinerung w-k-MSSV besitzt der Algorithmus noch immer pseudopolynomielle Laufzeit $O(n^5 \cdot W^4)$, wobei W das größte Kantengewicht im Graphen ist [DT].

Da das Problem k-MSSV darin besteht k Kanten aus einem Graphen zu löschen, sodass die Summe über das Quadrat der Volumina der verbleibenden Zusammenhangskomponenten minimiert wird, ist es außerdem ein Spezialfall von w-MACP, wobei die Knotengewichtsfunktion gerade der Volumenfunktion entspricht. Die beschriebenen Zusammenhänge sind noch einmal schematisch in Abbildung 3.1 dargestellt und eine Übersicht über ihre Komplexität ist in Tabelle 3.1 gegeben. Neben den Weiß hinterlegten Ergebnissen aus bereits bestehenden Arbeiten, enthält Tabelle 3.1 auch die in dieser Arbeit gezeigten Ergebnisse. Diese sind in Grau hinterlegt.

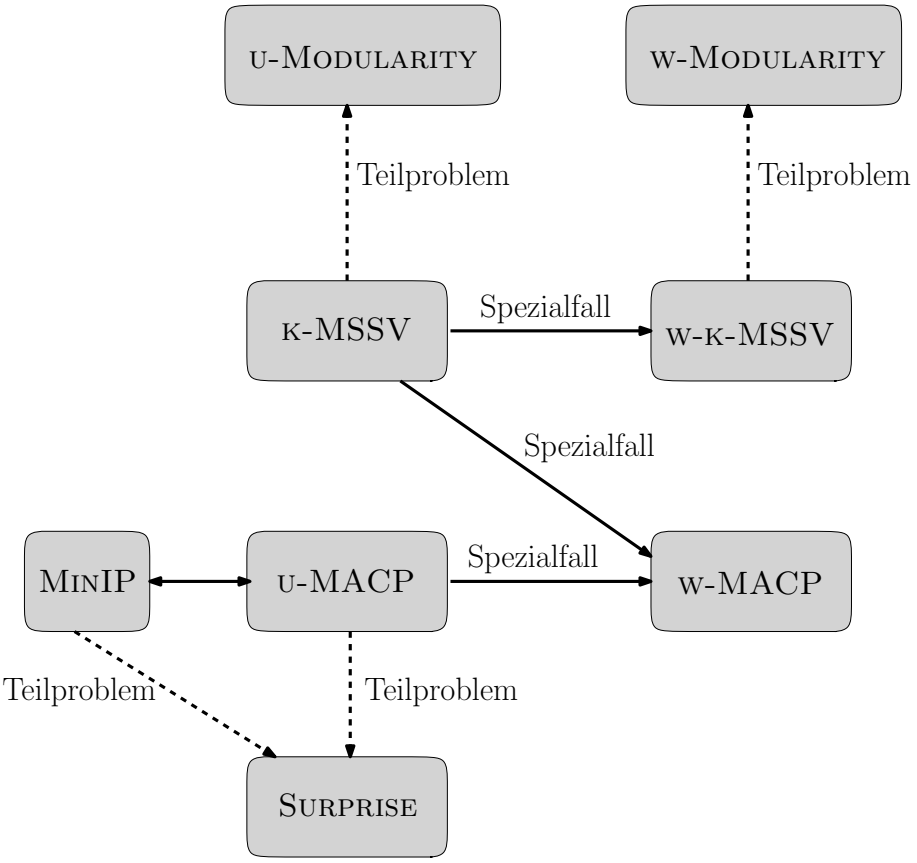


Abbildung 3.1: Zusammenhang zwischen den vorgestellten Problemen

Problem	**allgemeine Graphen**	**bipartite Graphen**	**planare Graphen**	**Bäume**	**Pfade**	**Kreise**
U-MACP	- $\mathcal{NP}$-*schwer*	- $\mathcal{NP}$-*schwer* [LT11] (ohne Beweis)	- $\mathcal{NP}$-*schwer* (Kap. 6.3)	- $O(n^5)$ (Kap. 5)	- Charakterisierung (Kap. 4.2)	Charakterisierung (Kap. 4.2)
W-MACP	- $\mathcal{NP}$-*schwer*	- $\mathcal{NP}$-*schwer* [LT11]	- $\mathcal{NP}$-*schwer* (Kap. 6.2)	- $O(n^5 \cdot w_{max}^2)$ mit maximalem Knotengewicht w_{max} (Kap. 5)	- $O(n^3)$ für alle K (Kap. 4.1) - $O(n^2 \cdot K)$ für festes K (Kap. 4.1)	- $O(n^4)$ für alle K (Kap. 4.1) - $O(n^3 \cdot K)$ für festes K (Kap. 4.1)
SURPRISE	- $\mathcal{NP}$-*schwer* [FKW14]	?	?	- $O(n^5)$ (Kap. 5)	- $O(n^3)$ (Kap. 4.2)	- $O(n^3)$ (Kap. 4.2)
U-MODULARITY	- $\mathcal{NP}$-*schwer* [BDG+07]	?	?	- $O(n^5)$ [DT]	- Charakterisierung [BDG+07]	- Charakterisierung [BDG+07]
W-MODULARITY	- $\mathcal{NP}$-*schwer*	?	- $\mathcal{NP}$-*schwer*	- $\mathcal{NP}$-*schwer* [DT] - $O(n^5 \cdot W^4)$ mit W maximales Kantengewicht [DT]		
K-MSSV	- $\mathcal{NP}$-*schwer*	?	?	- $O(n^5)$ [DT]	- $O(n^3)$ für alle k (Kap. 4.1) - $O(n^2 \cdot k)$ für festes k (Kap. 4.1)	- $O(n^4)$ für alle k (Kap. 4.1) - $O(n^3 \cdot k)$ für festes k (Kap. 4.1)
W-K-MSSV	- $\mathcal{NP}$-*schwer*	?	- $\mathcal{NP}$-*schwer*	- $\mathcal{NP}$-*schwer* [DT] - $O(n^5 \cdot W^4)$ mit W maximales Kantengewicht [DT]		

Tabelle 3.1: Übersicht der vorgestellten Probleme und der Komplexität für die Berechnung optimaler Lösungen auf verschiedenen Klassen von Graphen.

4. Pfade und einfache Kreise

4.1 w-MACP auf Pfaden und Kreisen

Im Folgenden wird das gewichtete Minimum-Average-Contamination-Problem w-MACP auf Pfaden und Kreisen untersucht. Hierbei wird ein dynamischer Algorithmus angegeben, der w-MACP mit Komplexität $O(n^3)$ löst. Sei $P = (V, E)$ ein Pfad.

Definition 15 (Teilpfad)**.** *Zu einem Pfad $P = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$ und $E = \big\{\{v_i, v_{i+1}\} \mid 1 \leq i \leq n-1\big\}$ und einem beliebigen Knoten $v_u \in V$ heißt $P(v_u) = (V(v_u), E(v_u))$ mit $V(v_u) = \{v_1, \ldots, v_u\}$ und $E(u) = \big\{\{v_i, v_{i+1}\} \mid 1 \leq i \leq u-1\big\}$* Teilpfad von P an Knoten v_u.

Es wird eine Funktion $F(v_u, l)$ eingeführt, die den Wert einer optimalen Lösung von w-MACP auf dem Teilpfad $P(v_u)$ bei Löschen von l Kanten darstellt. Abbildung 4.1a illustriert das beschrieben Szenario.

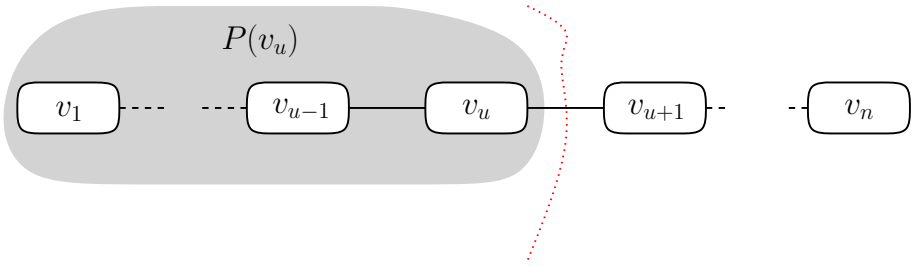
Für den Basisfall des Algorithmus werden alle Knoten $v_u \in V$ wie folgt initialisiert.

$$F(v_u, 0) = \left(\sum_{i=1}^{u} w(v_i)\right)^2 \tag{4.1}$$

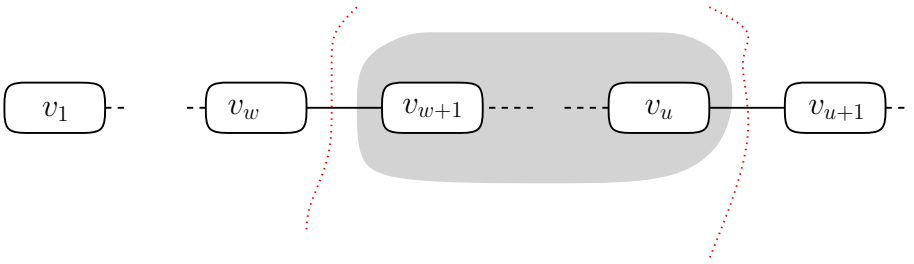
Aufbauend auf dem Basisfall lässt sich nun der Wert $F(v_u, l)$ für jeden Knoten $v_u \in V \setminus \{v_1\}$ und jedes $l > 0$ berechnen.

$$F(v_u, l) = \min_{1 \leq w < u} F(v_w, l-1) + \left(\sum_{i=w+1}^{u} w(v_i)\right)^2 \tag{4.2}$$

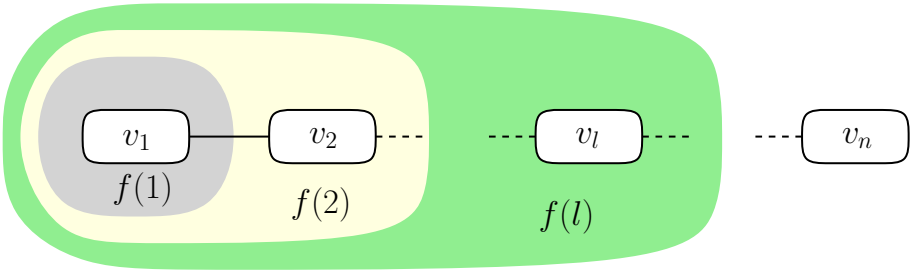
Betrachtet wird die Zusammenhangskomponente, die den Knoten v_u in einer optimalen Lösung von w-MACP für den Teilpfad $P(v_u)$ enthält. Diese werde durch Knoten v_{w+1} und v_u für $1 \leq w < u$ begrenzt. Vergleiche hierzu Abbildung 4.1b. Somit muss die Kante $\{v_w, v_{w+1}\}$ gelöscht worden sein. Der Wert einer solchen optimalen Lösung von w-MACP für den Teilpfad $P(v_u)$ ergibt sich durch das Quadrat des Gewichts der Komponente, die v_u enthält und der optimalen Lösung für Teilpfad $P(v_w)$ bei Löschen von $l-1$ Kanten, da Kante $\{v_w, v_{w+1}\}$ bereits gelöscht wurde. Unter allen möglichen Knoten v_w mit $w < u$ wird



(a) Teilpfad $P(v_u)$ zu Knoten v_u.



(b) Knoten v_{w+1} begrenzt zusammen mit v_u eine Zusammenhangskomponente.



(c) Prafixsummen der Knotengewichte.

Abbildung 4.1: Veranschaulichung von Teilpfaden und Berechnung der Funktionen F, sowie f.

Algorithmus 1 : Pfad-w-MACP

Data : Pfad $P = (V, E)$
Result : $F(v_n, K)$ für alle $1 \leq K \leq n-1$

```
f(1) = w(v_1)
F(v_1, 0) = w(v_1)^2
for j = 2 bis n do
    f(j) = f(j-1) + w(v_j)
    F(v_j, 0) = f(j)^2
for u = 1 bis n do
    for K = 0 bis |E(P(v_u))| do
        F(v_u, K) = min_{1≤w<u} { F(v_w, K-1) + (f(u) - f(w))^2 }
return F(v_n, K)
```

nun derjenige gewählt, durch den die Lösung von w-MACP auf Teilpfad $P(v_u)$ minimal ist.

Eine optimale Lösung von w-MACP des gesamten Pfades P bei Löschen von K Kanten ergibt sich durch $F(v_n, K)$. Das Gewicht der Komponente, die Knoten v_u enthält, lässt sich unter Verwendung von Präfixsummen effizienter berechnen. Hierzu dient die folgende Funktion.

$$f(j) = \sum_{i=1}^{j} w(v_i) \tag{4.3}$$

Abbildung 4.1c verdeutlicht die Berechnung der Präfixsummen. Für die Komponente, die durch Knoten v_{w+1} und v_u begrenzt ist, gilt für deren Gewicht:

$$\sum_{i=w+1}^{u} w(v_i) = f(u) - f(w) \tag{4.4}$$

Hierdurch wird eine Abfrage des Gewichts der Komponente, die durch v_{w+1} und v_u begrenzt ist, in $O(1)$ möglich, wenn $f(u)$ und $f(w)$ bereits berechnet wurden. Alle Werte $f(j)$ lassen sich gemäß folgender Gleichung in Zeit $O(n)$ vorberechnen.

$$f(j) = \begin{cases} w(v_1) & j = 1 \quad (4.5) \\ f(j-1) + w(v_j) & j > 1 \quad (4.6) \end{cases}$$

Algorithmus 1 stellt das beschriebene Verfahren noch einmal übersichtlich dar.

Lemma 5. *Für ein fest gewähltes $0 \leq K^* \leq n-1$ kann $F(v_n, K^*)$ in $O(n^2 \cdot K^*)$ berechnet werden. Die Berechnung von $F(v_n, K)$ für alle $0 \leq K \leq n-1$ ist in $O(n^3)$ Zeit möglich.*

Beweis. Die Berechnung der Präfixsummen $f(j)$ ist mit Laufzeit $O(n)$ möglich. Es gibt insgesamt $n \cdot K$ Paare von Werten $F(v_u, K)$ in P und jeder Wert dieser Paare kann durch die Verwendung der Präfixsummen $f(j)$ in $O(n)$ Zeit berechnet werden. Somit ergibt sich für eine fest gewählte Anzahl K^* von in P zu schneidenden Kanten eine Laufzeit von $O(n^2 \cdot K^*)$. Soll $F(v_n, K)$ für alle $0 \leq K \leq n-1$ berechnet werden, ist die Laufzeit durch $O(n^3)$ gegeben. □

Besitzt der gegebene Graph keine Pfadstruktur, sondern ist ein einfacher Kreis $C = (V, E)$, so kann w-MACP durch Verwendung von Algorithmus 1 gelöst werden. Wird aus C eine Kante $e = \{v_e, w_e\} \in E$ gelöscht, so stellt der Graph $P = (V, E \setminus \{e\})$ einen Pfad mit n Knoten dar. Somit lässt sich für P eine optimale Lösung $F(v_e, K-1)$ von w-MACP unter Verwendung von Algorithmus 1 in $O(n^2 \cdot K)$ Zeit berechnen, wenn Knoten w_e den Basisknoten für Algorithmus 1 bildet. Wird dieses Vorgehen für alle möglichen Kanten $e \in E$ wiederholt, so ergibt das Minimum über die Werte $F(v_e, K-1)$ für alle möglichen Kanten $\{v_e, w_e\}$ eine optimale Lösung von w-MACP auf C, die in $O(n^3 \cdot K)$ Zeit berechnet werden kann. Für die Optimierung für alle Werte von $0 \leq K \leq n$ beträgt die Laufzeit auf Pfaden folglich $O(n^4)$.

4.2 Surprise und u-MACP auf Pfaden und Kreisen

Im Folgenden werden Surprise-optimale Clusterungen auf Pfaden untersucht.

Für jeden Pfad $P = (V, E)$ gilt $|E| = |V| - 1$. Mit Satz 1 folgt direkt, dass eine Surprise-optimale Clusterung $\zeta = \{C_1, \ldots, C_k\}$ eines Pfades wieder Pfadstruktur hat. Wird jeder der k Cluster zu einem Knoten kontrahiert, ergibt sich wieder ein Pfad mit $k-1$ Kanten. Das heißt die Clusterung ζ des ursprünglichen Pfades hat genau $k-1$ Interclusterkanten und $n-k$ Intraclusterkanten.

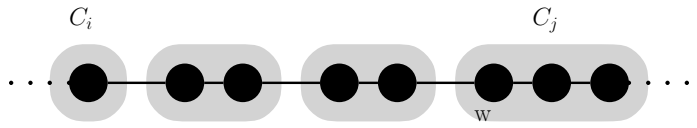
Satz 2 (Balanciertheit von Surprise-optimalen Clusterungen auf Pfaden)**.** *Sei P ein Pfad und $\zeta = \{C_1, \ldots, C_k\}$ eine bezüglich* Surprise *optimale Clusterung von P. Dann gilt: $\big|\, |C_i| - |C_j| \,\big| \leq 1$ für alle $i \neq j$.*

Beweis. Annahme: Es existieren zwei Cluster C_i, C_j, sodass $|C_i| + 1 \leq |C_j| - 1$. Wähle den Knoten $w \in C_j$, der auf dem Pfad am nächsten zu C_i liegt, entferne ihn aus Cluster C_j und füge ihn zu Cluster C_{j-1}, dem nächsten Cluster auf dem Pfad von C_j nach C_i, hinzu. Führe diesen Schritt iterativ fort und wähle für Cluster C_l mit $i < l < j$ denjenigen Knoten, der auf dem Pfad am nächsten zu C_i liegt, entferne ihn aus Cluster C_l und füge ihn zu Cluster C_{l-1} hinzu. Dieses Vorgehen ist in Abbildung 4.2 abgebildet.

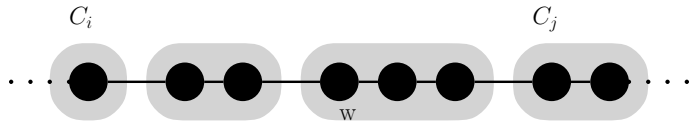
Durch die beschriebene Vorgehensweise ändert sich die Anzahl der Intraclusterkanten $i_e(\zeta)$ nicht. Die Anzahl der Intraclusterkanten in allen Clustern C_l mit $l \neq i, j$ bleibt erhalten, wohingegen die Anzahl der Intraclusterkanten in C_j um eine Kante sinkt, dagegen in C_i um eine Kante steigt. Zu zeigen bleibt, dass durch das "Verschieben" des Knotens w die Anzahl der Intraclusterpaare $i_p(\zeta)$ sinkt. Offensichtlich bleibt die Anzahl der Intraclusterpaare in allen Cluster C_l mit $l \neq i, j$ erhalten, weshalb es genügt zu zeigen, dass die Anzahl von Intraclusterpaaren in C_i und C_j insgesamt sinkt.

Sei $n_i := |C_i|$, $n_j := |C_j|$.

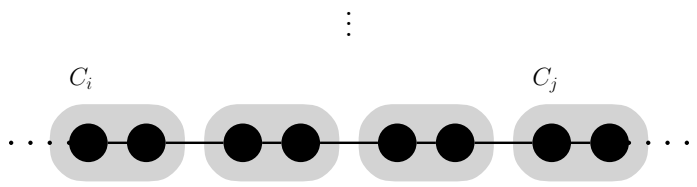
$$
\begin{aligned}
p(C_i \cup \{w\}) + p(C_j \setminus \{w\}) &= \frac{(n_i+1)^2 - (n_i+1)}{2} + \frac{(n_j-1)^2 - (n_j-1)}{2} \\
&= \frac{n_i^2 + 2n_i + 1 - n_i - 1}{2} + \frac{n_j^2 - 2n_j + 1 - n_j + 1}{2} \\
&= \frac{n_i^2 - n_i}{2} + \frac{n_j^2 - n_j}{2} + \frac{2n_i - 2n_j + 2}{2} \\
&= \frac{n_i^2 - n_i}{2} + \frac{n_j^2 - n_j}{2} + \underbrace{n_i - n_j + 1}_{<0} \\
&< \frac{n_i^2 - n_i}{2} + \frac{n_j^2 - n_j}{2} = p(C_i) + p(C_j)
\end{aligned}
$$



(a) Ursprüngliche Clusterung; nicht optimal.



(b) Clusterung nach erster Verschiebung.



(c) Clusterung, die bezüglich SURPRISE besser ist, als ursprüngliche Clusterung.

Abbildung 4.2: Zum Beweis von Satz 2

Insgesamt sinkt die Anzahl der Intraclusterpaare in der Clusterung, was nach Lemma 2 zu einer bezüglich SURPRISE besseren Clusterung führt, ein Widerspruch. Somit können die angenommenen Cluster C_i und C_j nicht existieren und es gilt die Behauptung. □

Der optimale Wert einer Clusterung mittels SURPRISE muss also nur unter Clusterungen gesucht werden, die balanciert sind. Wird eine feste Anzahl $1 \leq k \leq n$ von Clustern gewählt, so müssen in jedem Cluster mindestens $\lfloor n/k \rfloor$ Knoten enthalten sein. Ausgehend von dieser Aufteilung werden die verbleibenden Knoten gleichmäßig auf beliebige Cluster aufgeteilt. Die Anzahl der Cluster mit $\lfloor n/k \rfloor + 1$ Knoten ist dann $n \mod k$, die Anzahl der Cluster mit $\lfloor n/k \rfloor$ Knoten gerade $k - (n \mod k)$. Für die Gesamtzahl von Intraclusterpaaren in einer balancierten Clusterung gilt deshalb:

$$i_p(\zeta) = (n \mod k) \cdot \binom{\lfloor \frac{n}{k} \rfloor}{2} + \big(k - (n \mod k)\big) \cdot \binom{\lfloor \frac{n}{k} \rfloor + 1}{2} \tag{4.7}$$

Dies entspricht einer optimalen Lösung des Problems MINIP auf P, wobei die Anzahl der Intraclusterkanten gerade $n-k$ ist. Folglich muss, um den Wert einer optimalen Clusterung bezüglich SURPRISE zu berechnen, das Problem

$$\min_{1 \leq k \leq n} \mathcal{S}(\zeta) = \frac{1}{\binom{\binom{n}{2}}{n-1}} \sum_{i=n-k}^{n-1} \binom{(n \mod k) \cdot \binom{\lfloor \frac{n}{k} \rfloor}{2} + \big(k - (n \mod k)\big) \cdot \binom{\lfloor \frac{n}{k} \rfloor + 1}{2}}{i} \cdot \binom{\binom{n}{2} - (n \mod k) \cdot \binom{\lfloor \frac{n}{k} \rfloor}{2} + \big(k - (n \mod k)\big) \cdot \binom{\lfloor \frac{n}{k} \rfloor + 1}{2}}{n-1-i} \tag{4.8}$$

gelöst werden. Durch analytisches Lösen von Gleichung (4.8) wären SURPRISE-optimale Clusterungen auf Pfaden vollständig charakterisiert. Da sich dies schwierig gestaltet, löst Algorithmus 2 das Problem algorithmisch.

Lemma 6. *Algorithmus 2 hat eine Laufzeit von $O(n^3)$.*

Algorithmus 2 : PFAD-SURPRISE

Data : Pfad $P = (V, E)$
Result : $S_{min}(P)$, der minimale SURPRISE-Wert von P

```
1 S_min(P) = ∞
2 S_k = ∞
3 for k = 1 bis n do
4     i_p(k) = (n mod k) · (⌊n/k⌋ choose 2) + (k − (n mod k)) · (⌊n/k⌋ choose 2)+1
5     S_k = S(k, i_p(k))
6     if S_k < S_min(P) then
7         S_min(P) = S_k
8 return S_min(P)
```

Beweis. Da es $O(k)$ Zeit braucht, um einen Binomialkoeffizienten auszurechnen, wird zur Berechnung eines SURPRISE-Wertes für beliebiges aber festes $i_e(\zeta) = k-1$ gerade $O(n \cdot k)$ Zeit benötigt, da über alle Anzahlen von Intraclusterkanten zwischen $k-1$ und $n-1$ iteriert werden muss. Wird dieser Schritt für alle möglichen Anzahlen von Cluster $1 \leq k \leq n$ durchgeführt, ergibt sich eine Gesamtlaufzeit von $O(n^3)$, da zusätzlich $k \leq n$ gilt. □

Ist der gegebene Graph kein Pfad, sondern ein einfacher Kreis, ergeben sich nur marginale Änderungen, die gezeigten Aussagen bleiben im Wesentlichen bestehen. Insbesondere verliert Satz 2 nicht an Gültigkeit. Für jede Clusterung mit k Clustern eines Kreises ist die Anzahl der Interclusterkanten durch k und die Anzahl der Intraclusterkanten durch $n-k$ gegeben. Der Beweis von Satz 2 für Kreise erfolgt dann analog, zu dem für Pfade. Auch die Suche nach SURPRISE-optimalen Clusterung auf Kreisen lässt sich analog durchführen.

Da die Probleme MINIP und U-MACP bezüglich optimalen Lösungen äquivalent sind, folgt direkt, dass Satz 2 auch für U-MACP bestand hat. Hieraus lässt sich sofort eine Charakterisierung einer optimalen Lösung von U-MACP auf P angeben. Werden K Kanten gelöscht, so enthält die induzierte Clusterung $K+1$ Cluster und es gilt für eine optimale Lösung $Opt(P, K)$ von U-MACP:

$$\begin{aligned} Opt(P,K) =& \big(n \mod (K+1)\big) \left(\left\lfloor \frac{n}{K+1} \right\rfloor \right)^2 \\ &+ \Big(K+1 - \big(n \mod (K+1)\big)\Big) \left(\left\lfloor \frac{n}{K+1} \right\rfloor + 1 \right)^2 \end{aligned} \tag{4.9}$$

Ist der gegebene Graph kein Pfad, sondern ein einfacher Kreis, so entstehen in einer Lösung von U-MACP bei Löschen von K Kanten auch K Cluster. Die Berechnung einer optimalen Lösung erfolgt dann analog.

5. Bäume

In diesem Kapitel soll ein dynamischer Algorithmus zur Optimierung von w-MACP auf ungerichteten Bäumen angegeben werden. Der Algorithmus orientiert sich an einem in [DT] vorgestellten Algorithmus, der das Problem k-MSSV für alle $0 \leq k \leq n-1$ löst, um Modularity auf ungewichteten Bäumen zu optimieren. Der von Dinh und Thai in [DT] angegebene Algorithmus lässt sich in kanonischer Weise auf allgemeine Knotengewichte erweitern, indem statt der Volumenfunktion zur Initialisierung des Algorithmus die Gewichte der allgemeinen Knotengewichtsfunktion benutzt werden.

Wie Lemma 4 besagt, sind Surprise und u-MACP bezüglich Optimalität äquivalent. Somit ist klar, dass der hier vorgestellte Algorithmus auch zur Lösung des Problems Surprise benutzt werden kann, was in Abschnitt 5.2 detaillierter erläutert wird.

Clusterungen auf Bäumen, die bezüglich Surprise optimal sind, haben eine spezielle Struktur, die es erleichtert diese Clusterungen zu charakterisieren. Da die Cluster nach Satz 1 zusammenhängend sind, bilden sie, wenn die einzelnen Cluster als Knoten auffasst werden, wieder einen Baum mit insgesamt $l-1$ Kanten, wobei l die Anzahl der Cluster ist. Für das Optimierungsproblem zu Surprise auf Bäumen und damit für w-MACP bedeutet dies folgendes.

Bemerkung 2. *Eine* Surprise*-optimale Clusterung ζ eines Baumes mit l Clustern enthält genau $l-1$ Interclusterkanten und $i_e(\zeta) = n-l$ Intraclusterkanten. Zwei Cluster einer Clusterung sind also entweder direkt über genau eine Kante verbunden, oder über keine.*

5.1 w-MACP auf Bäumen

Sei $T = (V, E)$ ein Baum, $|V| = n$, $r \in V$ eine Wurzel des Baumes. $T^u = (V^u, E^u)$ bezeichne den Unterbaum von T, der an Knoten u gewurzelt ist. Die Kinder eines Knoten u seien mit $u_1, u_2, \ldots, u_{t(u)}$ bezeichnet, wobei $t(u) = \left\{ \begin{array}{ll} deg(u)-1 & u \neq r \\ deg(u) & sonst \end{array} \right\}$ ist.

Definition 16 (Partieller Teilbaum T_i^u von u)**.** *Sei $T_0^u = \big(\{u\}, \emptyset\big)$ der partielle Teilbaum, der nur Knoten u enthält. Dann ist $T_i^u = (V_i^u, E_i^u)$ mit der Knotenmenge $V_i^u = \{u\} \cup T^{u_1} \cup \ldots \cup T^{u_i}$ und Kantenmenge $E_i^u = \big\{\{u, u_k\} \mid 1 \leq k \leq i\big\} \cup E^{u_1} \cup \ldots \cup E^{u_i}$ der partielle Teilbaum mit i Kindern von u.*

Abbildung 5.1 stellt das Konzept von partiellen Teilbäumen eines Knoten u schematisch dar. Der partielle Teilbaum T_0^u besteht für jeden Knoten u nur aus u selbst. Wird i um 1

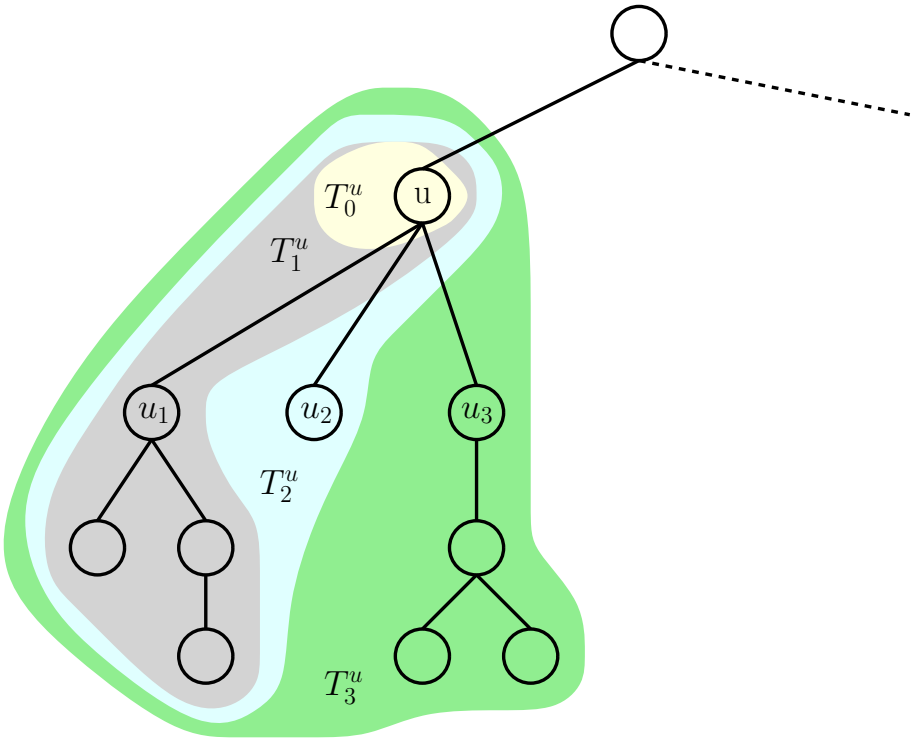


Abbildung 5.1: Partielle Teilbäume von Knoten u.

erhöht, wird der Unterbaum T^{u_i}, der an Kind u_i hängt, zum partiellen Unterbaum T^u_{i-1} hinzugefügt.

Optimale Lösungen von w-MACP lassen sich für einen Knoten u im Baum aus optimalen Lösungen von Teilbäumen des Knoten u berechnen. Hierzu dienen die folgenden Funktionen.

- $F^u(k)$: Das Minimum der Summe über die Quadrate der Komponenten-Gewichte in Teilbaum T^u bei Löschen von k Kanten in T^u
- $F^u(k,\nu)$: Das Minimum der Summe über die Quadrate der Komponenten-Gewichte in Teilbaum T^u bei Löschen von k Kanten in T^u, wenn die Komponente, die u enthält, das Gewicht ν besitzt.
- $F^u_i(k,\nu)$: Das Minimum der Summe über die Quadrate der Komponenten-Gewichte im partiellen Teilbaum T^u_i bei Löschen von k Kanten in T^u_i, wenn die Komponente, die u enthält, das Gewicht ν besitzt.

Die jeweiligen Werte für $F^u(k,\nu)$ und $F^u(k)$ ergeben sich in einfacher Weise aus den Werten von $F^u_i(k,\nu)$:

$$F^u(k,\nu) = F^u_{t(u)}(k,\nu) \tag{5.1}$$

$$F^u(k) = \min_{w(u)\le\nu\le w(T^u)} F^u(k,\nu) \tag{5.2}$$

Ergo genügt es, sich auf die Berechnung der Funktion $F^u_i(k,\nu)$ zu konzentrieren. Zu Beginn werden alle Werte von $F^u_i(k,\nu)$ wie folgt initialisiert.

$$F^u_i(0,\nu) = \begin{cases} w(T^u_i)^2 & \nu = w(T^u_i) \\ \infty & \text{sonst} \end{cases} \tag{5.3}$$

$$F_0^u(k,\nu) = \begin{cases} w(u)^2 & \nu = w(u) \\ \infty & \text{sonst} \end{cases} \tag{5.4}$$

Gleichung (5.3) betrachtet den Fall, dass keine Kanten im Teilbaum T^u gelöscht werden sollen. Dann entspricht die Summe über die Quadrate der Komponentengewichte im partiellen Teilbaum T_i^u gerade dem Quadrat des Komponentengewichts des Teilbaumes selbst. Gleichung (5.4) behandelt den Fall $i = 0$, in dem der partielle Teilbaum gerade aus Knoten u besteht. In diesem Fall ist die Summe über die Quadrate der Komponentengewichte in T_0^u gerade das Quadrat Gewichts von Knoten u selbst.

Um $F_i^u(k,\nu)$ für einen beliebigen Knoten u zu berechnen, müssen $F_{i-1}^u(l,\mu)$, $F^{u_i}(l,\mu)$, als auch $F^{u_i}(l)$ bereits für alle $0 \leq l \leq k$ und $0 \leq \mu \leq \nu$ berechnet sein. Dann ergibt sich der Wert für $F_i^u(k,\nu)$ wie folgt:

$$F_i^u(k,\nu) = \min \begin{cases} \min\limits_{0\leq l\leq k-1} \{F_{i-1}^u(l,\nu) + F^{u_i}(k-l-1)\} & (5.5) \\ \min\limits_{0\leq l\leq k, 0\leq \mu\leq \nu} \{F_{i-1}^u(l,\mu) + F^{u_i}(k-l,\nu-\mu) + 2\mu(\nu-\mu)\} & (5.6) \end{cases}$$

Gleichung (5.5) und 5.6 genügen einer anschaulichen Erklärung. Um k Kanten aus dem partiellen Teilbaum T_i^u zu löschen, werden zwei Fälle unterschieden.

- **Fall** (5.5). Wird die Kante $\{u, u_i\}$ im Baum gelöscht, so können bei Löschen von l Kanten im partiellen Teilbaum T_{i-1}^u nur noch $k-l-1$ Kanten im Unterbaum von u_i gelöscht werden. Die Komponente, die Knoten u enthält, stimmt außerdem mit der Komponente in T_{i-1}^u überein, die u enthält, da die neu hinzugekommene Kante $\{u, u_i\}$ gelöscht wurde und somit das Gewicht der Komponente, die u enthält, nicht zunimmt.

- **Fall** (5.6). Wird die Kante $\{u, u_i\}$ nicht gelöscht, so kann sich das Gewicht der Komponente, die u enthält, in Abhängigkeit der gelöschten Kanten in T^{u_i} und T_{i-1}^u ändern. Deshalb genügt es nicht das Maximum über die Anzahl an zu löschenden Kanten l zu bilden, sondern es ist nötig auch über alle möglichen Gewichte der Komponente, die u enthält, zu iterieren. Werden also l Kanten in T_{i-1}^u gelöscht und die Komponente, die u enthält, hat Gewicht μ, so bleiben $k-l$ Kanten in T^{u_i} und ein Komponenten-Gewicht von $\nu-\mu$. Die Werte von F_{i-1}^u und F^{u_i} beinhalten jeweils das Quadrat der Gewichte μ bzw. $\nu-\mu$ der Komponenten, die Knoten u, bzw. u_i im Teilbaum enthalten. Folglich müssen diese abgezogen werden und das neue Gewicht der Komponente, die u enthält addiert werden. Es ergibt sich also insgesamt ein Faktor von $-\mu^2 - (\nu-\mu)^2 + \big(\mu + (\nu-\mu)\big) = 2\mu(\nu-\mu)$, der zu den bestehenden Werten addiert wird.

Die Lösung des Problems w-MACP , sodass K Kanten im Baum gelöscht werden, ergibt sich an der Wurzel r des Baumes durch den Wert von $F^r(K)$. Algorithmus 3 stellt den vorgestellten Algorithmus schematisch dar.

Lemma 7 (Komplexität von w-MACP auf Bäumen)**.** *Der Wert einer optimalen Lösung von* w-MACP *kann für einen gegebenen Baum $T = (V,E)$ in $O\big(n^5 \cdot w_{max}^2\big)$ berechnet werden, wobei w_{max} der maximale Wert eines Knotens in T ist und $n := |V|$.*

Algorithmus 3 : BAUM-W-MACP

Data : Baum $T = (V, E)$
Result : $F^r(K)$ für alle $0 \leq K \leq m$
forall the $u \in V$ *in topologischer Ordnung* **do**
  for $i = 0$ *bis* $t(u)$ **do**
    for $k = 0$ *bis* $|E(u)|$ **do**
      for $\nu = 0$ *bis* $w(T^u)$ **do**
        Berechne $F_i^u(k, \nu)$, $F^u(k, \nu)$ und $F^u(k)$ mit Gleichung (5.5) und (5.6)
return $F^r(K)$

Algorithmus 4 : BAUM-SURPRISE

Data : Baum $T = (V, E)$
Result : optimaler SURPRISE-Wert für $T = (V, E)$
$F^u(K) =$ BAUM-W-MACP$\big(T = (V, E)\big)$
return $\min_{0 \leq K \leq n-1} \mathcal{S}(n - 1 - K, \frac{1}{2}F^r(K) - \frac{1}{2}n)$

Beweis. Sei $|V| =: n$. Um über alle Knoten und deren Kinder zu iterieren wird $O(n)$ Zeit benötigt. Es gibt weiterhin $O\big(n \cdot w(T^u)\big)$ Kombinationen für Variablen k und ν für einen Knoten u und Unterbaum T^u. Die Berechnung jedes $F_i^u(k, \nu)$ benötigt daher $O\big(n+n \cdot w(T^u)\big) = O\big(n \cdot w(T^u)\big)$ Zeit, da über alle möglichen Anzahlen an zu schneidenden Kanten, als auch über alle möglichen Größen der Komponente, die u in T^u enthält, iteriert werden muss. Mit der zusätzlich Abschätzung $w(T^u) \leq n \cdot w_{max}$ ergibt sich insgesamt eine Laufzeit von $O\big(n \cdot n \cdot w(T^u) \cdot n \cdot w(T^u)\big) = O\big(n^3 \cdot w(T^u)^2\big) = O\big(n^5 \cdot w_{max}^2\big)$, wobei w_{max} der maximale Wert eines Knotens im Baum ist. □

5.2 Surprise auf Bäumen

Um eine Lösung des Optimierungsproblems zu SURPRISE auf einem Baum $T = (V, E)$ zu berechnen, gibt Lemma 4 den Weg vor. Das Vorgehen besteht darin zuerst mit Algorithmus 3 eine Lösung $F^r(K)$ des Problems U-MACP als Spezialfall von W-MACP zu berechnen, um daraus anschließend den Wert einer optimalen SURPRISE-Clusterung zu folgern. Sei $F^r(K)$ also die Lösung von U-MACP, in der K Kanten gelöscht werden. Dann ist die Anzahl der Intraclusterkanten in der induzierten Clusterung $i_e(\zeta) = n - 1 - K$. Nach dem Beweis zu Lemma 4 ist weiterhin die Anzahl der Intraclusterpaare $i_p(\zeta) = \frac{1}{2}F^r(K) - \frac{1}{2}|V|$. Also ergibt sich der Wert einer bezüglich SURPRISE optimalen Clusterung auf T wie folgt:

$$\mathcal{S}_{min}(T) = \min_{0 \leq K \leq n-1} S\left(n - 1 - K, \frac{1}{2}F^r(K) - \frac{1}{2}|V|\right) \tag{5.7}$$

Algorithmus 4 bietet eine schematische Darstellung der beschriebenen Vorgehensweise.

Korollar 1 (Komplexität von SURPRISE auf Bäumen)**.** *Der Wert einer bezüglich* SURPRISE *optimalen Clusterung kann auf einem Baum $T = (V, E)$ mit $n := |V|$ in $O(n^5)$ berechnet werden.*

Beweis. Die Laufzeit ergibt sich als Spezialfall des Algorithmus für W-MACP auf Bäumen mit $w_{max} = 1$. □

Algorithmus 4 liefert lediglich den optimalen Wert der Clusterung, nicht die Clusterung selbst. Nichts desto trotz sollte es leicht möglich sein die Implementierung von Algorithmus 4 derart zu verändern, dass die optimale Clusterung ohne einen Verlust von Laufzeit gefunden werden kann.

6. Minimum-Contamination-Probleme auf planaren Graphen

Das folgende Kapitel soll die Frage erläutern, wie schwer das Lösen von Minimum-Average-Contamination-Problemen auf planaren Graphen ist. Planare Graphen sind von Interesse, da durch Planarität einige Optimierungsprobleme auf Graphen in Polynomialzeit lösbar sind, die auf allgemeinen Graphen $\mathcal{NP}$*-schwer* sind. Besonders Schnitt- und Flussprobleme lassen sich auf planaren Graphen meist effizienter lösen, als auf allgemeinen. Die wohl wichtigste Charakterisierung von planaren Graphen stellt der Satz von Kuratowski dar. Er verwendet den Begriff der Unterteilung, der zunächst kurz erläutert wird.

Definition 17 (Unterteilung eines Graphen)**.** *Sei $G = (V, E)$ ein Graph. Werden Kanten aus G durch einfache Pfade ersetzt, so heißt der neu entstandene Graph Unterteilung von G.*

Werden Kanten eines planaren Graphen mit weiteren Knoten unterteilt, so bleib der entstehende Graph planar.

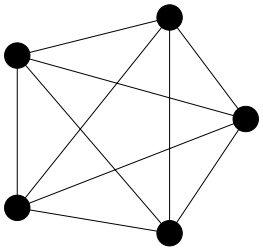
Satz 3 (Satz von Kuratowski)**.** *[Kur, Mak97] Ein Graph ist planar, genau dann, wenn er keine Unterteilung von K_5 oder $K_{3,3}$ als Subgraph enthält.*

Die Graphen K_5 und $K_{3,3}$ sind in Abbildung 6.1 dargestellt, wobei Abbildung 6.1c ein Beispiel einer Unterteilung von K_5 zeigt.

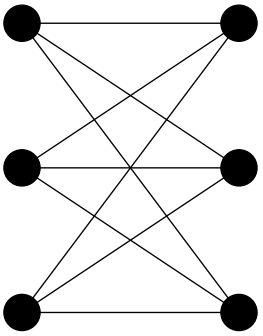
Satz 3 stellt ein wertvolles Werzeug dar, um zu entscheiden, ob ein gegebener Graph planar ist oder nicht. Im Folgenden wird davon Gebrauch gemacht, um zu zeigen, dass die Reduktion von Li und Tang zum Beweis der $\mathcal{NP}$*-vollständigkeit* von DECISION-w-MACP [LT11] einen möglicherweise nicht planaren Graphen erzeugt.

6.1 w-MACP auf planaren Graphen - Gegenbeispiel zu Beweis aus [LT11]

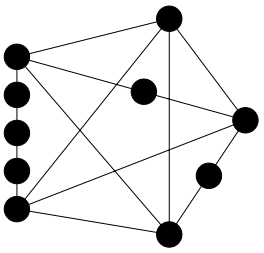
Der von Li und Tang angegebene Beweis soll zeigen, dass DECISION-w-MACP auf bipartiten, planaren Graphen $\mathcal{NP}$*-vollständig* ist [LT11]. Die Grundidee des Beweises besteht aus einer Polynomialzeit-Reduktion einer Spezialform des Problems 3-DM auf DECISION-w-MACP.



(a) K_5 - vollständiger Graph mit 5 Knoten.



(b) $K_{3,3}$ - vollständiger bipartiter Graph mit 3 Knoten in beiden Partitionen.



(c) Graph G ist Unterteilung des K_5 und ist nach Satz 3 somit nicht planar.

Abbildung 6.1: Illustration zu Satz 3.

Entscheidungsproblem 2 (Dreidimensionales Matching (3-DM))**.** *Seien A, B, C drei disjunkte Mengen mit $|A| = |B| = |C| = p$ und eine Menge von Tripeln $T \subseteq A \times B \times C$ gegeben. Gibt es eine Menge $S \subseteq T$ der Größe p, sodass jedes Element $g \in A \cup B \cup C$ in genau einem Tripel $s \in S$ enthalten ist?*

Die Spezialform 3-DM$_2$ des Problems 3-DM, die der Beweis von Li und Tang verwendet, besteht darin die Anzahl der Vorkommen eines Elements $g \in A \cup B \cup C$ in einem Tripel $d \in T$ auf genau zwei zu beschränken. Das zugehörige Entscheidungsproblem ist *$\mathcal{NP}$-vollständig* [CC03, LT11]. Reduktion 1 stellt den Kern der angegebenen Reduktion dar.

Reduktion 1. Aus einer Instanz $I = \big((A, B, C), T\big)$ mit $|A| = p, |T| = 2p$ von 3-DM$_2$ wird ein Graph G_I wie folgt konstruiert. Für jedes Element $g \in A \cup B \cup C$ werden Knoten $p_1(g)$ und $p_2(g)$ eingefügt, die die Vorkommen der Elemente $g \in A \cup B \cup C$ repräsentieren. Des Weiteren werden Verbindungsknoten Δg eingeführt, die jeweils über Kanten $\{\Delta g, p_1(g)\}$ und $\{\Delta g, p_2(g)\}$ mit $p_1(g)$, bzw. $p_2(g)$ verbunden sind. Die dritte Gruppe von Knoten $p(d)$ repräsentieren Tripel $d \in T$. Sie besitzen Kanten zu den Knoten $p_1(g)$, bzw. $p_2(g)$, deren repräsentiertes Element in d vorkommt.

Die Gewichte der einzelnen Knoten und die Parameter für die Erfüllbarkeit des Problems DECISION-W-MACP sind hier nicht angegeben, da sie bezüglich der Planarität von wenig Interesse sind. Für die vollständige Reduktion siehe [LT11]. Es wird behauptet, dass der Graph G_I für alle Instanzen I von 3-DM$_2$ planar ist, was mit dem folgenden Beispiel widerlegt werden soll.

Beispiel 1. *Beispiel zu* 3-DM$_2$

$$
\begin{aligned}
&I_1 = \big((A, B, C), T\big) \\
&A = \{a, b, c\}\ B = \{d, e, f\}\ C = \{g, h, k\} \\
&T = \big\{(a, d, g), (a, e, h), (b, e, k), (b, f, g), (c, f, h), (c, d, k)\big\}
\end{aligned}
$$

Abbildung 6.2a zeigt den für Beispiel 1 durch Reduktion 1 konstruierten Graphen G_{I_1}.

Lemma 8. *Der für Beispiel 1 mittels Reduktion 1 konstruierte Graph G_{I_1} ist nicht planar.*

Beweis. In G_{I_1} sind zwei Knoten $p(d_i), p(d_j)$ mit $d_i, d_j \in T$ über einen Pfad, der ausschließlich aus Knoten mit Grad 2 besteht erreichbar, wenn d_i und d_j ein Element $g \in A \cup B \cup C$ gemeinsam haben. Werden diese Pfade in G_{I_1} durch einfache Kanten ersetzt, entsteht der in 6.2b dargestellte Graph G'_{I_1}, der isomorph zu $K_{3,3}$ ist. Nach Satz 3 ist der konstruierte Graph G_{I_1} also nicht planar, da er eine Unterteilung des Graphen $K_{3,3}$ enthält. □

Jede Instanz I von 3-DM$_2$, in der die Menge aller Tripel T eine Obermenge der Menge aller Tripel aus Beispiel 1 bildet, erzeugt einen nicht-planaren Graphen G_I. Der in [LT11] angegebene Beweis besitzt noch immer Gültigkeit für bipartite Graphen. Darüber hinaus, lässt sich die Idee hinter Reduktion 1 verwenden, um einen neuen Beweis zur *$\mathcal{NP}$-vollständigkeit* von DECISION-w-MACP anzugehen, was Gegenstand des folgenden Abschnittes ist.

6.2 w-MACP auf planaren Graphen

Dieser Abschnitt macht sich zur Aufgabe die *$\mathcal{NP}$-vollständigkeit* von DECISION-w-MACP auf planaren Graphen zu zeigen. Hierzu wird, anstatt von einer Instanz von 3-DM$_2$, von einer Instanz I des Problems (2-3)-PLANARES 3-DM auf DECISION-w-MACP reduziert. Die angegebene Reduktion orientiert sich stark an der von Li und Tang angegebenen Reduktion [LT11], trifft aber an verschiedenen Stellen kleine Änderungen, wodurch der Reduktionsgraph planar bleibt. Den Änderungen geschuldet, bedarf es auch einer angepassten Argumentation im angeschlossenen Beweis.

Entscheidungsproblem 3 ((2-3)-Planares Dreidimensionales Matching ((2-3)-PLANARES 3-DM))**.** *Sei $I = \big((A, B, C), T\big)$ eine Instanz von* 3-DM*, in der jedes Element $g \in A \cup B \cup C$ entweder in zwei oder drei Tripeln in T vorkommt und $G = (V, E)$ mit*

$$V = \{d \mid d \in T\} \cup A \cup B \cup C$$
$$E = \big\{\{g, d\}\big\} \mid d \in T, g \in A \cup B \cup C \text{ und } g \in d\big\}$$

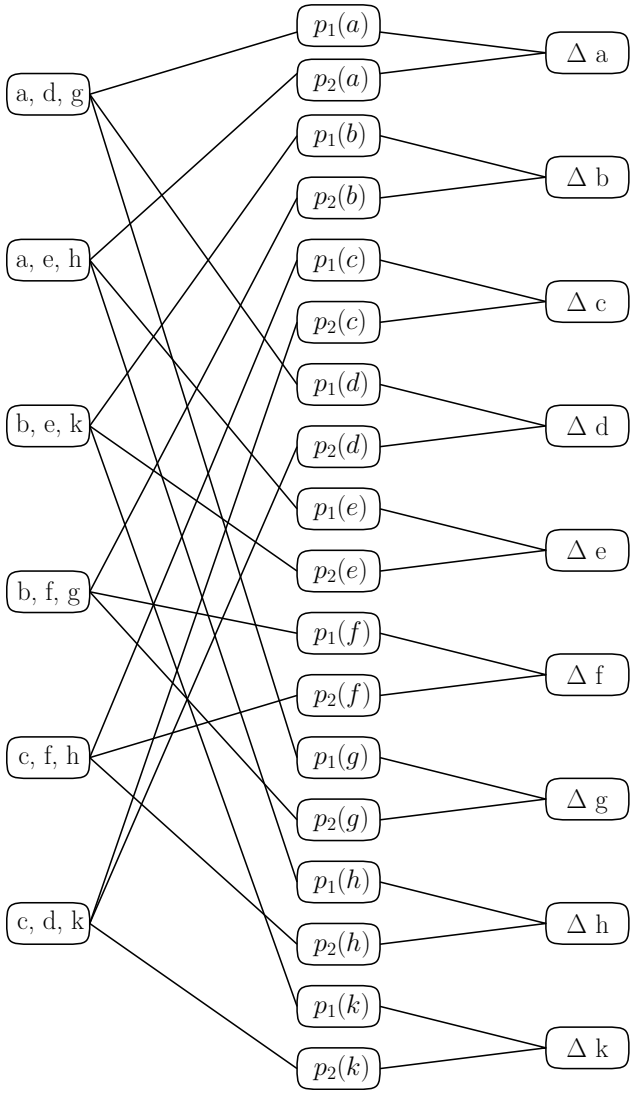
der Graph, der I beschreibt. I heißt planar, falls G planar ist. Gibt es eine Menge $S \subseteq T$ der Größe p, sodass jedes Element $g \in A \cup B \cup C$ in genau einem Tripel $s \in S$ enthalten ist?

Das Entscheidungproblem (2-3)-PLANARES 3-DM ist *$\mathcal{NP}$-vollständig* . Dies folgt unmittelbar aus einem in [DF86] angegebenen Beweis zur *$\mathcal{NP}$-vollständigkeit* von PLANARES 3-DM, da in diesem Beweis nur ein Vorkommen in zwei oder drei Tripeln für alle Variablen $g \in A \cup B \cup C$ vorausgesetzt wird.

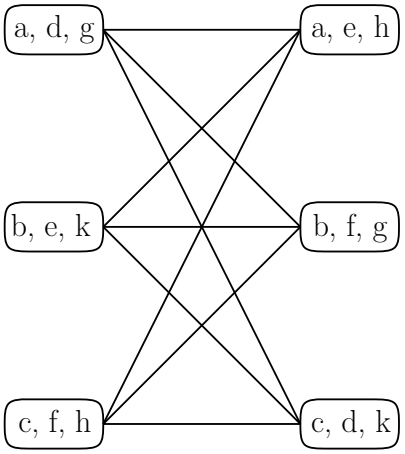
Im Folgenden seien A_i, B_i und C_i die Mengen der Elemente aus A, B, bzw. C, die in T genau in i Tripeln vorkommen.

Definition 18.
$A_i := \{g \in A \mid \big|\{t \in T \mid g \in t\}\big| = i\}$
$B_i := \{g \in B \mid \big|\{t \in T \mid g \in t\}\big| = i\}$
$C_i := \{g \in C \mid \big|\{t \in T \mid g \in t\}\big| = i\}$

Für jede Instanz von (2-3)-PLANARES 3-DM gilt $A = A_2 \cup A_3$, $B = B_2 \cup B_3$ und $C = C_2 \cup C_3$.



(a) Graph G_{I_1} zu Beispiel 1 nach Reduktion 1 aus [LT11].



(b) Graph G'_{I_1} zu Beispiel 1. Der Graph G'_{I_1} entsteht durch Knotenkontraktion des Graphen in a), ist isomorph zu $K_{3,3}$ und damit nicht planar.

Abbildung 6.2: Illustration zu Reduktion 1 für Beispiel 1

Reduktion 2. Reduktion von (2-3)-PLANARES 3-DM auf w-MACP.

Sei $I = \big((A, B, C), T\big)$ eine Instanz von (2-3)-PLANARES 3-DM. Aus I wird auf folgende Art und Weise ein Graph $G_{Red}(I) = \big(V_{Red}(I), E_{Red}(I)\big)$ konstruiert. Für jedes Vorkommen in T eines Elements $g \in A \cup B \cup C$ werden Knoten $p_1(g)$, $p_2(g)$, bzw. $p_3(g)$, jeweils mit Gewicht 1 in $G_{Red}(I)$ eingefügt. Jedes Element d aus T wird durch einen Knoten $p(d)$ repräsentiert, der jeweils mit genau einem Knoten $p_i(g), 1 \leq i \leq 3$ des in d enthaltenen Vorkommen der Variablen g über eine Kante verbunden wird. Zu jedem Element $g \in A_2$ wird ein Knoten Δg mit Gewicht 2 eingeführt, der mit den Knoten $p_1(g)$ und $p_2(g)$ über Kanten verbunden wird. Für jede Variable $g \in A_3$ werden zwei Knoten $\Delta_1 g$ und $\Delta_2 g$ mit Gewicht 2 eingeführt, die über Kanten $\{p_1(g), \Delta_1 g\}, \{p_2(g), \Delta_1 g\}, \{p_2(g), \Delta_2 g\}, \{p_3(g), \Delta_2 g\}$ mit den Knoten $p_i(g)$ der Variablen $g \in A_3$ verbunden werden. Zu jedem Element $g \in B \cup C$ wird ein Knoten Δg eingefügt. Falls $g \in B_2 \cup C_2$, bekommt dieser Gewicht $N + 2$ und es werden Kanten $\{p_1(g), \Delta g\}, \{p_2(g), \Delta g\}$ eingefügt. Für $g \in B_3 \cup C_3$ bekommt der Knoten Δg Gewicht $N + 1$ und es werden Kanten $\{p_1(g), \Delta g\}, \{p_2(g), \Delta g\}, \{p_3(g), \Delta g\}$ eingefügt. Der erhaltene Graph $G_{Red}(I)$ stellt nun eine Instanz von w-MACP mit Parametern $K = 3\,|T| - |A|$ und $B = (4\,|A| + |A_3|)(N + 3)^2$ dar. Der Parameter N wird im späteren Verlauf des Beweises festgelegt. An dieser Stelle sei angemerkt, dass es nur von Nöten ist, dass N eine gewisse Mindestgröße besitzt. Die beschriebene Reduktion ist polynomiell, da für jedes Element $d \in T$ genau ein Knoten und für jedes $g \in A \cup B \cup C$ höchstens 5 Knoten eingefügt werden und die Knotengewichtsfunktion des Graphen bei binärer Codierung polynomiell in der Eingabegröße ist.

Zur Veranschaulichung ist in Beispiel 2 eine Beispielinstanz I von (2-3)-PLANARES 3-DM gegeben, deren Graph $G_{Red}(I)$ in Abbildung 6.3a dargestellt ist.

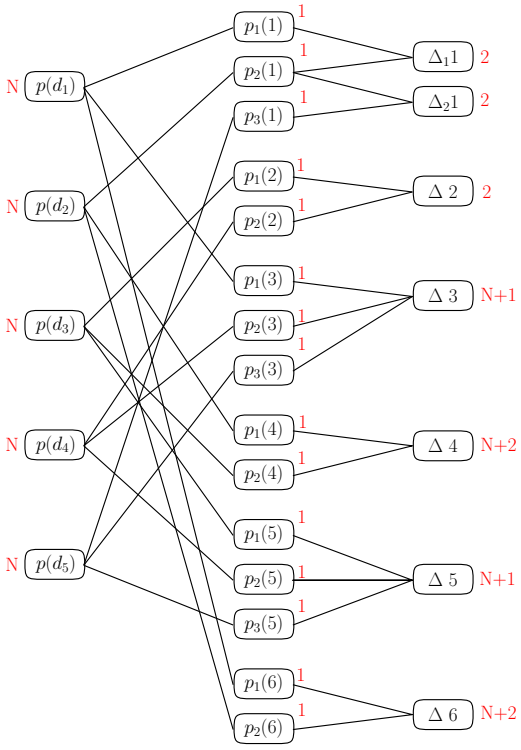
Beispiel 2.

$$
\begin{aligned}
&I = \big((A, B, C), T\big) \textit{ mit}\\
&A = \{1, 2\}, B = \{3, 4\}, C = \{5, 6\}\\
&T = \{(1, 3, 6), (1, 4, 6), (2, 4, 5), (2, 3, 5), (1, 3, 5)\}
\end{aligned}
$$

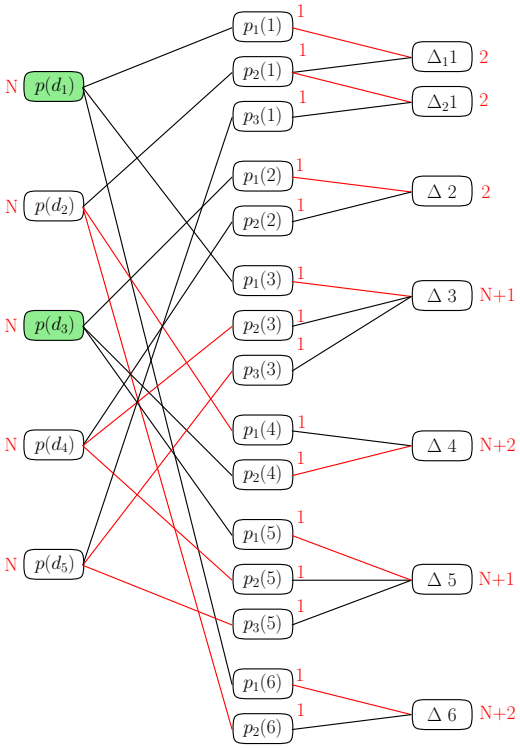
Zunächst ist zu zeigen, dass $G_{Red}(I)$ für alle Instanzen I von (2-3)-PLANARES 3-DM planar ist. Hierbei hilft das Konzept der Unterteilung eines Graphen in Verbindung mit Satz 3.

Lemma 9. *Sei I eine Instanz von* (2-3)-PLANARES 3-DM. *Dann ist $G_{Red}(I)$ planar.*

Beweis. Sei hierzu I eine Instanz von (2-3)-PLANARES 3-DM und $G(I)$ der in Entscheidungsproblem 3 beschriebene Graph zu I. $G(I)$ ist per Definition planar. Sei $g \in B \cup C$ und $p(g)$ der zugehörige Knoten in $G(I)$ und $p(d_1), p(d_2)$ und gegebenenfalls $p(d_3)$ die Knoten, für deren repräsentierte Tripel $g \in d_i$ gilt. Durch Umbenennung von $p(g)$ zu Δg und Unterteilung der Kanten $\{p(d_1), p(g)\}, \{p(d_2), p(g)\}$ und $\{p(d_3), p(g)\}$ mit neuen Knoten $p_1(g), p_2(g), p_3(g)$ entsteht die Konstruktion des Graphen $G_{Red}(I)$, die nach Satz 3 planar bleibt, da sie eine Unterteilung eines planaren Graphen ist. Für alle Knoten $g \in A_2$ erfolgt die Konstruktion analog. Sei also nun $g \in A_3$ ein Element, das durch Knoten $p(g)$ repräsentiert wird und $p(d_1), p(d_2)$, sowie $p(d_3)$ die Knoten der Tripel, in denen g vorkommt. Unterteile zunächst die Kanten $\{p(g), p(d_1)\}$ und $\{p(g), p(d_3)\}$ von Knoten $p(d_1)$, bzw. $p(d_3)$ ausgehend, jeweils mit zwei Knoten $p_1(g)$ und $\Delta_1 g$, bzw. $p_3(g)$ und $\Delta_2 g$. Durch Umbenennung von $p(g)$ in $p_2(g)$ entsteht die in $G_{Red}(I)$ beschriebene Struktur des Graphen, womit aus Satz 3 folgt, dass $G_{Red}(I)$ für alle Instanzen I von (2-3)-PLANARES 3-DM planar ist. □



(a) $G_{Red}(I)$ für Beispiel 2



(b) Für grün eingefärbte Knoten $p(d)$ ist das Tripel d im perfekten Matching enthalten. Rot eingefärbte Kanten sind in der Menge E', die aus $G_{Red}(I)$ gelöscht wird.

Abbildung 6.3

Es bleibt zu zeigen, dass $G_{Red}(I)$ genau dann ein dreidimensionales Matching S enthält, wenn es eine Clusterung ζ mit Gewicht $B = \sum_{C\in\zeta} w(C)^2 = (4\,|A| + |A_3|)(N+3)^2$ gibt, sodass K Kanten geschnitten werden.

Definition 19 (schwerer Knoten)**.** *Ein Knoten $v \in V_{Red}(I)$ heißt schwer, wenn $w(v) \geq N$ gilt.*

Wird die in Reduktion 2 eingeführte Zahl N groß genug gewählt, so lässt sich über eine Lösung von DECISION-w-MACP folgendes Lemma ableiten.

Lemma 10. *Für jede erfüllbare Instanz I' des Problems* DECISION-w-MACP *mit erfüllender Clusterung ζ, enthält jeder Cluster $C \in \zeta$ höchstens einen schweren Knoten.*

Beweis. Sei $I' = \big((V,E), K, B\big)$ eine erfüllbare Instanz von DECISION-w-MACP mit erfüllender Clusterung ζ, sodass $K = 3\,|T| - |A|$ und $B = (4\,|A| + |A_3|)(N+3)^2$ ist. Es ist klar, dass jeder schweren Knoten in einem Cluster enthalten sein muss. Da es $4\,|A| + |A_3|$ schwere Knoten gibt, gilt also

$$\sum_{C\in\zeta} w(C)^2 \geq (4\,|A| + |A_3|)N^2. \tag{6.1}$$

Außerdem sei nun $N > 40\,|A| + 10\,|A_3|$. Somit gilt:

$$\begin{aligned}
(4\,|A| + |A_3|)(N+3)^2 &= (4\,|A| + |A_3|)N^2 + (6N+9)(4\,|A| + |A_3|) \\
&\leq (4\,|A| + |A_3|)N^2 + (6N+9)(4\,|A| + |A|) \\
&< (4\,|A| + |A_3|)N^2 + (6N+N)5\,|A| \\
&= (4\,|A| + |A_3|)N^2 + 35N \underbrace{|A|}_{<\frac{1}{40}N} \\
&< (4\,|A| + |A_3|)N^2 + N^2 \\
&< (4\,|A| + |A_3|)N^2 + 2N^2 \\
&= (4\,|A| + |A_3| + 2)N^2
\end{aligned} \tag{6.2}$$

Angenommen ein Cluster $C \in \zeta$ enthält mehr als einen schweren Knoten, d.h. es gibt einen Cluster C mit $w(C) \geq 4N^2$. Dann gilt:

$$\begin{aligned}
\sum_{C\in\zeta} w(C)^2 &\geq 4N^2 + (4\,|A| + |A_3| - 2)N^2 \\
&= N^2(4\,|A| + |A_3| + 2) \\
&> (4\,|A| + |A_3|)(N+3)^2
\end{aligned}$$

Dies stellt einen Widerspruch zur Annahme dar und folglich gilt die Behauptung.

□

Ausgehend von Lemma 10 kann nun das folgende Lemma gezeigt werden.

Lemma 11. *Eine Instanz $I = \big((A,B,C), T\big)$ von* (2-3)-PLANARES 3-DM *ist genau dann erfüllbar, wenn* DECISION-w-MACP *mit Parametern $K = 3\,|T| - |A|$ und $B = (4\,|A| + |A_3|)(N+3)^2$ auf $G_{Red}(I)$ erfüllbar ist.*

Beweis. "⇒": Sei $S \subset T$ ein perfektes dreidimensionales Matching von I und $G_{Red}(I)$ der für I in Reduktion 2 beschriebene Graph. Dann enthalte die Menge E' die folgenden Kanten.

- Für alle $d = (a, b, c) \notin S$ für die $p(d)$ mit Knoten $p_i(a), p_j(b)$ und $p_l(c)$ adjazent ist, seien die Kanten $\{p(d), p_j(b)\}$ und $\{p(d), p_l(c)\}$ mit $b \in B$ und $c \in C$ in E'.
- Für alle $d = (a, b, c) \in S$ für die $p(d)$ mit Knoten $p_i(a), p_j(b)$ und $p_l(c)$ adjazent ist und $a \in A_2$ gilt, seien die Kanten $\{p_i(a), \Delta a\}$, $\{p_j(b), \Delta b\}$ und $\{p_l(c), \Delta c\}$ in E'.
- Zusätzlich werden für alle $d = (a, b, c) \in S$ mit $a \in A_3$ die in Abbildung 6.4 angezeigten Kanten zu E' hinzugefügt.

Wie für Beispiel 2 in Abbildung 6.3b leicht ersichtlich ist, hat nun jede verbleibende Zusammenhangskomponente in $G' = (V_{Red}(I), E_{Red}(I) \setminus E')$ genau Gewicht $N + 3$.

In jedem Element $d \in T$ muss genau ein Element aus $a \in A$ enthalten sein. Folglich enthält T zwei Tripel für beide Vorkommen aller Elemente in A und ein weiteres für jedes Element aus A_3, sodass sie folgende Identität gilt:

$$|T| = 2\,|A| + |A_3| \tag{6.3}$$

Hieraus lässt sich die Anzahl der Kanten herleiten, die E' enthält. Für jedes Tripel $d \in S$ enthält E' drei Kanten. Weiterhin ist eine weitere Kante für jedes Element $g \in A_3$ in E' und für jedes Element $d \notin S$, das nicht im Matching S enthalten ist, werden genau zwei Kanten zu E' hinzugefügt, vergleiche hierzu Abbildung 6.4. Folglich gilt:

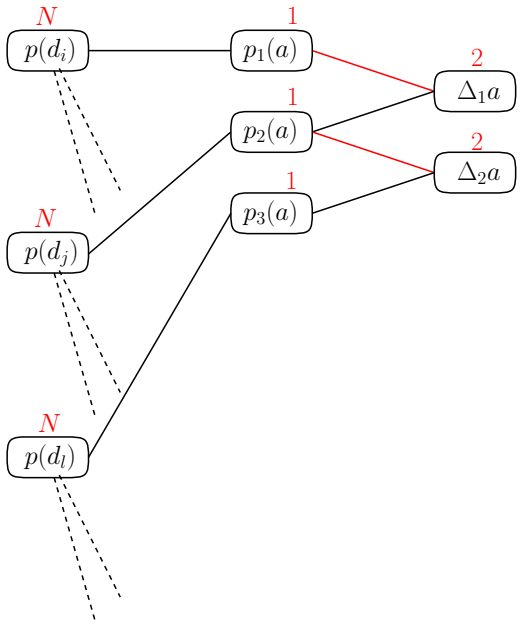
$$\begin{aligned}
\left|E'\right| &= 3\,|S| + \underbrace{|A_3|}_{=|T|-2|A|} + 2\,|T \setminus S| \\
&= 3\,\underbrace{|S|}_{=|A|} + |T| - 2\,|A| + 2(|T| - \underbrace{|S|}_{=|A|}) \\
&= 3\,|T| - |A|
\end{aligned}$$

Werden alle Kanten aus E' in $G_{Red}(I)$ gelöscht, entsteht der Graph $G' = (V_{Red}(I), E_{Red}(I) \setminus E')$. Er zerfällt in natürlicher Weise in eine Clusterung $\zeta = \{C_1, \ldots, C_q\}$, die durch seine Zusammenhangskomponenten induziert wird. Die Anzahl der Cluster q in G' setzt sich zusammen aus einer Komponente für jeden schweren Knoten, also einen für jedes Tripel $d \in T$ und einen für jedes $g \in B \cup C$. Somit ergibt sich:

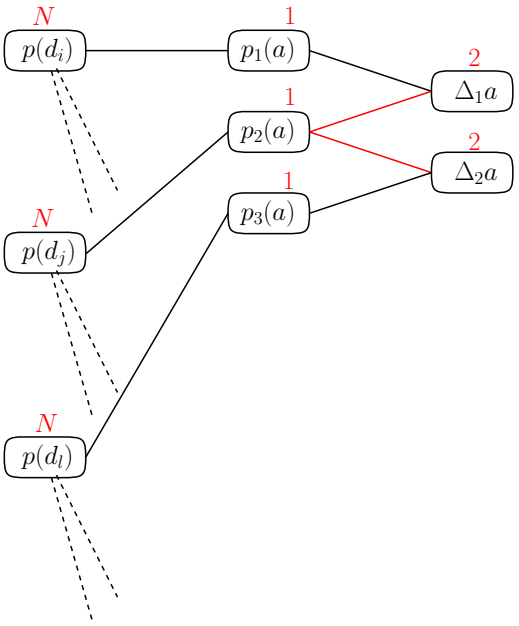
$$\begin{aligned}
q &= |T| + |B| + |C| \\
&= |T| + |A| + |A| \\
&= 2\,|A| + |A_3| + 2\,|A| \\
&= 4\,|A| + |A_3|
\end{aligned}$$

Da es genau $q = 4\,|A| + |A_3|$ Cluster mit jeweils Gewicht $N+3$ in G' gibt, gilt $\sum_{C \in \zeta} w(C)^2 = (4\,|A| + |A_3|)(N + 3)^2$. Somit kann aus einer erfüllenden Belegung für (2-3)-PLANARES 3-DM eine erfüllende Clusterung von DECISION-W-MACP mit Parametern $K = 3\,|T| - |A|$ und $B = (4\,|A| + |A_3|)(N + 3)^2$ auf $G_{Red}(I)$ bestimmt werden.

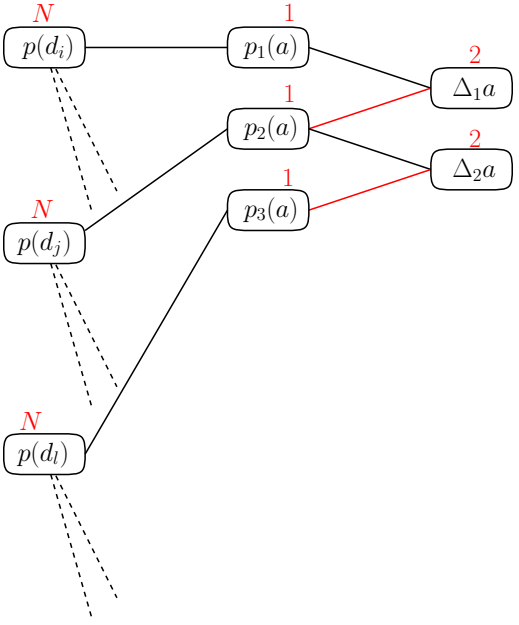
"⇐": Sei I' eine erfüllende Clusterung von DECISION-W-MACP mit $K = 3\,|T| - |A|$ und $B = (4\,|A| + |A_3|)(N + 3)^2$. Dann existiert eine Kantenmenge E' mit $|E'| = K$,



(a) Fall 1: $d_i \in S$

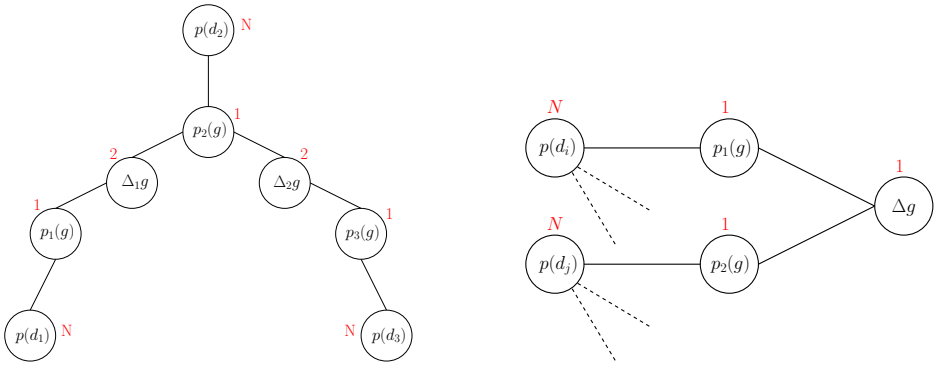


(b) Fall 2: $d_j \in S$



(c) Fall 3: $d_l \in S$

Abbildung 6.4: Dargestellt sind die drei Fälle für den Schnitt in Komponenten zu $g \in A_3$. Alle Kanten, die in der Menge E' enthalten sind, sind rot dargestellt.



(a) Sternförmige Anordnung bei Knoten $p(g)$ mit $g \in A_3$. Zum Trennen aller Pfade zwischen schweren Knoten sind zwei Schnitte nötig.

(b) Für $g \in A_2$ muss in der zu g gehörigen Komponente nur eine Kante geschnitten werden, um alle schweren Knoten zu trennen.

Abbildung 6.5: Illustration für alle benötigten Schnittkanten für Knoten $p(g)$ von Elemente $g \in A$

sodass sich nach Löschen von allen Kanten $e \in E'$ aus $G_{Red}(I)$ eine Clusterung ζ von $G' = (V_{Red}(I), E_{Red}(I) \setminus E')$ ergibt, für die

$$\sum_{C \in \zeta} w(C)^2 \leq (4\,|A| + |A_3|)(N+3)^2$$

gilt. Lemma 10 besagt, dass jeder Cluster $C \in \zeta$ höchstens einen schweren Knoten enthalten darf. Folglich ist es nötig alle Pfade, die zwischen schweren Knoten verlaufen und keinen weiteren schweren Knoten enthalten, zu schneiden. Die Anzahl solcher Pfade ergibt sich wie folgt:

Für jeden Knoten Δg mit $g \in B_3 \cup C_3$ müssen drei Kanten geschnitten werden, um alle Pfade zwischen ihm und anderen schweren Knoten zu trennen, für alle Knoten Δg mit $g \in B_2 \cup C_2$ genau zwei. Des Weiteren müssen für alle Knoten Δg mit $g \in A_3$ zwei und für jeden Knoten Δg mit $g \in A_2$ eine Kante geschnitten werden, was in Abbildung 6.5 verdeutlicht ist. Anders betrachtet, wird für jedes Element in A eine Kante geschnitten und zusätzlich für jedes $g \in A_3$ eine weitere Kante. Damit ergibt sich folgende Gesamtzahl der benötigten Kanten im Schnitt, um alle schweren Knoten zu trennen.

$$\begin{aligned}
\left|E'\right| &= 3\big(\,|\{\Delta g \mid g \in B_3\}| + |\{\Delta g \mid g \in C_3\}|\,\big) \\
&\quad + 2\big(\,|\{\Delta g \mid g \in B_2\}| + |\{\Delta g \mid g \in C_2\}|\,\big) \\
&\quad + |A| + \underbrace{|A_3|}_{=|T|-2|A|} \\
&= 3\big(\,|B_3| + |C_3|\,\big) + 2\big(\,|B_2| + |C_2|\,\big) + |A| + |T| - 2\,|A| \\
&= 3\big(\,|A_3| + |A_3|\,\big) + 2\big(3\,|A| - |T| + 3\,|A| - |T|\,\big) + |T| - |A| \\
&= 6\big(\,|T| - 2\,|A|\,\big) + 4\big(3\,|A| - |T|\,\big) + |T| - |A| \\
&= 2\,|T| + |T| - |A| \\
&= 3\,|T| - |A|
\end{aligned}$$

Da jeder Cluster höchstens einen schweren Knoten enthalten kann und da um alle Pfade zwischen schweren Knoten zu trennen bereits $3\,|T| - |A|$ Kanten benötigt werden, kann es

nicht vorkommen, dass ein Cluster existiert, der keinen schweren Knoten enthält. Somit enthält jeder Cluster genau einen schweren Knoten. Nach Konstruktion von $G_{Red}(I)$ gilt:

$$\sum_{v \in V_{Red}(I)} w(v) = \sum_{C \in \zeta} w(C) = (4\,|A| + |A_3|)(N+3)$$

Da es $4\,|A| + |A_3|$ schwere Knoten gibt und jeder Knoten in einem Cluster enthalten sein muss, aber keine zwei schweren Knoten in einem Cluster enthalten sein können(vgl. Lemma 10), kann

$$\sum_{C \in \zeta} w(C)^2 \leq (4\,|A| + |A_3|)(N+3)^2$$

nur gelten, wenn $w(C) = N+3$ für alle Cluster $C \in \zeta$ gilt.

Im Folgenden bezeichne ein *gültiger Cluster* eine Menge $\{p(d), p(a), p(b), p(c)\} \subset V_{Red}(I)$ von Knoten in $G_{Red}(I)$ mit $a \in A, b \in B$ und $c \in C$. Es bleibt zu zeigen, dass zu jeder Variablen $g \in A \cup B \cup C$ genau einer der Knoten $p_1(g), p_2(g)$, bzw. $p_3(g)$ in einem gültigen Cluster liegt, das heißt ein dreidimensionales Matching S induziert wird.

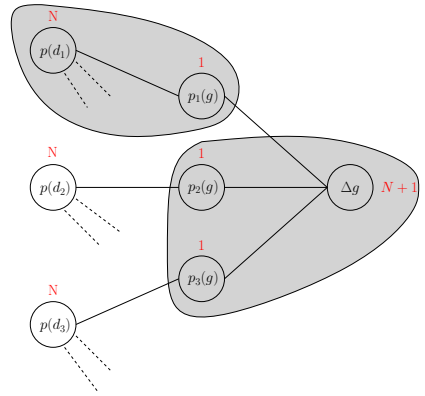
Dies wird zunächst für alle $g \in B \cup C$ gezeigt. Sei hierzu $g \in B \cup C$ und $C \in \zeta$ ein Cluster, der $p_1(g)$ enthält.

- **Fall 1:** $p(d)$ ist schwerer Knoten in C. Dann ist C offensichtlich ein gültiger Cluster, da es keine andere Möglichkeit gibt, sodass $w(C) = N+3$ gilt. Für die verbleibenden Knoten gibt es die folgenden zwei Fälle:
  - **Fall 1.1:** $g \in B_3 \cup C_3$. Dann enthält der Cluster, in dem Knoten Δg ist auch die Knoten $p_2(g)$ und $p_3(g)$ und bilden somit einen Cluster C' mit Gewicht $N+3$, der nicht gültig ist.
  - **Fall 1.2:** $g \in B_2 \cup C_2$. Dann enthält der Cluster, in dem Knoten Δg ist auch den Knoten $p_2(g)$ und bildet ebenfalls einen nicht gültigen Cluster C' mit Gewicht $N+3$.
- **Fall 2:** Es gibt kein $d \in T$, sodass $p(d)$ in C liegt. Dann muss der schwere Knoten Δg in C liegen und C ist kein gültiger Cluster. Es treten zwei Fälle auf:
  - **Fall 2.1:** $g \in B_3 \cup C_3$. Dann muss C einen Knoten $p_i(g)$ mit $i \neq 1$ enthalten. Folglich muss der dritte Knoten $p_j(g), j \notin \{1, i\}$, der ein Vorkommen der Variablen g repräsentiert, in einem anderen Cluster liegen, der seinerseits ein gültiger Cluster sein muss.
  - **Fall 2.2:** $g \in B_2 \cup C_2$. Dann gilt bereits $w(C) = N+3$ und der Knoten $p_2(g)$ muss in einem gültigen Cluster liegen, um zu gewährleisten, dass $w(C) = N+3$ für alle Cluster C gilt.

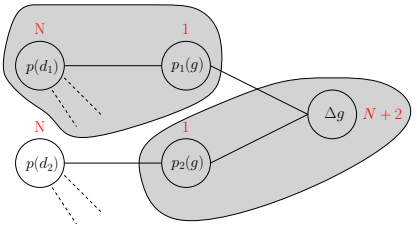
Die einzelnen Fälle für $g \in B \cup C$ sind zur Übersicht in Abbildung 6.6 dargestellt.

Da es für jedes $b \in B$ einen gültigen Cluster C gibt, der einen Knoten $p_i(b)$ enthält, gibt es insgesamt mindestens $|B| = |A|$ gültige Komponenten. Es bleibt also nur noch zu zeigen, dass für $g \in A$ höchstens ein Knoten $p_i(g)$ in einem gültigen Cluster liegen kann. Sei hierzu C der Cluster der $p_1(g)$ enthält.

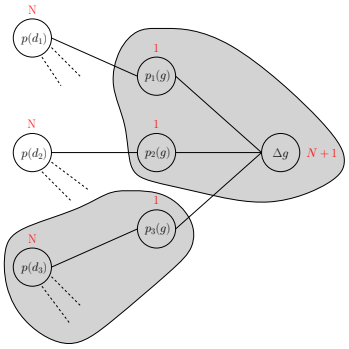
- **Fall 1:** $g \in A_2$.
  - **Fall 1.1:** C ist gültiger Cluster. Dann muss Knoten Δg mit Knoten $p_2(g)$ und dem zu $p_2(g)$ adjazenten Knoten $p(d_j)$ einen nicht gültigen Cluster bilden.



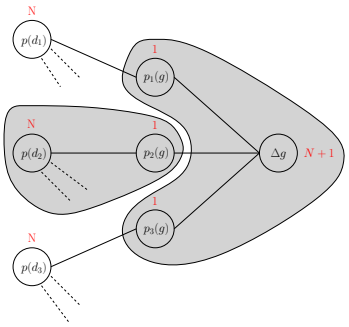
(a) Fall 1.1



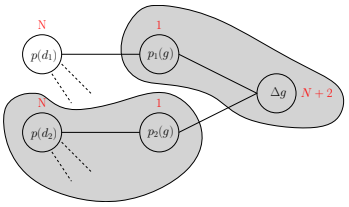
(b) Fall 1.2



(c) Fall 2.1 Möglichkeit 1



(d) Fall 2.1 Möglichkeit 2



(e) Fall 2.2

Abbildung 6.6: Fallunterscheidnung für $g \in B \cup C$

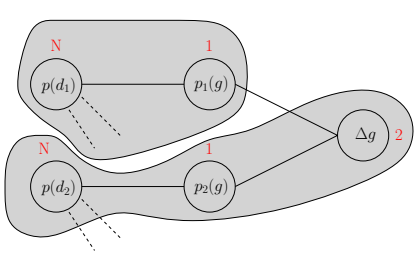
  - **Fall 1.2:** C ist nicht gültiger Cluster. Dann muss C den Knoten Δg enthalten und der Cluster C', der Knoten $p_2(g)$ enthält, ist ein gültiger Cluster.

- **Fall 2:** $g \in A_3$.
  - **Fall 2.1:** C ist gültiger Cluster. Dann müssen die Knoten $p_2(g)$, $\Delta_1 g$ und $p(d_j)$ den ersten und die Knoten $p_3(g)$, $\Delta_2 g$ und $p(d_l)$ den zweiten nicht gültigen Cluster bilden, da keine andere Möglichkeit besteht Cluster der Größe $N+3$ zu erhalten.
  - **Fall 2.2:** C ist nicht gültiger Cluster. Dann muss ein gültiger Cluster C' existieren, der entweder $p_2(g)$ oder $p_3(g)$ enthält:
    * **Fall 2.2.1:** C' enthält $p_2(g)$. Die Knoten $\Delta_1 g$, $p_1(g)$ und $p(d_i)$ müssen zusammen in einem Cluster liegen und die Knoten $\Delta_2 g$, $p_3(g)$ und $p(d_l)$ müssen zusammen in einem Cluster liegen. Beide Cluster sind jeweils nicht gültig.
    * **Fall 2.2.2:** C' enthält $p_3(g)$. Analog Fall 2.1 (symmetrisch).

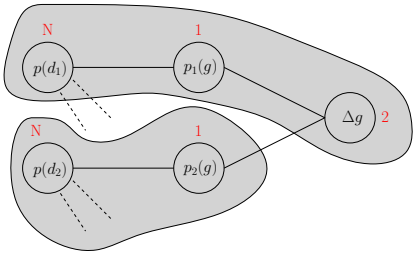
Für $g \in A$ sind die beschrieben Fälle in Abbildung 6.7 dargestellt.

Insgesamt induziert eine erfüllende Clusterung von DECISION-W-MACP also genau $|A|$ gültige Cluster in $G_{Red}(I)$, in der jede Variable in genau einem Cluster enthalten ist, ein perfektes dreidimensionales Matching S der Instanz I von (2-3)-PLANARES 3-DM. □

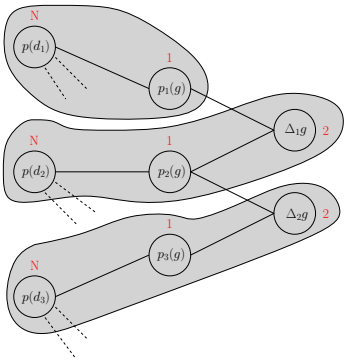
Zu einer gegebene Clusterung können die Summe der Quadrate der Cluster-Gewichte und die Anzahl an gelöschten Kanten in polynomieller Zeit berechnet werden. Somit liegt das



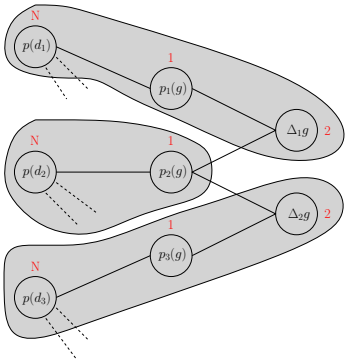
(a) Fall 1.1



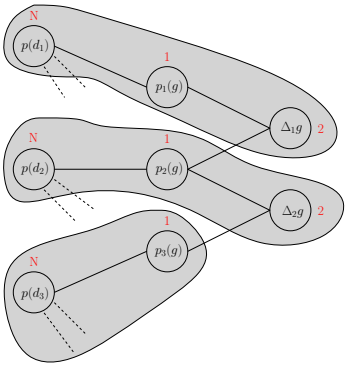
(b) Fall 1.2



(c) Fall 2.1



(d) Fall 2.2.1



(e) Fall 2.2.2

Abbildung 6.7: Fallunterscheidung für $g \in A$

Problem DECISION-W-MACP in der Komplexitätsklasse $\mathcal{NP}$ und mit Lemma 11 folgt das folgende Korollar.

Korollar 2. DECISION-W-MACP *ist $\mathcal{NP}$-vollständig auf planaren Graphen.*

6.3 u-MACP auf planaren Graphen

Es besteht weiterhin die Frage, ob das Problem DECISION-U-MACP auf planaren Graphen *$\mathcal{NP}$-vollständig* bleibt. Diese Frage ist insbesondere durch den Zusammenhang zwischen SURPRISE und U-MACP interessant. Im Folgenden soll dieser Frage nachgegangen werden, um sie letztendlich zu bejahen. Hierzu wird Reduktion 2 aus Abschnitt 6.2 leicht angepasst, indem Knoten $v \in G_{Red}(I)$, die ein Gewicht $w(v) > 1$ haben, durch planare Teilgraphen ersetzt werden, die bestimmte Eigenschaften bezüglich ihres minimalen Schnittes erfüllen.

Definition 20 (Kantenzusammenhang eines Graphen)**.** *Sei $G = (V, E)$ ein Graph. Dann heißt*

$$\lambda(G) = \min_{S \text{ ist Schnitt von } G} |S| \tag{6.4}$$

der Kantenzusammenhang *von G.*

Wenn für einen Graphen $\lambda(G) \geq k$ gilt, dann existiert kein Schnitt, der den Graphen in mehrere Zusammenhangskomponenten zerteilt, der weniger als k Kanten enthält. Reduktion 2 wird nun so angepasst, dass der erhaltene Graph nur Knoten mit Gewicht 1 enthält und trotzdem planar bleibt.

Reduktion 3. Ausgehend von einer Instanz $I = \big((A, B, C), T\big)$ von (2-3)-PLANARES 3-DM wird ein Graph $G'_{Red}(I)$ aufgebaut. Hierbei wird mit dem Graphen $G_{Red}(I)$, gemäß Reduktion 2 begonnen. Sei $\mathcal{H}_d$ ein ungewichteter planarer Graph mit N Knoten und

$\lambda(\mathcal{H}_d) \geq 4$. Ersetze jeden Knoten $p(d) \in G_{Red}(I)$ mit $d \in T$ durch eine Kopie des Graphen $\mathcal{H}_d$, lösche alle Kanten $\{p(g), p(d)\}$ mit $g \in A \cup B \cup C$ und ersetze sie durch neue Kanten $\{p(g), h_d\}$, wobei h_d ein beliebiger Knoten von $\mathcal{H}_d$ ist. Für alle Knoten $\Delta g \in G_{Red}(I)$ mit $w(\Delta g) = M \geq N$ verfahre analog mit Graphen $\mathcal{H}_{\Delta g}$ der Größe M. Zu jedem $g \in A_2$ wird ein Knoten κg und eine Kanten $\{\Delta g, \kappa g\}$ eingefügt. Für jedes Element $g \in A_3$ werden zwei Knoten $\kappa_1 g$ und $\kappa_2 g$, sowie Kanten $\{\Delta_1 g, \kappa_1 g\}$ und $\{\Delta_2 g, \kappa_2 g\}$ eingefügt. Schlussendlich werden alle Gewicht von Knoten im nun entstandenen Graphen $G'_{Red}(I)$ auf 1 gesetzt. Abbildung 6.8 zeigt die beschriebenen Ersetzungen. Die angegebene Reduktion ist polynomiell, da $N > 40\,|A| + 10\,|A_3|$ so gewählt werden kann, dass die Komponenten $\mathcal{H}_d$ und $\mathcal{H}_{\Delta g}$ nur polynomielle Größe bezüglich der Eingabe besitzen.

Bemerkung 3 (Existenz von Planaren Graphen mit minimalem Schnitt von 4)**.** *Abschnitt 6.3.1 stellt sicher, dass planare Graphen mit beliebig vielen Knoten existieren, die einen minimalen Schnitt von mindestens 4 haben.*

Lemma 12 (Planarität von $G'_{Red}(I)$)**.** *Für jede Instanz $I = \big((A, B, C), T\big)$ von* (2-3)-PLANARES 3-DM *ist $G'_{Red}(I)$ planar.*

Beweis. Sei $I = \big((A, B, C), T\big)$ eine Instanz von (2-3)-PLANARES 3-DM. Dann ist $G_{Red}(I)$ nach Lemma 9 planar. Des Weiteren ist jede Komponente $\mathcal{H}_d$ und $\mathcal{H}_{\Delta g}$ planar. Durch Ersetzen von Knoten $p(d)$ durch Komponenten $\mathcal{H}_d$ bleibt die Planarität von $G_{Red}(I)$ erhalten. Der planare Subgraph $\mathcal{H}_d$ werde so eingebettet, dass Knoten h_d auf der Äußeren Facette liegt. Betrachte zu einer beliebigen planaren Einbettung von $G_{Red}(I)$ eine Facette, in der Knoten $p(d)$ liegt. Dann kann $p(d)$ durch $\mathcal{H}_d$ derart ersetzt werden, dass h_d Knoten $p(d)$ ersetzt und $\mathcal{H}_d$ in dieser Facette liegt. Da h_d auf der äußeren Facette in $\mathcal{H}_d$ eingebettet ist, entstehen durch die Ersetzung keine Kantenkreuzungen. Wird diese Ersetzung für alle Komponenten $\mathcal{H}_d$ und analog für alle Komponenten Δg durchgeführt, entsteht eine planare Einbettung. Da das Einfügen von Knoten mit Grad 1 ebenfalls nichts an der Planarität eines Graphen verändert, gilt somit, dass $G'_{Red}(I)$ für alle Instanzen $I = \big((A, B, C), T\big)$ von (2-3)-PLANARES 3-DM ein planarer Graph ist. □

Analog zu schweren Knoten, ergibt sich die folgende Definition für deren entsprechenden Komponenten im durch Reduktion 3 erzeugten Graphen $G'_{Red}(I)$.

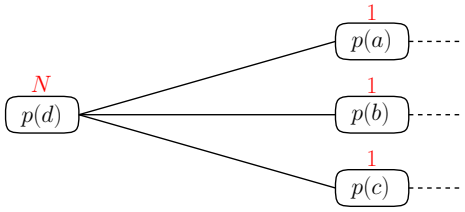
Definition 21 (schwere Komponente)**.** *Jede Komponente $\mathcal{H}_d$ bzw. $\mathcal{H}_{\Delta g}$ heißt* schwere Komponente*.*

Das folgende Lemma ergibt sich analog zu Lemma 10, indem Knoten durch Komponenten ersetzt werden.

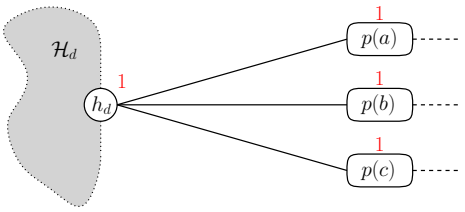
Lemma 13. *Für jede erfüllende Clusterung ζ des Problems* DECISION-U-MACP *auf $G'_{Red}(I)$ mit Parametern $K = 3\,|T| - |A|$ und $B = (4\,|A| + |A_3|)(N + 3)^2$, enthält jeder Cluster $C \in \zeta$ höchstens eine vollständige schwere Komponente.*

Um zu zeigen, dass DECISION-U-MACP *$\mathcal{NP}$-vollständig* ist, kann sich des Beweises zur *$\mathcal{NP}$-Vollständigkeit* von DECISION-W-MACP bedient werden. Ausgehend von einem perfekten dreidimensionalen Matching S kann auf dieselbe Weise eine Lösung von DECISION-U-MACP konstruiert werden, wie im Beweis zu Lemma 11. Dies führt zu folgendem Lemma.

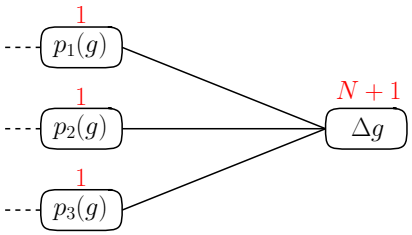
Lemma 14. *Sei S ein perfektes Matching der Instanz $I = \big((A, B, C), T\big)$ von* (2-3)-PLANARES 3-DM*. Dann enthält $G'_{Red}(I)$ eine erfüllende Clusterung ζ bezüglich des Problems* DECISION-U-MACP *mit Parametern $K = 3\,|T| - |A|$ und $B = (4\,|A| + |A_3|)(N + 3)^2$, die durch S induziert wird.*



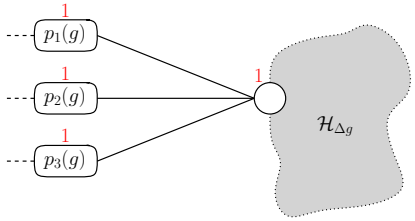
(a) Knoten $p(d)$ in $G_{Red}(I)$



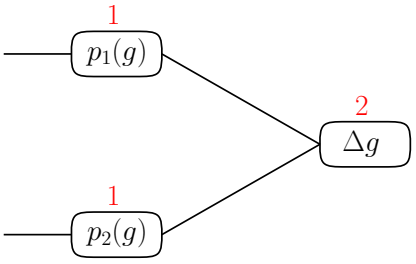
(b) Knoten h und Teilgraph $\mathcal{H}_d$ in $G'_{Red}(I)$



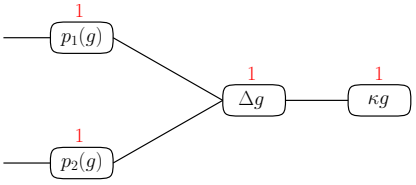
(c) Knoten $\Delta g \in G_{Red}(I)$ mit $g \in B \cup C$



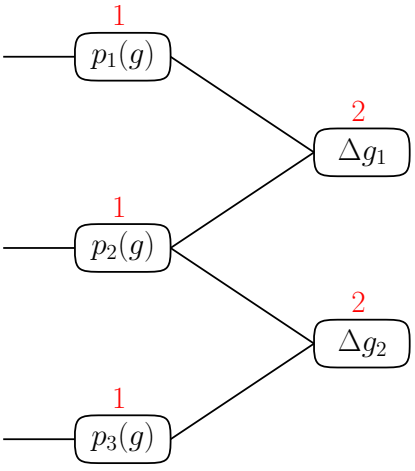
(d) Teilgraph $\mathcal{H}_{\Delta g} \in G'_{Red}(I)$ mit $g \in B \cup C$



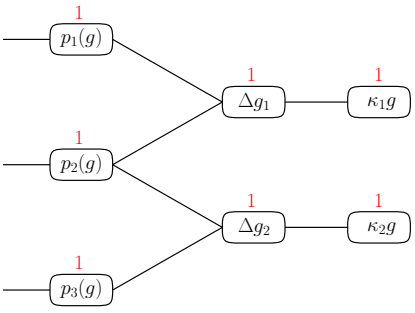
(e) Knoten $\Delta g \in G_{Red}(I)$ mit $g \in A_2$



(f) Knoten $\Delta g \in G'_{Red}(I)$ mit $g \in A_2$



(g) Knoten $\Delta g \in G_{Red}(I)$ mit $g \in A_3$



(h) Knoten $\Delta g \in G'_{Red}(I)$ mit $g \in A_3$

Abbildung 6.8: Die linke Seite zeigt Komponenten aus Reduktion 2, die rechte Seite die jeweilige Entsprechung in Reduktion 3.

Nachdem Lemma 13 bereits die Existenz von zwei schweren Komponenten pro Cluster ausschließt, bleibt für die Gegenrichtung zu zeigen, dass kein neu eingeführter Subgraph $\mathcal{H}_d$, bzw. $\mathcal{H}_{\Delta g}$ in einer erfüllenden Clusterung von U-MACP mit Parametern $K = 3\,|T| - |A|$ und $B = (4\,|A| + |A_3|)(N+3)^2$ durchschnitten wird.

Lemma 15. *Für jede erfüllende Clusterung ζ des Problems* DECISION-U-MACP *auf $G'_{Red}(I)$ mit Parametern $K = 3\,|T| - |A|$ und $B = (4\,|A| + |A_3|)(N+3)^2$ liegt jeder Subgraph $\mathcal{H}_d$, bzw. $\mathcal{H}_{\Delta g}$ in genau einem Cluster.*

Beweis. Im Folgenden heißen zwei schwere Komponenten *benachbart*, falls es einen Weg zwischen ihnen gibt, der keine weitere schwere Komponente durchquert. Ein solcher Weg zwischen benachbarten schwere Komponenten heißt *schnitt-erforderlich*.

Zunächst wird die Menge von kantendisjunkten schnitt-erforderlichen Wegen in $G'_{Red}(I)$ zwischen Komponenten $\mathcal{H}_d$ und $\mathcal{H}_{\Delta g}$ mit $g \in B \cup C$ betrachtet. Von jeder Komponente $\mathcal{H}_d$ führt je ein Weg zu einer Komponente $\mathcal{H}_{\Delta b}$, sowie einer Komponente $\mathcal{H}_{\Delta c}$. Zusätzlich existiert genau ein schnitt-erforderlicher Weg zwischen den zwei schweren Komponenten $\mathcal{H}_{d_i}$ und $\mathcal{H}_{d_j}$, die zu den Knoten $p_1(a)$ bzw. $p_2(a)$ mit $a \in A_2$ adjazent sind. Da alle diese Wege kantendisjunkt sind, müssen genau $2\,|T| + |A_2|$ Kanten geschnitten werden, um diese Wege zu trennen. Betrachte nun die verbleibenden "Sternkomponenten" (siehe Abb. 6.9) für Elemente $g \in A_3$. Jede dieser "Sternkomponenten" induziert drei Wege zwischen schweren Komponenten innerhalb dieser Komponente, es genügt aber zwei Kanten zu schneiden, um derartige Wege zu durchtrennen, weil sie nicht kantendisjunkt sind. Die Gesamtzahl an Kanten, die in "Sternkomponenten" durchtrennt werden müssen ist daher $2\,|A_3|$ und folglich müssen in $G'_{Red}(I)$ insgesamt mindestens

$$\begin{aligned} K' &= 2\,|T| + |A_2| + 2\,|A_3| \\ &= 2\,|T| + |A| + \underbrace{|A_3|}_{|T|-2|A|} \\ &= 2\,|T| + |A| + |T| - 2\,|A| \\ &= 3\,|T| - |A| \end{aligned}$$

Kanten geschnitten werden, um alle schweren Komponenten von einander zu trennen. Sei R die Menge aller durchtrennten schweren Komponenten in einer erfüllenden Clusterung von U-MACP auf $G'_{Red}(I)$. Es gelte weiterhin $R = R_1 \cup R_2$. Hierbei enthalte R_1 die Komponenten $\mathcal{H}_{d_i}$, sodass $d_i = (a_i, b_i, c_i) \in T$ mit $a_i \in A_2, b_i \in B, c_i \in C$ gilt, sowie alle schweren Komponenten $\mathcal{H}_{\Delta_g}$. Die Menge R_2 enthalte die Komponenten $\mathcal{H}_{d_j}$ mit $d_j = (a_j, b_j, c_j) \in T$, sodass $a_j \in A_3, b_j \in B, c_j \in C$ gilt.

Annahme: $|R| > 0$.
Durch Schneiden der Komponenten in R_1 werden höchstens drei schnitt-erforderliche Wege in $G'_{Red}(I)$ durchtrennt. Für alle Komponenten in R_2 ergeben sich folgenden Möglichkeiten.

- **Fall 1:** Eine schwere Komponente wird durchschnitten. Dann verbleibt nur ein Weg zwischen schweren Komponenten in der "Sternkomponente". Das heißt das Durchschneiden der schweren Komponente bewirkt, dass nur noch ein Weg zwischen schweren Komponenten getrennt werden muss. Für das Durchschneiden der schweren Komponente sind mindestens 4 Kanten nötig.
- **Fall 2:** Zwei oder drei schwere Komponenten werden durchschnitten. Es verbleibt kein Weg mehr zwischen zwei schweren Komponenten in der "Sternkomponente". Für das Durchschneiden der schweren Komponenten sind mindestens 8 Kanten nötig.

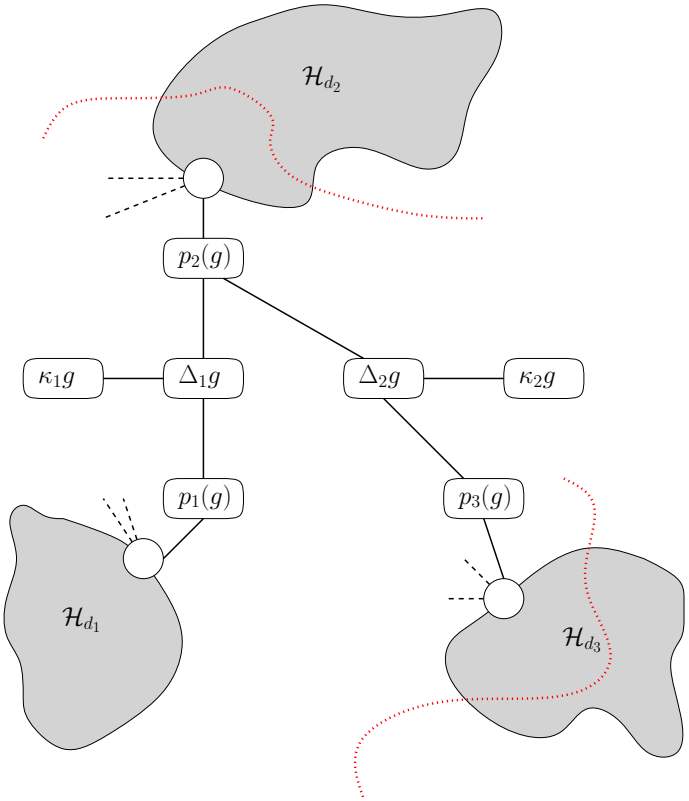


Abbildung 6.9: "Sternkomponente" zu $g \in A_3$. Wird eine beliebige Komponente $\mathcal{H}_{d_i}$ geschnitten, so geht gesamtheitlich betrachtet höchstens 1 Weg zwischen schweren Komponenten verloren.

Werden Fall 1 und 2 in der Gesamtheit betrachtet, so wird durch das Durchschneiden pro schwerer Komponente $\mathcal{H}_d$ innerhalb einer "Sternkomponente" höchstens der Verlust eines zu schneidenden Weges, außerhalb höchstens der Verlust zweier Wege zu Komponenten $\mathcal{H}_{\Delta b}$ und $\mathcal{H}_{\Delta c}$ bedingt, wobei pro Schnitt durch eine schwere Komponente mindestens 4 Kanten geschnitten werden. Insgesamt verbleiben im Graphen $G'_{Red}(I)$ also mindestens

$$\begin{aligned} & 3\,|T| - |A| - 3\,|R_1| - 3\,|R_2| \\ =& 3\,|T| - |A| - 3\big(\,|R_1| + |R_2|\,\big) \\ =& 3\,|T| - |A| - 3\,|R| \end{aligned}$$

Wege zwischen schweren Komponenten, die durchtrennt werden müssen. Da jede schwere Komponente in $G'_{Red}(I)$ per Definition aber einen minimalen Schnitt von 4 hat, sind nur noch höchstens $3\,|T| - |A| - 4\,|R|$ Kanten übrig, die geschnitten werden können, ein Widerspruch zur Tatsache, dass in einem Cluster nicht mehrere vollständige schwere Komponenten liegen können(vgl. Lemma 13). Folglich ist die Annahme $|R| > 0$ falsch und es gilt die Behauptung. □

Zusätzlich können die Kanten $\{\kappa g, \Delta g\}$, bzw. $\{\kappa_i g, \Delta_i g\}$ für alle $g \in A$ nicht durchschnitten werden, da sonst ein Cluster existiert, der nur aus Knoten κg, bzw. $\kappa_i g$ besteht, was zu einem Widerspruch führt. Nach Lemma 13 und Lemma 15 ist damit klar, dass sich die für eine Instanz von DECISION-U-MACP neu eingefügten Komponenten wie die entsprechenden Knoten in Reduktion 2 für eine Instanz von DECISION-W-MACP verhalten. In einer Lösung muss jede schwere Komponente in einem eigenen Cluster liegen und es darf keine Komponente zerschnitten werden. Somit lässt sich die gleiche Argumentation

wie im Beweis zu Lemma 11 für DECISION-W-MACP anwenden, woraus sich folgendes Korollar ergibt.

Korollar 3. *Das Problem* DECISION-U-MACP *ist $\mathcal{NP}$-vollständig auf planaren Graphen.*

6.3.1 Existenz 4-verbundener planarer Graphen

Ein planarer Graph mit $N \geq 6$ Knoten und einem minimalen Schnitt von mindestens 4, wie er in Reduktion 3 benötigt wird, lässt sich wie folgt konstruieren.

$H_N = (V, E)$ mit $V = \{links, rechts\} \cup \{1, \ldots, N-2\}$ und $E = \big\{\{i, i+1\} \mid 1 \leq i \leq N-2\big\} \cup \big\{\{links, i\} \mid 1 \leq i \leq N-2\big\} \cup \big\{\{rechts, i\} \mid 1 \leq i \leq N-2\big\} \cup \{1, N-2\}$

Abbildung 6.10 zeigt den beschriebenen Graphen H_N für den Basisfall $N = 6$ und für $N > 6$.

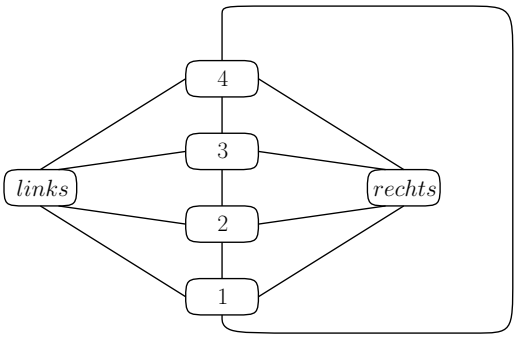
Lemma 16. *Der beschriebene Graph H_N ist planar und es gilt $\lambda(H_N) \geq 4$ für alle $N > 6$.*

Beweis. Die Planarität des Graphen folgt unmittelbar aus Abbildung 6.10b, in der der Graph schematisch planar dargestellt ist.

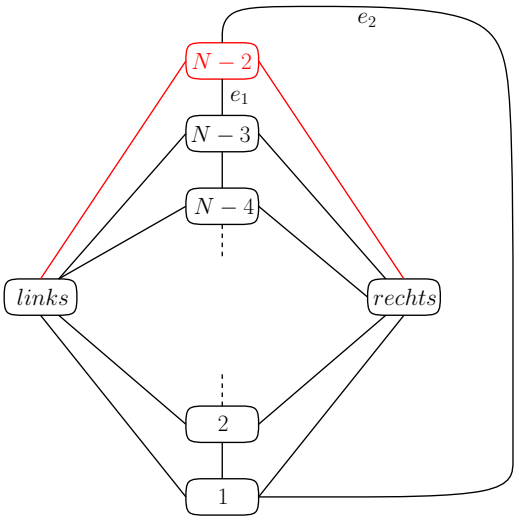
Zeige im Folgenden induktiv, dass $\lambda(H_N) \geq 4$ für alle $N \geq 6$ gilt. Sei zunächst $N = 6$. Betrachte den Dualgraphen H_6^* zu H_6. offensichtlich hat der in Abbildung 6.10c dargestellte Graph nur Kreise, die eine Länge von mindestens 4 haben. Somit gilt die Behauptung für $N = 6$.

Sei $N > 6$ beliebig, aber fest gewählt, H_N planar und es gelte $\lambda(H_N) \geq 4$. Betrachte einen beliebigen nichtleerer Schnitt S im Graphen H_{N+1}. Der neu eingefügte Knoten $N-1$ sei wie in Abbildung 6.10b rot gefärbt. Es gibt folgende Möglichkeiten:

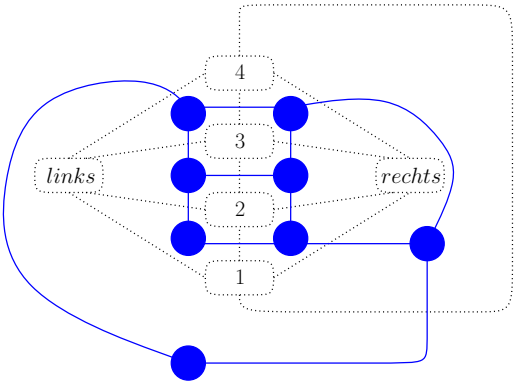
- **Fall 1:** S trennt schwarze Knoten. Es seien $e_1 = \{N-2, N-3\}, e_2 = \{N-2, 1\}$
  - **Fall 1.1:** Weder e_1 noch e_2 werden geschnitten.
    - Mindestens eine der roten Kanten $\{N-2, links\}, \{N-2, rechts\}$ ist in S enthalten. Dann muss S den schwarzen Subgraphen, der durch die Knoten $\{1, \ldots, N-3, links, rechts\}$ induziert wird, schneiden. Nach Induktionsvoraussetzung ist der minimale Schnitt des schwarzen Subgraphen mindestens 3, womit die Gesamtzahl der Kanten in S also mindestens 4 beträgt.
    - Keine der Kanten $\{N-2, links\}, \{N-2, rechts\}$ ist in S enthalten. Dann liegt die Menge von Knoten $\{links, rechts, N-2, N-3, 1\}$ in einer, durch den Schnitt induzierten Zerlegung des Graphen in Zusammenhangskomponten, in einer Zusammenhangskomponente A. Betrachte einen Knoten k mit $2 \leq k \leq N-4$, der in einer anderen Zusammenhangskomponente liegt. S muss die Kanten $\{links, k\}$ und $\{rechts, k\}$ enthalten. Außerdem müssen alle Wege von Knoten k zu Zusammenhangkomponente A durchtrennt werden. Da mindestens zwei solche Wege $(k, k+1, \ldots N-3)$ und $(k, k-1, \ldots 1)$ existieren, die disjunkt sind, muss der Schnitt durch den Graphen mindestens Größe 4 haben.
  - **Fall 1.2:** Mindestens eine der Kanten e_1 oder e_2 wird geschnitten. Da S schwarze Knoten trennt, muss der Schnitt S den Knoteninduzierten Subgraphen der Knoten $1, \ldots, N-3, links, rechts$ schneiden. Dieser Subgraph hat nach Induktionsvoraussetzung einen minimalen Schnitt von mindestens 3 Kanten, da er mit H_N bis auf eine Kante übereinstimmt. Zusammen mit der bereits im Schnitt enthaltenen Kante e_1 bzw. e_2, enthält der Schnitt demnach mehr als 4 Kanten.



(a) Graph H_6.



(b) Schema für den Graphen H_N für allgemeines $N > 6$. Der rote Knoten $N-2$ ist der als Unterteilung der Kante $\{N-3, 1\}$ in H_{N-1} neu eingefügte Knoten.



(c) Graph H_6 und sein Dualgraph H_6^* in blau dargestellt.

Abbildung 6.10: Illustration zum Beweis von Lemma 16 und Konstruktion von H_N.

- **Fall 2:** S trennt keine schwarzen Knoten. Dann muss der Schnitt im Graphen den roten Knoten $N-1$ vom Rest des Graphen trennen. Da $grad(N-1) = 4$ ist, gilt die Behauptung.

□

7. Zusammenfassung und Ausblick

Sowohl das gewichtete als auch das ungewichtete Minimum-(Average)-Contamination-Problem ist $\mathcal{NP}$-*schwer* auf planaren Graphen. Auf Bäumen kann W-MACP in $O(n^5 \cdot w_{max}^2)$ für alle Werte von K gelöst werden, wobei w_{max} das Maximalgewicht eines Knotens im betreffenden Baum ist. Für U-MACP auf Bäumen hat der angegebene Algorithmus daher eine Laufzeit von $O(n^5)$. Weil die Probleme MinIP und U-MACP bezüglich optimaler Lösungen äquivalent sind und MinIP ein Teilproblem von Surprise darstellt, lässt sich auch der Wert einer Surprise-optimalen Clusterung auf Bäumen in $O(n^5)$ finden. Auf Pfaden und Kreisen lässt sich W-MACP in $O(n^3)$, bzw. $O(n^4)$ für alle möglichen Werte von K lösen und der Wert einer Surprise-optimalen Clusterung lässt sich sowohl auf Pfaden, als auch auf Kreisen, in Zeit $O(n^3)$ finden.

Eine offene Frage bleibt, ob das Problem, Surprise auf planaren Graphen zu optimieren, $\mathcal{NP}$-*schwer* ist. Auch wenn dies für das Teilproblem MinIP gilt, lassen sich daraus nicht unmittelbar Aussagen über die Komplexität von Surprise treffen. Falls es einen Algorithmus geben sollte, der Surprise auf planaren Graphen in polynomieller Zeit optimiert, so kann dieser nicht darauf aufbauen das Teilproblem MinIP zu lösen, um eine Lösung von Surprise zu berechnen. Folglich ist die Existenz eines derartigen Algorithmus äußerst unwahrscheinlich. Dieselbe Frage stellt sich zur Komplexität des beschriebenen Problems K-MSSV. Da U-MACP als Teilproblem von W-MACP $\mathcal{NP}$-*schwer* auf planaren Graphen ist, liegt die Vermutung nahe, dass dies auch für K-MSSV gelten könnte. Ungeachtet von dieser Vermutung, steht die erforderliche Reduktion mit anschließendem Beweis zur $\mathcal{NP}$-*Schwere* von K-MSSV noch aus.

Ebenfalls liegt die Vermutung nahe, dass sich die in Kapitel 5 für Bäume angegebenen dynamischen Algorithmen, auf andere Klassen von Graphen erweitern lassen. Für serienparallele Graphen und andere Graphen mit konstanter Baumbreite könnte [BK08] hier Anhaltspunkte geben, wie eine derartige Erweiterung auszusehen hat.

Literaturverzeichnis

[ACSG01] Alex Aletà, Josep M. Codina, Jesús Sánchez und Antonio González: *Graph-partitioning based instruction scheduling for clustered processors.* In: *Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture*, MICRO 34, Seiten 150–159, 2001, ISBN 0-7695-1369-7.

[ACY06] James Aspnes, Kevin Chang und Aleksandr Yampolskiy: *Inoculation strategies for victims of viruses and the sum-of-squares partition problem.* Journal of Computer and System Sciences, 72(6):1077–1093, 2006.

[AM11] Rodrigo Aldecoa und Ignacio Marín: *Deciphering Network Community Structure by Surprise.* PLOS ONE, 6, September 2011.

[AM13] Rodrigo Aldecoa und Ignacio Marín: *Surprise maximization reveals the community structure of complex networks.* Scientific reports, 3, 2013.

[AMM05] Vicente Arnau, Sergio Mars und Ignacio Marín: *Iterative cluster analysis of protein interaction data.* Bioinformatics, 21(3):364–378, 2005.

[BBC04] Nikhil Bansal, Avrim Blum und Shuchi Chawla: *Correlation Clustering.* Machine Learning, 56(1-3), 2004, ISSN 0885-6125.

[BDG+07] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski und Dorothea Wagner: *On finding graph clusterings with maximum modularity.* In: *Graph-Theoretic Concepts in Computer Science*, Seiten 121–132. Springer, 2007.

[BGLL08] Vincent D Blondel, Jean Loup Guillaume, Renaud Lambiotte und Etienne Lefebvre: *Fast unfolding of communities in large networks.* Journal of Statistical Mechanics: Theory and Experiment, 2008(10):P10008, 2008.

[BK08] Hans L Bodlaender und Arie MCA Koster: *Combinatorial optimization on graphs of bounded treewidth.* The Computer Journal, 51(3):255–269, 2008.

[CC03] Miroslav Chlebík und Janka Chlebíková: *Approximation hardness for small occurrence instances of NP-hard problems.* In: *Algorithms and Complexity*, Seiten 152–164. Springer, 2003.

[CNM04] Aaron Clauset, Mark EJ Newman und Cristopher Moore: *Finding community structure in very large networks.* Physical review E, 70(6):066111, 2004.

[DF86] Martin E. Dyer und Alan M. Frieze: *Planar 3DM is NP-complete.* Journal of Algorithms, 7(2):174–184, 1986.

[DT] T.N. Dinh und M.T. Thai: *Towards Optimal Community Detection: From Trees to General Weighted Networks.* Internet Mathematics. accepted pending version.

[FKW14] Tobias Fleck, Andrea Kappes und Dorothea Wagner: *Graph Clustering with Surprise: Complexity and Exact Solutions.* In: *Proceedings of the 40th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'14)*, Lecture Notes in Computer Science. Springer, 2014. to appear.

[For10] Santo Fortunato: *Community detection in graphs.* Physics Reports, 486(3):75–174, 2010.

[GSW11] Robert Görke, Andrea Schumm und Dorothea Wagner: *Density-Constrained Graph Clustering.* In: Frank Dehne, John Iacono und Jörg Rüdiger Sack (Herausgeber): *Algorithms and Data Structures*, Band 6844 der Reihe *Lecture Notes in Computer Science*, Seiten 679–690. Springer Berlin Heidelberg, 2011, ISBN 978-3-642-22299-3.

[KM77] Sukhamay Kundu und Jayadev Midra: *A Linear Tree Partitioning Algorithm.* SIAM Journal on Computing, 6(1, March):151–154, 1977.

[KSM08] Masahiro Kimura, Kazumi Saito und Hiroshi Motoda: *Minimizing the Spread of Contamination by Blocking Links in a Network.* In: *AAAI*, Band 8, Seiten 1175–1180, 2008.

[Kur] Casimir Kuratowski: *Sur le problème des courbes gauches en Topologie.* Fundamenta Mathematicae, (1):271–283.

[LT11] Angsheng Li und Linqing Tang: *The complexity and approximability of minimum contamination problems.* In: *Theory and Applications of Models of Computation*, Seiten 298–307. Springer, 2011.

[Mak97] Yury Makarychev: *A short proof of Kuratowski's graph planarity criterion.* Journal of Graph Theory, 25(2):129–131, 1997.

[New04] Mark EJ Newman: *Analysis of weighted networks.* Physical Review E, 70(5):056131, 2004.

[NG04] Mark EJ Newman und Michelle Girvan: *Finding and evaluating community structure in networks.* Physical review E, 69(2):026113, 2004.