FernUniversität in Hagen

Fakultät für Mathematik und Informatik

Fair Correlation Clustering in Kaktusgraphen

Masterthesis
Master of Science
Praktische Informatik

Autor: Moritz Hiksch
Betreuer: Dr. Winfried Hochstättler
Lehrstuhl: Diskrete Mathematik
Abgabedatum: 26.01.2024

Abstract

This thesis is about the limits of the computatnional complexity of Fair Correlation Clustering. Fair Correlation Clustering is a generalization of the Correlation Clustering problem. The Correlation Clustering problem is a problem of pattern matching in which a set is partitioned in a way, that similar objects are clustered together and dissimilar are clustered apart. The Fair Correlation Clustering adds the requirement, that the color ratio of the set must be transferred to the clusters. The requirement of fairness arises from the increasing applications of clustering algorithms within modern society and the concern about discrimination that could be caused by those practices. Fair Correlation Clustering provides a solution to those concerns, but is not efficiently computable.

A recent publication [12] provided crucial insights and showed that problem instances in form of forests with two colors in a ratio of 2:1 are efficiently solvable. In this thesis a specialization of the problem is defined that regards inputs in form of cactus graphs, colored with only two colors and a coloring ratio of 2:1. An algorithm is designed to solve the problem and on the algorithm's basis it is proven, that the problem is computable in polynomial time. Efficiently solvable problem instances are thus not restricted to forests, but reach at least into the territory of cactus graphs. Based on the algorithm for the specialized problem an algorithm for the more general problem of Fair Correlation Clustering for inputs in the form of cactus graphs with an arbitrary amount of colors as well as an arbitrary coloring ratio is formulated. It is proven, that the general problem is not solvable in polynomial time.

Zusammenfassung

In dieser Arbeit werden die Grenzen der Komplexität des Fair Correlation Clusterings erforscht. Fair Correlation Clustering ist eine Generalisierung des Correlation Clusteringproblems. Beim Correlation Clustering wird nach einer Partition einer Menge gesucht, in welcher ähnliche Objekte zusammen gruppiert werden und unähnliche in unterschiedliche Cluster eingeteilt werden. Das Fair Correlation Clustering verarbeitet gefärbte Mengen und fordert, dass die Farbverhältnisse auf die Cluster übertragen werden. Diese Anforderung entstammt dem zunehmenden Einfluss von Clustering Algorithmen in Verbindung mit maschinellem Lernen auf die Gesellschaft, sowie der damit verbundenen Sorge um Diskriminierung. Das Fair Correlation Clustering bietet einen Ansatz zur Lösung dieser Sorgen, ist aber nicht effizient lösbar.

Die kürzlich veröffentlichte Publikation von Casel et. al. [12] zeigt, dass Probleminstanzen in Form von Wäldern mit zwei Farben in einem Verhältnis von 1:2 effizient lösbar sind. In dieser Arbeit wird eine Spezialisierung des Fair Correlation Clustering definiert, welche sich auf Eingaben in Form von Kaktusgraphen mit zwei Farben in einem Verhältnis von 2:1 beschränkt. Für dieses Problem wird ein Algorithmus basierend auf einem dynamischen Programm formuliert. Auf Basis dieses Algorithmus wird bewiesen, dass auch dieses Problem in polynomieller Zeit lösbar ist. Effizient lösbare Probleminstanzen sind somit nicht auf Wälder begrenzt, sondern erstrecken sich mindestens auf Kaktusgraphen. Der Algorithmus wird auf das generellere Problem des Fair Correlation Clustering mit Beschränkung auf Kaktusgraphen mit beliebig vielen Farben in beliebigem Verhältnis erweitert und gezeigt, dass dieser Algorithmus nicht in polynomieller Zeit ausführbar ist.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

In den Vereinigten Staaten wird die wählende Bevölkerung variierend nach Staat ca. alle 10 Jahre, nach der Volkszählung (engl. Census) in neue Wahlkreise eingeteilt. Bei Wahlen wird die Bevölkerung eines Wahlkreises durch genau eine Stimme repräsentiert. Die Stimme des Wahlkreises wird durch einfache Mehrheit bestimmt. Die Aufgabe, die Wählerschaft in neue Wahlkreise zu gruppieren, wird von der Partei übernommen, die zu diesem Zeitpunkt an der Macht ist.

Bei der sogenannten Wahlkreisschiebung (engl. Gerrymandering) werden Regionen so in Wahlkreise aufgeteilt, dass eine Partei allein durch die Gruppierung der Bürger einen Vorteil erhält. Dazu werden anhand von Stichproben Prognosen für die Stimmen von Nachbarschaften erstellt. Mithilfe dieser Prognosen wird die Bevölkerung so gruppiert, dass die Wähler anderer Parteien durch eine geringere Anzahl an Stimmen repräsentiert wird. In der Praxis kommen dazu verschiedene Methoden zum Einsatz. Eine dieser Methoden zielt darauf ab, die Wähler der eigenen Partei in möglichst viele kleine Wahlkreise einzuteilen, sodass diesen möglichst viele Stimmen zugeordnet werden und die Wähler anderer Parteien in möglichst wenige große Bezirke, sodass diesen weniger Stimmen zugeordnet werden, auch wenn sie evtl. einen größeren Anteil der Bevölkerung darstellen.

Das Problem der Wahlkreisschiebung beruht darauf, dass außer dem geographischen Zusammenhang von Wahlkreisen für die Gruppierung nicht viele Anforderungen gestellt werden. Damit ist die Definition von Wahlbezirken offen für nahezu jede erdenkliche Größe und Form. Wahlkreisschiebung findet nicht nur in den Vereinigten Staaten statt, sondern auch in anderen Ländern, wie Großbritannien, Frankreich und Deutschland. Um die Wahlkreisschiebung hat sich eine regelrechte Industrie gebildet. In dieser werden professionelle statistische Auswertungen und Methoden der sogenannten Clusteranalyse verwendet, um die Gruppierung der Bevölkerung zu den eigenen Gunsten zu optimieren. [40][34][26]

Die Klassifizierung oder Gruppierung von Objekten wird als Clustering (dt. Gruppierung) bezeichnet. In der Disziplin der Clusteranalyse werden verschiedene Definitionen des Clusterings behandelt, die für verschiedene Anwendungszweck geeignet sind. Die Definitionen von Clusteringproblemen können auf verschiedene Arten unterteilt werden. Sie lassen sich bspw. danach einteilen, wie vielen Clustern ein Datenobjekt zugeordnet werden kann. Einerseits gibt es Probleme, bei deren Lösungen jedes Objekt genau einem Cluster zugeordnet wird, diese werden als "hard-clustering"bezeichnet. Andererseits gibt es solche, bei denen sie mehreren Clustern zugeordnet werden können, was als "soft-clustering"bezeichnet wird. Das soft-clustering kann noch weiter unterteilt werden, in Statistische, Hierarchische und Fuzzy Clusterings. Beim Statistischen Clustering wird zu jedem gegebenen Datenobjekt/-punkt x und jedem Cluster C_i ein Wert $P(C_i|x) \in [0,1]$ angegeben, welcher die Wahrscheinlichkeit angibt, dass dieser Punkt diesem Cluster zugeordnet wird. Beim Hierarchischen Clustering werden Datenobjekte hierarchisch strukturierten Gruppen zugeordnet, so dass ein Objekt

einem Cluster und automatisch jedem hierarchisch darüber liegenden Cluster zugeordnet wird. Ein Beispiel dafür ist die biologische Taxonomie, bei welcher Lebewesen einer Art und damit automatisch einer Gattung, einer Familie, einer Ordnung, einer Klasse, einem Stamm, einem Reich und einer Domäne zugeordnet werden. Beim Fuzzy-Clustering werden die Objekte anteilig Clustern zugeordnet. Dadurch können sich Cluster überschneiden. [20][31][23]

Es gibt über hundert Algorithmen zur Lösung von Clusteringproblemen. Jede Definition und jede Gruppierung kann in gewissem Kontext eine Berechtigung aufweisen. Welche Definition des Clusterings verwendet wird, wird in Abhängigkeit des Kontextes vom Anwender entschieden. Die Wahl des Clusteringproblems stellt eine potentielle Quelle für gezielten oder unbeabsichtigten Bias (dt. Verzerrung von Ergebnissen) dar. Die meisten Algorithmen verwenden zudem Parameter, die ebenfalls vom Anwender vorgegeben werden müssen. Unter anderem können die Parameter angeben wie viele Objekte in einer Gruppe enthalten sein dürfen oder wie viele Gruppen es insgesamt geben soll. Als wären das nicht genug Möglichkeiten zur Einflussnahme, werden in der Praxis oftmals mehrere Ergebnisse berechnet und aus diesen das "aussagekräftigste" Ergebnis ausgesucht. [33][23][44][12][3]

In der Publikation "Clustering: Science or Art?"[33] wird die Diskussion über die Existenz natürlicher absoluter Gruppierungen aufgegriffen. Der Einfluss des Anwenders wird als Argument genutzt, um die Clusteranalyse von der Wissenschaft zur Kunstform zu degradieren. Am Ende der Publikation wird die Bedeutung einer solchen Kategorisierung relativiert. Die Kritik an der Praxis des Clusterings bleibt jedoch bestehen. Eine einheitliche Definition und einheitliche Vorgehensweisen könnten dabei helfen, dem Gebiet mehr Legitimität zu verleihen und Problemen wie der Wahlkreisschiebung entgegen zu wirken.

Um Ergebnisse nachvollziehbarer zu gestalten, wird eine einheitliche Definition des Clusteringproblems benötigt. Alle Definitionen des Clusterings haben eine Gemeinsamkeit: Das Kernziel jedes Clusteringproblems besteht darin, die Datenobjekte so zu gruppieren, dass die Objekte eines Clusters möglichst "ähnlich" zueinander sind und möglichst "unterschiedlich" zu Objekten anderer Gruppen. Durch diese Gemeinsamkeit lässt sich eine Generalisierung aller Clusteringprobleme definieren. Die Generalisierung ist bekannt unter dem Namen Correlation Clustering. [1][12][31]

In Kapitel 1.1.6 wird die formale Definition zum Problem des Correlation Clusterings gegeben. Bei diesem Problem werden abstrakte Datenobjekte zusammen mit einem Maß für ihre paarweise Ähnlichkeit bzw. Unterschiedlichkeit als Eingabe vorgegeben und eine Gruppierung als Ergebnis bestimmt. Außer dieser Eingabe werden keinerlei Parameter vorgegeben. Der Einfluss des Anwenders wird dadurch auf die Definition des Begriffs "Ähnlichkeit" beschränkt. Potentielle Quellen für Bias durch die Wahl von Problemdefinition, Parametern und die gezielte Selektion von Ergebnissen werden durch das Correlation Clustering umgangen. [44][12][3]

Für das Correlation Clustering bestehen jedoch eigene Herausforderungen. In der Mathematik wird meist nach Algorithmen zur Lösung möglichst genereller Probleme gesucht, da diese sich zur Lösung aller Spezialfälle verwenden lassen. Damit ist der Aufwand zur Erstellung von Algorithmen geringer, weil nicht für jede Form von Eingaben ein Algorithmus gefunden werden muss. Der Nachteil besteht darin, dass generellere Algorithmen niemals effizienter sein können als Algorithmen zur Lösung von Spezialisierungen. Das ist auch bei den Clusteringproblemen der Fall. Viele Clusteringprobleme sind effizient lösbar. Das Correlation Clustering ist nicht effizient lösbar. Stattdessen müssen in der Praxis Approximationsalgorithmen genutzt werden, welche Näherungen an exakte Lösungen berechnen. In Kapitel 1.1.3 wird eine Einführung zur Mathematik bezüglich des Ressourcenaufwands zur Lösung von Problemen gegeben. Kapitel 1.3.1 beleuchtet die Forschung rund um das Correlation Clustering, welche sich hauptsächlich auf die Verbesserung des Ressourcenaufwands fokussiert.

Der Ressourcenaufwand ist allerdings nicht das einzige Problem des Correlation Clusterings. Clusteringprobleme werden im Bereich des maschinellen Lernens verwendet. Sie stellen einen Grundpfeiler des maschinellen Lernens dar. Clusterings werden verwendet, um in Datensätzen unbekannte Muster in Form von Gruppierungen zu entdecken. Das Clustering wird daher im Kontext des maschinellen Lernens auch als Pattern Recognition bezeichnet und in die Klasse der unüberwachten Lernalgorithmen eingeordnet. Mit Hilfe der Algorithmen des unüberwachten Lernens werden Muster in Daten gesucht und dies mit so wenig menschlichem Zutun wie möglich. Diese Verfahren werden häufig zur Vorverarbeitung von Daten verwendet, die Ergebnisse können dann für das Training von überwachten Lernalgorithmen genutzt werden. Die überwachten Lernalgorithmen approximieren das Clustering meist mit dem Ziel neue, bisher unbekannte Datenobjekte klassifizieren zu können.

Maschinelles Lernen zieht langsam aber sicher in allen Bereichen des menschlichen Lebens ein. Clusteringalgorithmen werden vermehrt zur Klassifizierung von Personen verwendet. Die Wahlkreisschiebung ist dabei nur eines von vielen Beispielen für die Anwendung von Clusteringalorithmen unter Verwendung personenbezogener Daten. Weitere Anwendungszwecke sind die Verarbeitung von Versicherungsansprüchen im Bereich der Medizin, die Verteilung von Ressourcen beim Städte- oder Häuserbau, die Klassifizierung von Menschen in Gefahrenstufen, wie bspw. an Flughäfen, oder auch in Gefängnissen zur Entscheidungsfindung bei Anträgen auf frühzeitige Entlassung. [31][35][36][41][43][2]

Aufgrund des unausweichlichen gesellschaftlichen Einflusses werden nicht nur vermehrt Aufrufe für Nachvollziehbarkeit laut, sondern auch solche für mehr Fairness. Genauer ausgedrückt wird die intensivere Behandlung von statistischem Bias gefordert und die Garantie, dass verwendete Algorithmen frei von Diskriminierung sind. Kapitel 1.3.2 gibt eine Einführung in den aktuellen Diskurs zu Fairness und Ansätze zur Definition algorithmischer Fairness.

Zwar ist die Vereinheitlichung der Clusteringprobleme ein Schritt, der zu einfacher nachvollziehbaren und damit einfacher überprüfbaren Ergebnissen führt, jedoch werden die Anforderungen an algorithmische Fairness, wie sie in Kapitel 1.3.2 definiert sind, in der Problemdefinition des Correlation Clusterings nicht berücksichtigt. Aus diesem Grund wurde das Fair Correlation Clustering entwickelt. Eine formale Definition wird in 1.1.6 gegeben. Das Fair Correlation Clustering stellt eine Generalisierung des Correlation Clusterings dar und kann somit keinen effizienteren Lösungsweg aufweisen, als das Correlation Clustering selbst.

Diese Arbeit widmet sich genau diesem Problem. Das Fair Correlation Clustering ist ein besonders schwieriges Problem, für das es keine effiziente Lösung gibt und für das bekannt ist, dass es keine effiziente Lösung geben kann. Die Aufgabe dieser Arbeit besteht darin, die Grenzen der effizient lösbaren Probleminstanzen zu erforschen. Casel, Friedrich, Schirneck und Wietheger zeigen in [12], dass Probleminstanzen, die auf Wälder beschränkt sind und zwei Farben im Verhältnis 2:1 effizient lösbar sind. Die Aufgabe dieser Arbeit besteht darin zu überprüfen, ob die Grenzen der effizient lösbaren Probleminstanzen über Wälder hinausgeht und auch Kaktusgraphen mit zwei Farben im Verhältnis 2:1 effizient lösbar sind. Dazu werden die von Casel et. al. [12] vorgestellten Algorithmen auf Kaktusgraphen erweitert und deren Komplexität hergeleitet. Dazu werden die Ergebnisse von Buchin [10] verwendet, welche dynamische Programmierung auf Kaktusgraphen anwendet, um ein anderes Partitionsproblem zu lösen. Die Algorithmen von Casel et. al. werden in Kapitel 2.1 vorgestellt und die Erweiterungen sowie deren Komplexitätsgrenzen in Kapitel 2.2. Dynamische Programmierung wird in Kapitel 1.1.5 erläutert.

Das Ziel besteht darin die Grenzen des Ressourcenaufwands genauer zu bestimmen. Daher werden neben der mathematischen Herleitung der Algorithmen auch deren Komplexitätsgrenzen hergeleitet. Im Zuge der Herleitung der Grenzen für die neuen Algorithmen werden auch genauere Grenzen hergeleitet als die in [12] vorgestellten.

Damit diese Arbeit möglichst unabhängig von den mathematischen Kenntnissen des Lesers ist, wird in Kapitel 1.1 eine Einführung in die hierfür wichtigsten mathematischen Bereiche gegeben. Danach wird im Kapitel 1.3 der aktuelle Stand der Wissenschaft präsentiert. In Kapitel 2 wird der Hauptteil der Arbeit vorgestellt, gefolgt vom Fazit in Kapitel 3.

Am Ende jedes Kapitels ist unten rechts ein Link zum Inhaltsverzeichnis angegeben.

Inhaltsverzeichnis

1.1 Grundlagen

In diesem Kapitel werden die Symbole und Notationen erläutert, die in dieser Arbeit verwendet werden. Nach dieser Einführung sollte die Basis zum Verständnis der nachfolgenden Kapitel gegeben sein. In Kapitel 1.1.1 werden die Grundlagen zur Mengenlehre, Logik und Abbildungen vermittelt, in Kapitel 1.1.2 folgen Notationen, die weniger geläufig sind und sich auf diese Arbeit und die von Casel et. al. [12] beschränken. Das Kapitel 1.1.3 ist eine kurze Einführung in die Komplexitätstheorie. Dem folgt in Kapitel 1.1.4 eine kurze Einführung in die Grundlagen der Graphentheorie. Danach wird in Kapitel 1.1.5 das dynamische Programm zur Anwendung auf Bäumen beschrieben.

Inhaltsverzeichnis

1.1.1 Mengenlehre, Logik und Abbildungen

Purkert et. al. bezeichnen in [37] die Mengenlehre zusammen mit der Logik als Grundstein der Mathematik. Der Hauptteil dieser Arbeit (siehe Kapitel 2) stützt sich stark auf die Definitionen der Mengenlehre und Logik und nimmt sich dabei einige Freiheiten bei der Notation, die folgend beschrieben werden

Definition 1.1.1 (Identifikator) *Ein Identifikator oder auch Bezeichner sei gegeben durch eine beliebige Folge an Symbolen, mit der keine Standardoperationen assoziiert werden. Identifikatoren werden zur Benennung und zum Referenzieren von Objekten verwendet.*

Mit Standardoperationen sind gänge Symbole wie die der arithmetischen Operationen gemeint $(+, -, \cdot, \div, =, \leq$, etc.)

Der Begriff des Identifikatoren wird in der Mathematik weniger häufig verwendet. Es handelt sich um einen Begriff der meist in der Informatik verwendet wird. Er unterscheidet sich vom Begriff der Variable durch die Wertzuweisung. Er kann zu verschiedenen Zeitpunkten verschiedene Objekte oder Werte darstellen.

Definition 1.1.2 (Wertzuweisung) *Einem Identifikator a kann über die Wertzuweisung $\leftarrow$ ein Wert bzw. eine Identität zugeordnet werden, $a \leftarrow 5$. Diese kann auch verändert werden. Dies setzt eine zeitliche Reihenfolge voraus. Über das Gleichheitszeichen $''=''$ können Aussagen zum Wert eines Identifikators getroffen werden $\neg(a = 6)$ (mit $\neg$ zur Negation der Aussage)*

Definition 1.1.3 (Mengen) *Eine Menge M stellt eine Zusammenfassung (Komprehension) an Objekten dar. Eine Menge hat keine Ordnung und es können keine Duplikate von Objekten enthalten sein. In der Definition einer Menge müssen die Objekte so definiert werden, dass diese eindeutig identifiziert werden können. Zwei Mengen sind genau dann gleich, wenn sie die gleichen Objekte beinhalten. Mengen können durch die Auflistung der enthaltenen Objekte definiert werden, dies wird durch geschweifte Klammern notiert*

$$\{o_1, o_2, o_3, ..., o_n\}$$

Wenn dazu wie folgt ein Identifikator angegeben wird $M = \{o_1, o_2, ..., o_n\}$, dann kann die Menge durch M referenziert werden.

Zwei Mengen gleichen sich genau dann, wenn sie genau dieselben Objekte beinhalten.

Bekannte standardisierte Mengen sind die verschiedenen Zahlenmengen $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$. Bei diesen Mengen stellen die Objekte Zahlen dar. Sie werden hier nicht weiter erläutert.

Definition 1.1.4 (Elementbeziehung) *Ein Objekt, dass sich in einer Menge befindet, wird als Element dieser Menge bezeichnet. Wenn ein Element mit dem Identifikator o in einer Menge M enthalten ist, kann dies wie folgt notiert werden $o \in M$ (gelesen als "o ist Element der Menge M"). Die Zugehörigkeit mehrerer Objekte o_1 und o_2 zu einer Menge kann wie folgt notiert werden $o_1, o_2 \in M$.*

Definition 1.1.5 (Kardinalität) *Die Kardinalität einer Menge M beschreibt die Anzahl der Elemente in dieser Menge. Sie wird angegeben mit $|M| \in \mathbb{N}$.*

Definition 1.1.6 (Teilmenge) *T ist genau dann eine Teilmenge von O, wenn T mindestens ein Objekt aus der Menge O beinhaltet und kein Objekt, das nicht auch in O enthalten ist. Die Beziehung wird notiert als $T \subseteq O$ bzw. $O \supseteq T$. O wird als Obermenge von T bezeichnet. Eine echte Teilmenge ist eine Teilmenge, die weniger Objekte beinhaltet als die Obermenge, also $|T| < |O|$ und wird mit $T \subset O$ bzw. $O \supset T$ angegeben.*

Anhand von Mengen kann die Abbildung als Relation zwischen Elementen zweier Mengen definiert werden.

Definition 1.1.7 (Abbildung) *Eine (unäre) Abbildung, auch Funktion genannt, ist eine Zuordnung von Elementen einer Menge auf Elemente einer anderen oder derselben Menge. Seien die zwei Mengen B und C gegeben. Eine Abbildung der Elemente von B auf Elemente von C wird wie folgt notiert*

$$B \to C$$

Dabei muss jedem Element von B genau ein Element von C zugeordnet werden. Einer Abbildung lässt sich wie folgt ein Identifikator zuweisen

$$A : B \to C$$

Die Menge, deren Elemente auf die andere abgebildet wird (in gegebenem Fall A) wird als Urbild, Quellmenge oder Definitionsbereich bezeichnet. Die Elemente dieser Menge werden als unabhängige Variablen oder im Kontext der Informatik auch als Parameter bezeichnet. Die Menge, auf die abgebildet wird, wird als Abbild, Ziel- oder Bildmenge oder als Wertebereich bezeichnet. Die Elemente dieser Menge werden als abhängige Variablen bezeichnet.

Wenn zur Definition der Abbildung Eigenschaften der unabhängigen und der abhängigen Variablen benötigt werden, wie bspw. die Kardinalität eines Elementes, werden diesen wie folgt Identifikatoren zugewiesen

$$A : x \mapsto y$$

oder

$$A : x \in B \mapsto y \in C$$

Der Wert einer Abbildung für ein Objekt $b \in B$ wird wie folgt gegeben $A(x)$. Eine Gleichung stellt eine logische Aussage zur Relation zweier Objekte $b \in B$ und $c \in C$ und der Abbildung dar und kann wie folgt notiert werden $A(b) = c$. Diese Aussage impliziert, dass das Element $b \in B$ durch die Abbildung A auf das Element $c \in C$ abgebildet wird.

Aus einer Abbildung $a : A \to B$ ist nicht immer eine Abbildung $B \to A$ ableitbar. Eine Abbildung wird als surjektiv bezeichnet, wenn jedes Element der Bildmenge mindestens einem Element des Definitionsbereichs zugeordnet wird. Sie wird als injektiv bezeichnet, wenn kein Wert der Bildmenge mehr als einem Element des Definitionsbereichs zugeordnet wird. Eine Abbildung ist bijektiv wenn sie sowohl sur- als auch injektiv ist. Nur wenn eine Abbildung $a : A \to B$ bijektiv ist, kann aus ihr eine Abbildung $B \to A$ abgeleitet werden.

Definition 1.1.8 (Logikoperatoren) *Seien A und B Aussagen, die entweder wahr oder falsch sein können, dann werden folgenden Operatoren als Logikoperatoren bezeichnet:*

Negation: $\neg A$
A ist nicht wahr

Konjunktion: $A \wedge B$
A und B sind wahr

Disjunktion: $A \vee B$
mindestens eine der beiden Aussagen A und B ist wahr

Kontravalenz: $A \oplus B$
entweder ist A oder B wahr

Implikation: $A \Rightarrow B$
wenn A wahr ist, dann ist B wahr

Äquivalenz: $A \Leftrightarrow B$
A und B haben denselben Wahrheitswert

Die Logikopertoren können verwendet werden, um logische Aussagen über Elemente auszudrücken.

Definition 1.1.9 (Quantoren) *Sei M eine Menge. Die folgenden Symbole werden verwendet, um Aussagen zu einer Menge zu machen, statt nur für einzelne Objekte in der Menge.*

Existenzquantor: $\exists m \in M$
Es existiert mindestens ein Element in M.

Eindeutigkeitsquantor: $\exists! m \in M$
Es existiert genau ein Element in M.

Allquantor: $\forall m \in M$
Jedes Element in M.

Quantoren werden Aussagen durch ":" zugeordnet. Dabei kann sich die Aussage auf den Identifikator beziehen. Sei A eine Aussage, dann kann diese Aussage mittels der Quantoren für alle Elemente $\forall m \in M : A(m)$, mindestens ein Element $\exists m \in M : A(m)$ und genau ein Element $\exists! m \in M : A(m)$ getroffen werden.

Mit Hilfe von Quantoren und Logikopertoren können Mengen auch als eine Sammlung an Eigenschaften definiert werden.

Definition 1.1.10 (Eigenschaften) *Mengen können neben der Auflistung von Objekten auch als eine Menge an Eigenschaften definiert werden, die als logische Aussagen dargestellt werden. Jedes Objekt, dass alle Eigenschaften erfüllt, ist in der Menge enthalten.*

Sei M eine Menge. Seien die Eigenschaften durch eine Aussage A gegeben (mehrere Aussagen können durch Konjunktion zu einer einzigen zusammengefasst werden). Dann kann die Menge M durch folgende Implikation über der Obermenge O definiert werden.

$$\forall x \in O : A(x) \Rightarrow x \in M$$

Solche Implikation können alternativ wie folgt notiert werden. Als erstes wird ein Identifikator vergeben. Dem Identifikator folgt ein vertikaler Strich und diesem die Auflistung der Eigenschaften, getrennt durch Kommata oder $\wedge$ (logisches "und").

$$M = \{x \in O | A(x)\}$$

In der Definition wurde nur eine Aussage verwendet, was daran liegt, dass sich eine Menge an Aussagen durch die Konjunktion zu einer einzigen Aussage vereinen lässt. Die folgende Implikation mit den zwei Aussagen A und B

$$\forall x \in O : (A(x) \wedge B(x)) \Rightarrow x \in M$$

lässt sich alternativ wie folgt als Mengendefinition angeben

$$M = \{x \in O | A(x) \wedge B(x)\}$$

Es hat sich etabliert, die Konjunktionen in der Definition von Mengen durch Kommata zu ersetzen, sodass alternativ

$$M = \{x \in O | A(x), B(x)\}$$

notiert werden kann. Wenn auch für die effiziente Berechnung nicht ganz so vorteilhaft, kann die Definition von Mengen mittels Eigenschaften auch ohne Angabe einer Obermenge erfolgen.

Hier zwei Beispiele für die Definition von Mengen.

Beispiel 1.1.1 *Folgende Menge beinhaltet bestimmte Elemente aus der Menge der natürlichen Zahlen* $\mathbb{N}$.

$$\{x \in \mathbb{N} | x > 5\}$$

Die Menge beinhaltet alle natürlichen Zahlen die größer sind als 5. *Die Größe der Zahlen lässt sich wie folgt auch nach oben begrenzen.*

$$M = \{x \in \mathbb{N} | (x > 5) \wedge (x < 12)\}$$

oder äquivalent

$$M = \{x \in \mathbb{N} | (x > 5), (x < 12)\}$$

Die Elementbeziehung lässt sich als logische Äquivalenz ausdrücken

$$(m \in M) \Leftrightarrow ((m \in \mathbb{N}) \wedge (5 < m < 12))$$

Die Klammersetzung dient zur Verdeutlichung. Gleichungen, Aussagen und Operatoren werden folgend nicht mehr in einzelne Klammern gesetzt.

In folgendem Beispiel wird verdeutlicht, dass Objekte auch komplexer sein können, als einfache Zahlenwerte. Die gegebenen Eigenschaften treffen Aussagen zur Kardinalität der Objekte.

Beispiel 1.1.2 *Die Menge M enthält alle Objekte der Obermenge O, die bestimmte Anforderungen zur Kardinalität erfüllen.*

$$M = \{o \in O | 3 \leq |o|, 5 \geq |o|\}$$

Die Objekte in M sind somit Mengen oder andere Objekte, die eine Kardinalität besitzen, die zwischen 3 *und* 5 *liegt.*

Definition 1.1.11 (Potenzmenge) *Sei die Menge M gegeben, dann ist die Potenzmenge $\mathcal{P}(M)$ definiert als*

$$\mathcal{P}(M) = \{m | m \subseteq M\}$$

Definition 1.1.12 *Das Symbol $\emptyset$ stellt die leere Menge dar, eine Menge ohne Objekte. Sie wird formal definiert als $\{\}$ durch Auflistung aller Objekte darin oder äquivalent über Eigenschaften $\emptyset = \{x | x \notin x\}$.*

Mit Hilfe der Logikoperatoren, der Definition von Mengen über deren Eigenschaften lassen sich die Mengenoperationen wie folgt definieren.

Definition 1.1.13 (Mengenoperationen) *Seien A und B zwei Mengen, dann wird M wie folgt definiert:*

Vereinigung: $M = A \cup B = \{e | e \in A \vee e \in B\}$
M beinhaltet alle Elemente, die in A oder B enthalten sind.

Schnittmenge: $M = A \cap B = \{e | e \in A \wedge e \in B\}$
M beinhaltet alle Elemente, die in A und B enthalten sind.

Differenz: $M = A \backslash B = \{e | e \in A \wedge e \notin B)\}$
M beinhaltet alle Elemente, die in A enthalten sind, aber nicht in B.

Symmetrische Differenz: $M = A \Delta B = \{e | e \in A \oplus e \in B\}$
M beinhaltet alle Elemente, die entweder in A oder in B enthalten sind.

Alle angegebenen Mengenoperationen sind symmetrisch, sodass $A \cup B = B \cup A$, davon ausgenommen ist lediglich die Differenz. Ähnlich wie bei arithmetischen Termen werden die Operationen von links nach rechts ausgewertet.

Folgende Darstellung (Quelle: https://de.wikipedia.org/wiki/Mengenlehre) gibt für jede Mengenoperation ein Beispiel im zweidimensionalen kontinuierlichen Raum, bei dem zwei Mengen A und B durch deren Grenzen in Form von Kreisen dargestellt werden. Von links nach rechts werden die Vereinigung, die Schnittmenge, die Differenz und die Symmetrische Differenz dargestellt.

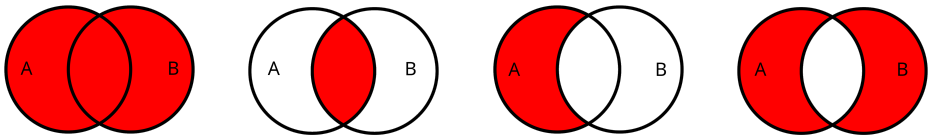

Abbildung 1.1: Mengenoperationen

Beispiel 1.1.3 *In folgender Menge stellen Paare aus Mengen zusammen jeweils ein Element dar, wenn diese Mengen die gegebenen Eigenschaften erfüllen*

$$\{\{x, y\} \in I | x \cap y \neq \emptyset \wedge x \Delta y \neq \emptyset\}$$

Die Elemente dieser Menge sind selbst Mengen, mit jeweils zwei Elementen, die eine gemeinsame echte Teilmenge aufweisen. Das heißt die Mengen jedes Elements gleichen sich in einem Teil und unterscheiden sich in einem anderen Teil. Die Obermenge I ist in diesem Beispiel nicht von Belang und wird nicht weiter spezifiziert. Es sei zu beachten, dass die Elemente in $\{x, y\}$ keine Reihenfolge aufweisen.

Definition 1.1.14 (Tupel) *Ein Tupel ist eine geordnete Zusammenfassung von Objekten, in welcher Duplikate möglich sind. Tupel werden durch runde Klammern angegeben $v = (o_1, o_2, ..., o_n)$.*

Runde Klammern werden nicht nur zur Definition von Tupeln verwendet, sondern auch zur Angabe von Auswertungsreihenfolgen und zur Angabe von Parametern für Abbildungen, sowie zur Bildung von Vektoren.

Definition 1.1.15 (Kartesisches Produkt) *Das kartesische Produkt ist eine Operation zwischen zwei oder mehr Mengen A und B, die definiert ist als*

$$A \times B := \{(a, b) | a \in A \wedge b \in B\}$$

Seien $n \in \mathbb{N}_{\geq 2}$ Mengen A_i gegeben, wobei i eine Variable mit $i \in [n]$ (siehe Kapitel 1.1.2) sei und Teil der Identifikatoren. Dann ist das kartesische Produkt dieser Mengen definiert als

$$A_1 \times A_2 \times ... \times A_n := \{(a_1, a_2, ..., a_n) | \forall i \in [n] : a_i \in A_i\}$$

Das kartesische Produkt einer Menge mit sich selbst wird notiert wie die Potenz $A \times A = A^2$. Somit gebe A^n, mit $n \in \mathbb{N}_{\geq 2}$ das n-fache kartesische Produkt mit sich selbst an. Es gelte $A^n = A \times A^{n-1}$, für alle $n > 2$.

Das kartesische Produkt kann bei der Definition von Vektoren und der n-ärer Abbildungen verwendet werden.

Definition 1.1.16 (Vektoren) *Ein Vektor v ist ein Tupel, das ausschließlich Zahlenwerte als Objekte beinhalten. Ein Vektor mit $n \in \mathbb{N}$ reellen Zahlen wird notiert als*

$$v \in \mathbb{R}^n$$

Für Vektoren sind bestimmte Rechenoperationen definiert. In dieser Arbeit wird nur die Addition von Vektoren verwendet, weshalb nur diese beschrieben wird. Die Addition von Vektoren vor, dass die Vektoren dieselbe Kardinalität aufweisen. Das i-te Element des ersten Operanden wird mit dem i-ten Element des zweiten Addiert.

Definition 1.1.17 (n-äre Abbildung) *Seien die Mengen X, Y und Z gegeben, sowie die Menge K als das kartesische Produkt $K = X \times Y$, dann kann eine unäre Abbildung A der Menge K auf die Menge Z notiert werden als*

$$A : K \to Z$$

oder äquivalent

$$A : X \times Y \to Z$$

Aussagen zu bestimmten Werten als Parameter der Abbildung werden als $A(x, y)$, mit $x \in X$ und $y \in Y$ notiert.

Definition 1.1.18 (Partition) *Sei eine Menge M gegeben, dann ist eine Partition P über M eine Einteilung in i Klassen $P = \{C_1, C_2, C_3, ..., C_i\}$, mit $1 \leq i \leq |M|$ und $\forall C_i \in P : C_i \neq \emptyset$, sowie $\forall (C_i, C_j) \in \{(c_i, c_j) \in P^2 | c_i \neq c_j\} : C_i \cap C_j = \emptyset$ und $\bigcup_{C_i \in P} C_i = M$.*

Beispiel 1.1.4 *Folgende Menge verdeutlicht die Anwendung von kartesischen Produkten in Kombination mit Tupeln*

$$\{(x, y) \in J^2 | x \cap y \neq \emptyset \wedge x \Delta y \neq \emptyset \wedge |x| < 5\}$$

In diesem Fall handelt es sich bei der Obermenge $J^2 = J \times J$ um ein kartesisches Produkt. Die runden Klammern geben an, dass es sich um ein geordnetes Objekt handelt. Die Anforderung $|x| < 5$ bezieht sich somit immer auf das erste Element des Tupels.

Inhaltsverzeichnis

1.1.2 Notationen

Die folgenden Notationen sind weniger geläufig, als die des vorhergehenden Kapitels. Da sich diese Arbeit stark an der Publikation [12] orientiert, werden deren Notationen übernommen.

Notation 1.1.1 *Sei ein Wert $a \in \mathbb{N}$ gegeben, dann sei $[a] = \{1, 2, 3, ..., a\}$.*

Notation 1.1.2 *Sei eine Menge A gegeben, dann sei die Potenzmenge von A gegeben mit $2^{[A]}$.*

Notation 1.1.3 *Sei eine Partitionierung P über einem Graphen gegeben, dann sei die Klasse, in der ein gegebener Knoten v eingeteilt wird, mit $P[v]$ gegeben.*

Notation 1.1.4 *Die Elemente eines Tupels werden wie folgt über eckige Klammern angegeben, $v[i]$, wobei v den Vektor und $i \in \mathbb{N}$ die Position angibt (beginnend bei Position 1).*

Notation 1.1.5 *Sei M eine Menge und $n \in \mathbb{N}_{>0}$, dann sei $\binom{M}{n} = \{m \in 2^{[M]} | n = |m|\}$.*

Inhaltsverzeichnis

1.1.3 Komplexitätstheorie

In diesem Kapitel werden die Grundlagen zur Diskussion der Komplexität des Fair Correlation Clusterings vermittelt. Im ersten Unterkapitel 1.1.3.1 werden Definitionen gegeben, die zum Verständnis essentiell sind, wie die eines Algorithmus, eines algorithmischen Problems und die von Rechnermodellen. Danach werden im nächsten Unterkapitel 1.1.3.2 die unterschiedlichen Zielsetzungen von Problemen beschrieben. Im darauf folgenden Unterkapitel 1.1.3.3 wird die Landau Notation und das Mastertheorem vorgestellt, die zur Herleitung der asymptotischen Grenzen für die Laufzeit von Algorithmen genutzt werden. Im letzten Unterkapitel 1.1.3.4 werden dann die verschiedenen Komplexitätsklassen und die Reduktion zum Nachweis der Zugehörigkeit beschrieben.

Die Komplexitätstheorie befasst sich mit dem Vergleich von Algorithmen. Algorithmen lassen sich nur sinnvoll vergleichen, wenn sie zur Lösung desselben Problems dienen. Algorithmen zur Lösung desselben Problems werden über ihren Ressourcenaufwand miteinander verglichen. In der vorliegenden Arbeit wird nur die Laufzeit als Ressource betrachtet. Der Speicherbedarf ist ebenfalls Gegenstand der Komplexitätstheorie, wird hier aber nicht behandelt. Die Laufzeit eines Algorithmus wird durch die Anzahl der verwendeten Operationen bestimmt. Die Operationen, die einem Algorithmus zur Verfügung stehen, müssen elementar sein. Sie werden in Rechnermodellen vorgegeben. Da derselbe Algorithmus zur Lösung verschiedener Eingaben unterschiedlich lang brauchen kann, wird mit Hilfe der Landau Notation eine asymptotische Grenzfunktion hergeleitet, welche die maximale Anzahl der Operationen in Abhängigkeit der Eingabegröße angibt.

Über die Komplexität des besten Algorithmus zur Lösung eines Problems kann die Komplexität des Problems selbst hergeleitet werden. Probleme werden aufgrund ihrer Komplexität in Komplexitätsklassen eingeteilt. Probleme einer Klasse können oftmals von Algorithmen für andere Probleme derselben Klasse gelöst werden. Diese Eigenschaft wird bei der Reduktion verwendet, um die Komplexität verschiedener Probleme zu beweisen.

Neben der Klassifizierung nach Effizienz, werden Probleme auch danach eingeteilt, welche Informationen in der Lösung gefordert sind, wie bspw. dem Entscheidungsproblem, bei dem ein Wert "ja" oder "nein" gefordert ist oder dem Optimierungsproblem, bei dem es einen Wert zu maximieren oder zu minimieren gilt. Aus Optimierungsproblemen, die nicht als effizient lösbar gelten, werden oftmals Approximationsprobleme hergeleitet. Approximationen sind Lösungen, die nicht unbedingt exakt sind, aber sich exakten Lösungen annähern. Die Approximationsproleme haben nur einen Sinn,

wenn sie effizient lösbar sind, daher werden sie nicht anhand ihrer Komplexität klassifiziert, sondern danach, wie nahe die Lösungen mindestens an eine exakte Lösung herankommen. Dies wird durch die Approximationsgüte ausgedrückt.

Diese Zusammenhänge werden in den folgenden Unterkapiteln im Detail betrachtet. Für eine tiefergehende Einführung sei auf das Bcuh von Wegener [42] und aus dem Buch von Ernst, Schmidt und Beneken auf die Kapitel [21] und [22] verwiesen, welche als Quellen dieses Kapitels dienen.

Inhaltsverzeichnis

1.1.3.1 Algorithmen und algorithmische Probleme

Definition 1.1.19 (Alphabet) *Eine endliche geordnete Menge wird als Alphabet bezeichnet.*

Definition 1.1.20 (Symbol) *Die Elemente eines Alphabets werden als Symbole bezeichnet.*

Definition 1.1.21 (Transitive Hülle) *Sei ein Alphabet Σ gegeben, dann wird das x-fache kartesische Produkt von Σ mit sich selbst als transitive Hülle Σ^+ bezeichnet, definiert als $\Sigma^+ = \Sigma^x$, mit $x \in \mathbb{N}_{>0}$.*

Definition 1.1.22 (Kleensche Hülle) *Die kleensche Hülle Σ^* eines Alphabets Σ ist definiert als*

$$\Sigma^* = \Sigma^+ \cup \{\epsilon\}$$

, wobei ϵ das leere Wort darstellt, das die Länge $|\epsilon| = 0$ aufweist.

In der kleensche Hülle sind alle Kombinationen enthalten, die sich aus den Symbolen des Alphabets bilden lassen zusammen mit dem leeren Wort.

Das leere Wort wird als Ausgangspunkt zur Erstellung anderer Wörter genutzt, mittels sogenannter Ableitungsregeln. Die Ableitungsregeln werden in Grammatiken definiert. Diese werden hier nicht vorgestellt, weil sie zum Verständnis nicht notwendig sind.

Definition 1.1.23 (Formale Sprache) *Sei Σ ein Alphabet, dann wird die Teilmenge L der kleenschen Hülle über diesem Alphabet $L \subseteq \Sigma^*$ als formale Sprache bezeichnet.*

Definition 1.1.24 (Wort) *Sei ein Alphabet Σ und eine formale Sprache $L \subseteq \Sigma^*$ gegeben, dann wird jedes Element $x \in L$ als Wort bezeichnet.*

Eine formale Sprache ist somit eine Menge an Symbolkombinationen, die als Wörter bezeichnet werden, und in der das leere Wort beinhaltet ist.

Beispiel 1.1.5 *Während die kleensche Hülle alle Symbolkombinationen bis zu einer bestimmten Länge beinhaltet, die mit gegebenen Symbolen möglich sind, beinhaltet eine Sprache ggf. nur eine echte Teilmenge davon. So ist "aefaeoij" eine Kombination an Symbolen aus dem deutschen Alphabet, die im Deutschen kein Wort darstellt.*

Definition 1.1.25 (Algorithmisches Problem) *Sei L eine formale Sprache über dem Alphabet Σ. Jedes Wort $x \in L$ werde als Eingabe bezeichnet und jeder Eingabe wird eine Menge zulässiger Ausgabewörter $w \in Y \subseteq \Gamma^*$ oder auch Lösungen zugeordnet, $S : L \to 2^{[Y]}$, wobei $\Sigma \subseteq \Gamma$ und $S(x) \neq \emptyset$. Das Tupel $(L, S : L \to 2^{[Y]})$ stellt ein algorithmisches Problem dar.*

Die Wörter $x \in L$ der Sprache L eines Problems $(L, S \to 2^{[Y]})$ werden als Probleminstanzen, Eingabewörter oder Eingaben bezeichnet. Die Wörter $y \in Y$ werden als Ausgabewörter, Ausgaben oder Lösungen bezeichnet.

Beispiel 1.1.6 *Das Übersetzen von Texten von einer Sprache in eine andere Sprache kann als algorithmisches Problem definiert werden. Ein Wort kann dabei ggf. in verschiedene Wörter korrekt übersetzt werden. Daher wird ein Eingabewort von der Abbildung $S : \Sigma \to 2^{[Y]}$ auf eine Menge von Ausgabewörtern abgebildet.*

Beispiel 1.1.7 *Mathematik ist eine Sprache. Das bedeutet, dass Eingabewörter auch durch mathematische Symbole dargestellt werden können. Die Symbole des Alphabets können (müssen aber nicht) auf Zahlen beschränkt sein. Damit kann eine Eingabe ein Vektor sein oder auch eine komplexere mathematische Struktur, bspw. ein Tupel, wie es bei Graphen der Fall ist.*

Beispiel 1.1.8 *Sei die Menge der Eingabewörter auf Vektoren beschränkt und die Menge der Ausgabewörter auf die beiden Wörter "wahr" und "falsch", dann ist ersichtlich, dass die Sprache L, die nur Vektoren beinhaltet, die Wörter von $\Gamma = \{''wahr'','' falsch''\}$ nicht beinhaltet.*

Definition 1.1.26 (Algorithmus) *Sei eine endliche Menge elementarer Operationen gegeben, sowie ein Problem $L, S : L \to 2^{[Y]}$, dann ist ein Algorithmus gegeben als eine endliche Sequenz an Operationen, welche eine Abbildung zur Berechnung eines Ausgabewortes zu gegebenem Eingabewort beschreibt, die in endlich vielen Schritten ausgeführt werden kann. Es handelt sich also um die Beschreibung Abbildung $L \to Y$. Dazu äquivalent ist die Abbildung $x \mapsto S(x)$, sodass ein Eingabe $x \in L$ auf eine Lösung $y \in S(x)$ abgebildet wird.*

Die endliche Menge elementarer Operationen, die zur Formulierung eines Algorithmus zur Verfügung gestellt werden, ist der Definition nach ein Alphabet. Dieses Alphabet wird vom sogenannten Rechnermodell definiert.

Definition 1.1.27 (Rechnermodell) *Ein Rechnermodell (auch abstrakte Maschine genannt) stellt eine Menge an elementaren Operationen dar und dazu eine unbeschränkte Anzahl an Speicherzellen. In der Menge der Operationen sind mindestens die arithmetischen Operationen, sowie Operationen zur Speicheradressierung und Wertzuweisung enthalten. Dazu ist eine Möglichkeit zur Repräsentation des Algorithmus gegeben, sowie ein Operationszähler, der die Position der nächsten Operation im Algorithmus angibt. Dazu muss mindestens eine Anfangsposition im Algorithmus und eine in den Speicherzellen gegeben sein.*

Während Computer Eingaben binär darstellen, wird in den meisten Rechnermodellen in jeder Speicherzelle genau ein Symbol gespeichert. In der Praxis ist eine Division sehr viel aufwendiger als bspw. eine Addition. Die meisten Rechnermodelle abstrahieren davon und betrachten alle elementaren Operationen als gleichwertig (uniform cost model), um die Komplexitätsanalyse zu vereinfachen. [42]
Es gibt verschiedene Rechnermodelle. Das wohl bekanntesten Rechnermodelle ist die Turingmaschine.

Definition 1.1.28 (Turingmaschine) *Eine Turingmaschine ist ein Tupel $(Q, q_0, \Sigma, \Gamma, \delta, Q')$. Dabei sei*

- *Q die Menge an "inneren Zuständen"*
- *q_0 die Startposition des Lesekopfes auf einem unendlich langen Band an Speicherzellen*
- *Σ ein endliches Eingabealphabet*
- *Γ ein endliches Arbeitsalphabet, deren Sympbole auf das Band geschrieben werden können, wobei $\Sigma \cup \epsilon \subseteq \Gamma$, dabei sei ϵ das leere Wort*

- $Q' \subseteq Q$ *die Menge an Haltezuständen, welche die Beendigung der Berechnung angibt*
- $\delta : \Gamma \times Q \to \Gamma \times Q \times \{-1, 0, +1\}$ *die Abbildung der Arbeitsschritte*

Zu Beginn der Ausführung stehen auf dem Band in den ersten $n = |x|$ Zellen die Eingabe $x \in L$. Alle anderen Zellen enthalten das Leerzeichen ϵ. Die Maschine weist den inneren Zutand q_0 auf und der Lesekopf befindet sich über der ersten Speicherzelle. In jedem Arbeisschritt kann die Turingmaschine das Symbol der Speicherzelle lesen, auf welchem der Lesekopf steht. Die Arbeitsvorschrift $\delta : \Gamma \times Q \to \Gamma \times Q \times \{-1, 0, +1\}$ gibt an, welches Symbol in die akuelle Zelle geschrieben wird, dazu wird der nächste Zustand angegeben und wie sich der Lesekopf relativ zu seiner aktuellen Position bewegen soll (eins nach links, eins nach recht, stehen bleiben). Sobald ein Haltezustand erreicht wird, stellen die ersten $m = |y|$ Symbole auf dem Band die $y \in \Gamma$ Ausgabe dar.

Im Kontext einer Turingmaschine kann ein Algorithmus als die Abbildung $\delta : \Gamma \times Q \to \Gamma \times Q \times \{-1, 0, +1\}$ zusammen mit den Zuständen q_0 und Q definiert werden. Das wird dadurch ersichtlich, dass alle Eingabewörter, die bei der Ausführung der Turingmaschine in einem Haltezustand enden, Wörter der akzeptierten Sprache sind. Alle anderen Wörter, sind nicht Teil der Sprache.

Der Sachverhalt, dass $\Sigma \subseteq \Gamma$ lässt sich dadurch erklären, dass der Text im Originalen als Eingabe $x \in L$, mit $L \subseteq \Sigma^*$ auf den ersten $n = |x|$ Speicherzellen gespeichert wird. Bei Ausführung des Algorithmus durch die Turingmaschine wird die Eingabe überschrieben. Wenn die Eingabe nur Teilweise überschrieben wird, werden Symbole aus Σ beibehalten und können dadurch in der Ausgabe vorhanden sein. Demnach können alle Symbole des Eingabealphabets in Ausgaben vorhanden sein und es folgt $\Sigma \subseteq \Gamma$.
Neben der Turinmaschine wird oft das Word-RAM Modell verwendet, welches, statt der schrittweisen Traversion über das Band, mit genau einem Schritt jede beliebige Speicherzelle adressieren kann, daher auch der Name Word-RAM (engl. random access machine). Dieser Unterschied verdeutlicht, dass die Laufzeit eines Algorithmus auch vom gewählten Rechnermodell abhängt.

Inhaltsverzeichnis

1.1.3.2 Problemvarianten

Es wird zwischen verschiedenen Arten von Problemen unterschieden. Die wohl natürlichste Problemvariante stellt das Suchproblem dar. Bei diesem wird für gegebene Eingabe genau ein Element aus der Lösungsmenge $s \in S(x)$ gesucht. Eine andere gängige Variante ist das Entscheidungsproblem, bei welchem für eine Eingabe x angegeben wird, ob diese eine zulässige Lösung hat bzw. ob das Wort x Teil der Sprache L ist. Das Entscheidungsproblem wird häufig zur Auswertung der Komplexität eines Problems verwendet, da in den meisten Fällen die Komplexität des Entscheidungsproblems mit der des Suchproblems übereinstimmt. Das liegt daran, dass das Suchproblem das Entscheidungsproblem löst und das Entscheidungsproblem oftmals durch die Lösung des Suchproblems gelöst wird.

Definition 1.1.29 (Suchproblem) *Sei ein algorithmisches Problem $(L, S : L \to \Gamma)$ gegeben. Die Lösung einer Eingabe $x \in L$ des Suchproblems stellt ein beliebiges Element $s \in S(x)$ dar.*

Definition 1.1.30 (Entscheidungsproblem) *Sei eine formale Sprache F über dem Alphabet Σ gegeben, mit $F \subseteq \Sigma^*$. Dann sei ein Entscheidungsproblem zu dieser Sprache gegeben als das Tupel $(\Sigma^*, S : L \to \{0, 1\})$, wobei die Abbildung definiert sei als*

$$S : x \mapsto \{y \in \{0, 1\} | (x \in F) \Rightarrow y = 1, (x \notin F) \Rightarrow y = 0\}$$

Eine weitere interessante Variante ist das Optimierungsproblem. Optimierungsprobleme sind eine häufig behandelte Gruppe an Problemen, nicht zuletzt, weil NP-schwierige Optimierungsprobleme durch Approximationen gelöst werden können, wenn in der Praxis keine exakte Lösung benötigt wird.

Definition 1.1.31 (Optimierungsproblem) *Sei ein Suchproblem $(L, S : L \to 2^{[Y]})$ gegeben, bei welchem jeder Eingabe $x \in L$ eine Menge zulässiger Lösungen $S(x)$ zugeordnet wird und jeder Lösung ein Wert durch eine Bewertungsfunkion $v : Y \to \mathbb{R}$. Optimierungsprobleme sind entweder Minimierungs- oder Maximierungsproblem, bei welchen nach einer zulässigen Lösung mit möglichst kleinem oder großem Wert gesucht wird. Ein Minimierunsproblem zu dem Suchproblem (L, S) ist gegeben durch das Tupel $(L, S_{min} : L \to 2^{[Y]})$, wobei*

$$S_{min}(x) = \{s \in S(x) | \forall s' \in S(x) : v(s) \leq v(s')\}$$

Analog wird ein Maximierungsproblem definiert als $(L, S_{max} : L \to 2^{[Y]})$, mit

$$S_{max}(x) = \{s \in S(x) | \forall s' \in S(x) : v(s) \geq s'\}$$

Definition 1.1.32 (Approximationsproblem) *Ein Approximationsproblem ist eine Optimierungsproblem, bei dem die Menge der zulässigen Lösungen $S(x)$ nicht die Teilmenge mit minimalen bzw. maximalem Wert $v(x, s)$ beschränkt werden, wie es bei exakten Lösungen von Optimierungsproblemen der Fall ist. Es handelt sich daher streng genommen um ein Suchproblem, bei dem aber zwischen Lösungen $s \in S(x)$ eine Ordnung besteht, die durch die Bewertungsfunktion gegeben wird, sodass bei Minimierungsproblemen Lösungen mit kleineren Werten bevorzugt werden.*

$$\forall s' \in S(x) : v(s) < v(s') \Rightarrow s \succ s'$$

Analoges gilt für Maximierungsprobleme.

Definition 1.1.33 (Approximationsgüte) *Sei ein Optimierungsproblem (L, S) gegeben, dann sei die Approximationsgüte einer Lösung $s \in S(x)$ in Abhängigkeit der Eingabe x gegeben als*

$$r(x, s) = \frac{\max_{s' \in S(x)}(v(s'))}{v(s)}$$

im Falle eines Maximerungsproblems bzw.

$$r(x, s) = \frac{v(s)}{\min_{s' \in S(x)}(v(s'))}$$

im Falle einer Minimierungsproblems.

Aus der Approximationsgüte $r(x, s)$ kann mit $\epsilon = r(x, s) - 1$ der Faktor hergeleitet werden, wie viel schlechter die Bewertung einer Approximation maximal sein darf als eine optimale Lösung. Im Kapitel 1.3.1 werden Approximationsgüten angegeben, mit $r(x, s) \leq 1$. Dabei handelt es sich um den Kehrwert der in Definition 1.1.33 beschriebenen Approximationsgüte für Maximierungsprobleme. Diese Schreibweise wird verwendet, weil dadurch die Unterscheidung zwischen Maximierungs- und Minimierungsproblemen einfacher ist.

Inhaltsverzeichnis

1.1.3.3 Komplexitätsanalyse

Nach der Definition der Algorithmen und Probleme kann nun die Analyse der Komplexität erläutert werden. Bei der Betrachtung der algorithmischen Komplexität werden - vereinfacht gesagt - die Anzahl der elementaren Operationen gezählt, die der Algorithmus für die Ausführung benötigt. Die Anzahl der Operationsschritte ist in den meisten Fällen abhängig von der Eingabegröße. Das bedeutet, dass zu jeder Eingabegröße die Anzahl der Operationen bestimmt werden muss. Die Eingabegröße wird grob als die Menge der Speicherzellen betrachtet, welche das Eingabewort x zur Repräsentation auf den Speicherzellen einnimmt, also $|x|$.

Es lassen sich Algorithmen definieren, die mit kleinen (oder großen) Eingaben sehr viel effizienter arbeiten als mit großen (bzw. kleinen). Dazu kommt, dass für viele Probleme Instanzen existieren, deren Lösung trivial ist. Demnach sind beim Vergleich der Laufzeit best-case Szenarios nicht besonders aussagekräftig. Die durchschnittliche Laufzeit (engl. average-case) ist meist nicht ermittelbar, da die Anzahl möglicher Probleminstanzen unendlich groß sein kann. Es bleibt daher nur der worst-case zum Vergleich. Dadurch genügt es, bei der Komplexitätsanalyse eine asymptotische Grenzfunktion zu schätzen. Diese Funktion gibt die maximale Anzahl an Operationsschritten in Abhängigkeit der Eingabegröße an.

Definition 1.1.34 (Landau Notation) *Die folgenden Mengen stellen die drei verschiedenen Typen von asymptotischen Schranken dar.*

$$o(f) = \{g \in \mathbb{R} \to \mathbb{R} | \forall (c > 0) : \exists n_0 : \forall (n \geq n_0) : g(n) < c \cdot f(n)\}$$

$$O(f) = \{g \in \mathbb{R} \to \mathbb{R} | \exists (c > 0) : \exists n_0 : \forall (n \geq n_0) : g(n) \leq c \cdot f(n)\}$$

$$\Omega(f) = \{g \in \mathbb{R} \to \mathbb{R} | \exists (c > 0) : \exists n_0 : \forall (n \geq n_0) : g(n) \geq c \cdot f(n)\}$$

Die asymptotischen Schranke $O(f)$ gibt eine Menge an Funktionen an, welche ab einem bestimmten Wert für c und n den Wert der Funktion f überschreitet oder diesem gleicht (daher auch asymptotisch/annähernd). $O(f)$ stellt eine Menge an Abbildungen dar, es hat sich jedoch etabliert $g = O(f)$ statt $g \in O(f)$ zu schreiben. Im Gegensatz zur gängigen Verwendung ist das Gleichheitszeichen in der Landau-Notation nicht symmetrisch, so gilt $O(n) = O(n^2)$, aber nicht $O(n^2) = O(n)$. Die Definition dieser Menge erlaubt es bei der Abschätzung der Grenzen nicht nur kleine Eingabeinstanzen zu vernachlässigen, sondern auch Faktoren. In dieser Arbeit wird nur die obere Schranke $O(f)$ verwendet. $o(f)$ stellt eine "harte" obere Grenze dar, die auch mit steigender Eingabegröße niemals überschritten wird. $\Omega(f)$ stellt eine "weiche" untere Grenze dar.

Die asymptotische Schranke eines Algorithmus wird nach folgendem Schema hergeleitet. Einzelnen Operationen wird eine "konstante asymptotische Schranke" von $O(1)$ zugeordnet. Das gleiche gilt für eine Folge von Operationen, mit konstanter Länge, weil laut Definition 1.1.34 konstante Faktoren vernachlässigt werden können. Das bedeutet, wenn mehrere Operationen auf eine Eingabe ausgeführt werden, und die Anzahl der Operationen unabhängig von der Größe der Eingabe ist, werden sie wie eine einzige behandelt und mit $O(1)$ gezählt. Erst wenn die Anzahl der Operationen abhängig von der Eingabegröße ist fallen sie stärker ins Gewicht.

Sequenzen deren Länge ein lineare Abhängigkeit von der Eingabegröße aufweisen liegen in $O(n)$, wobei sich n als die Angabe der Eingabegröße etabliert hat. Dies ist bei einfachen Schleifen der Fall, bei denen eine Menge an Operationen für jedes Element (Symbol) der Eingabe ausgeführt wird, sodass die Laufzeit von der Eingabegröße abhängt und mit dieser linear wächst. Geschachtelte Schleifen liegen je nach Tiefe in $O(n^k)$, wobei k die Verschachtelungstiefe angibt. Drei ineinander liegende Schleifen haben somit eine asymptotische Schranke von $O(n^3)$. Zwei ineinander geschachtelte Schleifen gehen für jedes Element in der Eingabe nochmal jedes Element der Eingabe durch.

Logarithmische Schranken kommen meistens zum Tragen, wenn die Datenmenge in einer Schleife, mit einem konstanten Faktor $c < 1$ multipliziert wird. Das ist bspw. bei der Halbierung der Fall, welche zum Beispiel bei der Suche in bereits sortierten Listen vorkommt.

Folgende Regeln gelten für die Landau Notation: [38]

- $g(n) = c \cdot f(n) + d = O(f)$, mit $c, d \in \mathbb{R}$
- $g(n) = a_k \cdot n^k + a_{k-1} \cdot n^{k-1} + ... + a_1 \cdot n = O(n^k)$, mit $\forall i \in [k] : a_i > 0$
- $n^i = o(n^j)$, mit $i, j \in \mathbb{R}$ und $j > i$
- $n^d = o(r^n)$, mit $d, r \in \mathbb{R}$ und $r > 1$, $d > 0$
- $g(n) + f(n) = O(g(n) + f(n))$ und $g(n) \cdot f(n) = O(g(n) \cdot f(n))$
- $g(n) = O(f)$, $f(n) = O(h)$ führt $g(n) = O(h)$ nach sich

Beispiel 1.1.9 *Sei eine Funktion $f(n) = c \cdot n^x$ und eine Funktion $g = n^y$ gegeben, mit $c > 1$ und $x < y$, dann gilt $f = O(g)$ und das unabhängig davon wie groß der Wert c ist.*

Wie im Beispiel erkennbar, gibt es eine Hierarchie zwischen den Operationen. Folgende Tabelle aus [21] verdeutlicht diese Hierarchie. Alle Schranken oben in der Tabelle werden von allen Schranken weiter unten beinhaltet. [38]

konstant	$O(1)$
logarithmisch	$O(log(n))$
polylogarithmisch	$O(log^k(n))$
linear	$O(n)$
linearithmisch	$O(n \cdot log(n))$
polynomiell	$O(n^k)$
exponentiell	$O(k^n)$
faktoriell	$O(n!)$

Tabelle 1.1: Komplexitätshierarchie

, mit $k \geq 1$.

Folgende Tabelle aus [21] verdeutlicht das unterschiedliche Wachstum der verschiedenen asymptotischen Laufzeiten. In der linken Spalte werden beispielhafte Größen der Eingaben angegeben, in den Spalten rechts davon werden Laufzeiten für verschiedene asymptotische Schranken angegeben.

n	$log(n)$	n	$n \cdot log(n)$	n^2	2^n
10	1μs	10μs	10μs	100μs	$\approx$ 1ms
100	2μs	100μs	200μs	10ms	$\approx 4 \cdot 10^{16}$Jahre
1000	3μs	1ms	3ms	1s	$\approx 8 \cdot 10^{288}$Jahre
10000	4μs	10ms	40ms	100s	∞
100000	5μs	100ms	500ms	167min	∞

Tabelle 1.2: Beispiele für asymptotisches Wachstum

Die Komplexität eines Problems ergibt sich aus der Komplexität des schnellsten Algorithmus der dieses Problem löst. Die algorithmische Komplexität eines Problems stellt ein Supremum dar, also eine untere Obergrenze. Mit der Entwicklung schnellerer Algorithmen kann diese Grenze nach unten korrigiert werden.

Inhaltsverzeichnis

1.1.3.4 Komplexitätsklassen

Probleme können entsprechend ihrer Komplexität in Komplexitätsklassen eingeteilt werden. Es existiert eine Vielzahl an Komplexitätsklassen, einige davon für die Approximationsprobleme. Für das Verständnis dieser Arbeit werden hauptsächlich Kenntnisse über die zwei Klassen P und NP benötigt und das vor Allem für das Kapitel 1.3.

Definition 1.1.35 (Komplexitätsklasse P) *Sei B ein Entscheidungsproblem, dann ist dieses nur Element der Klasse P (polynomial time), wenn eine Lösung von einer deterministischen Turingmaschine in polynomieller Zeit berechnet werden kann.*

Definition 1.1.36 (nicht-deterministische Turingmaschine) *Eine nicht-deterministische Turingmaschine ist eine Turingmaschine, bei welcher die Zustandsübergang durch eine Abbildung in Form von $\delta : \Gamma \times Q \times \{0,1\} \to \Gamma \times Q \times \{-1,0,+1\}$ ersetzt wird. Dieser Abbildung wird ein "Zufallsbit" (ein zufälliger Wert $b \in \{0,1\}$) übergeben, sodass in jedem Arbeitsschritt zufällig eine von zwei möglichen Abbildungen ausgewählt wird.*

Definition 1.1.37 (Komplexitätsklasse NP) *Sei L die Sprache eines Entscheidungsproblems. Dann ist das Entscheidungsproblem genau dann Element der Klasse NP (nondetermistic polynomial time), wenn eine nicht-deterministische Turingmaschine eine Eingabewort $x \in L$ auf mindestens einem der $2^{q(n)}$ möglichen Rechenwege in polynomieller Zeit $q(n)$ lösen kann und jede falsche Eingabe $x \notin L$ abgelehnt wird.*

Äquivalent dazu ist das Entscheidungsproblem genau dann in NP, wenn für eine gegebene Eingabe $x \in L$ und eine vorgegebenen Lösung y von einer deterministischen Turingmaschine in polynomieller Zeit als Lösung der Eingabe validiert werden kann, also bestimmt werden kann ob $y \in S(x)$ oder $y \notin S(x)$.

Die Klasse P beinhaltet alle Probleme, die mit polynomieller Laufzeit gelöst werden können. Sie werden als effizient lösbar betrachtet. Die Klasse NP ist Obermenge von P. Sie beinhaltet aber auch Probleme, die nicht in polynomieller Zeit lösbar sind. Es wird vermutet, dass $P \subset NP$. Für den Beweis für oder gegen das Theorem $NP \neq P$ steht eine Belohnung vom Clay Mathematical Institute aus.
Für die Herleitung der Zugehörigkeit zur Klasse P genügt es, einen Algorithmus mit polynomielle Laufzeit anzugeben. Für die Zugehörigkeit eines Problems zur Komplexitätsklasse NP wird eine Reduktion auf ein Problem erfordert, dessen Zugehörigkeit bereits bekannt ist. Eine Reduktion beschreibt den Nachweis, dass ein Problem mindestens so schwierig ist wie ein anderes Problem.

Definition 1.1.38 (Turingreduktion) *Sei ein Problem B_1 und ein Problem B_2 gegeben. Dann ist das Problem B_1 auf B_2 (Turing-)reduzierbar, wenn jede Probleminstanz von B_1 in eine Probleminstanz von B_2 umgewandelt werden kann, sodass alle Probleminstanzen von B_1 von einem Algorithmus zur Lösung von B_2 gelöst werden können und die Lösung in eine Lösung für B_1 überführt werden kann. Es wird dann $B_1 \leq_T B_2$ notiert.*

Definition 1.1.39 (Polynomielle Turingreduktion) *Ein Problem B_1 wird als polynomiell reduzierbar auf ein Problem B_2 bezeichnet, wenn die Umwandlung jedes Ein- oder Ausgabewortes in polynomieller Zeit möglich ist. Es wird dann $B_1 \leq_p B_2$ notiert.*

Für die Reduktionen gilt Transitivität. Seien drei Probleme A, B, C gegeben dann folgt aus $B \leq_p C$ und $A \leq_p B$, dass $A \leq_p C$. Folgende relativen Schwierigkeiten werden unterschieden.

Definition 1.1.40 (C-Einfach) *Sei $\mathcal{C}$ eine Komplexitätsklasse und A ein algorithmisches Problem. Dann wird das Problem A als $\mathcal{C}$-einfach bezüglich der polynomiellen Reduktion bezeichnet, wenn $\forall C \in \mathcal{C} : A \leq_p C$.*

Definition 1.1.41 (C-Schwierig) *Sei $\mathcal{C}$ eine Komplexitätsklasse und A ein algorithmisches Problem. Dann wird das Problem A als $\mathcal{C}$-schwierig bezüglich der polynomiellen Reduktion bezeichnet, wenn $\forall C \in \mathcal{C} : C \leq_p A$.*

Definition 1.1.42 (C-Äquivalent) *Sei $\mathcal{C}$ eine Komplexitätsklasse und A ein algorithmisches Problem. Dann wird das Problem A als $\mathcal{C}$-äquivalent bezüglich der polynomiellen Reduktion bezeichnet, wenn $\forall C \in \mathcal{C} : A \leq_p C \wedge C \leq_p A$. Es wird $A =_p C$ notiert.*

Definition 1.1.43 (C-Vollständigkeit) *Sei $\mathcal{C}$ eine Komplexitätsklasse und A ein algorithmisches Problem. Dann wird das Problem A als $\mathcal{C}$-vollständg bezüglich der polynomiellen Reduktion bezeichnet, wenn A als $\mathcal{C}$-schwierig gilt und $A \in \mathcal{C}$.*

Die Komplexitätsklasse P nimmt eine besondere Rolle in der Komplexitätstheorie ein. Probleme dieser Klasse werden als effizient lösbar bezeichnet. Dies beruht darauf, dass sich alle Rechnermodelle mit maximal polynomiellem Mehraufwand gegenseitig simulieren können. Das zieht nach sich, dass die Wahl des Rechnermodells einen Einfluss auf die Komplexität hat. Algorithmen können somit durch die Anpassung an Rechnermodelle verbessert werden, was die Aussagekraft von Vergleichen zwischen Algorithmen innerhalb von P schmälert.

Die Betrachtung von Approximationsproblemen und der damit einhergehende Verzicht auf eine exakte Lösung wird meistens dann in Kauf genommen, wenn ein Optimierungsproblem nicht effizient lösbar ist. In diesem Fall liegt die Hoffnung darin, durch den Kompromiss zu einer polynomiellen Laufzeit zu gelangen. Aus diesem Grund werden Approximationsalgorithmen nicht nach ihren Laufzeiten, sondern entsprechend ihrer Approximationsgüte klassifiziert.

Definition 1.1.44 (Komplexitätsklasse APX) *Sei A ein Approximationsalgorithmus, x eine Eingabe und $s_A(x)$ die Lösung von A, mit der Eingabe x, dann sei $r_A(x) = r(x, s_A(x))$ (siehe Definition 1.1.33). Sei dazu die Abbildung $r_A : \mathbb{N} \to \mathbb{R}_{\geq 1}$ gegeben, welche dem Algorithmus A eine maximale Approximationsgüte in Abhängigkeit der Eingabegröße $n \in \mathbb{N}$ zuordnet, mit $r_A(n) = sup\{r_A(x) | n \geq |x|\}$. Ein Approximationsproblem B liegt in der Komplexitätsklasse $APX(c)$, mit $c \geq 1$, wenn es einen Algorithmus A zur (approximativen) Lösung von B gibt, mit polynomieller Laufzeit und einer maximalen Approximationsgüte von $r_A(n) \leq c\}$.*

Unter einem c-Algorithmus wird meist ein Algorithmus der Klasse $APX(c)$ verstanden, also einem Algorihtmus A mit einer Approximationsgüte $r_A(x) = c$.

Definition 1.1.45 (Polynomielles Approximationsschema) *Ein polynomielles Approximationsschema (engl. polynomialtime approximation schema, PTAS) bezeichnet Approximationsalgorithmen, denen neben der Probleminstanz x ein Parameter $\epsilon \in \mathbb{Q}$ übergeben wird, der die Approximationsgüte vorgibt, wobei der Algorithmus eine polynomielle Laufzeit in Abhängigkeit von $|x|$ und ϵ aufweisen muss. Dabei gelte $\epsilon > 0$ und $r_A(x) = 1 + \epsilon$.*

Die Approximationsklassen werden nur für das Kapitel 1.3 benötigt. Der Hauptteil dieser Arbeit beschäftigt sich mit der exakten Lösung des Fair Correlation Clusterings.

Inhaltsverzeichnis

1.1.4 Graphentheorie

In der Graphentheorie werden Relationen verwendet, um möglichst effiziente Algorithmen für Probleme der linearen Programmierung (auch als lineare Optimierung bekannt) zu erstellen.

Definition 1.1.46 (Einfacher Graph) *Ein 2-Tupel $G = (V, E)$ wird als Graph bezeichnet, wenn V eine Menge an Objekten darstellt und E eine Menge an Relationen zwischen jeweils zwei Objekten aus V. Damit sei $E \subseteq V^2$. Wenn zusätzlich gilt, dass $e = (v_1, v_2) = (v_2, v_1) \in E$, handelt es sich um einen einfachen Graphen. Die Objekte V werden als Knoten bezeichnet und die Relationen in E als Kanten. welche eine Teilmenge des kartesischen Produktes der Knotenmenge mit sich selbst darstellt, also $E \subseteq V^2$. Für jedes Knotenpaar darf in einem einfachen Graphen nur eine Kante (2-Tupel) in E enthalten sein. Daraus folgt $|E| \leq |V^2|$, oder mit Landau-Notation $|E| = O(|V^2|)$ (siehe Kapitel 1.1.3).*

Die Knoten- oder Kantenmenge eines Graphen $G_1 = (A, B)$ sei neben den genauen Bezeichnern auch durch die Notation $V(G_1)$ oder $E(G_1)$ referenzierbar.

Folgend ist ein Beispielgraph auf verschiedene Arten dargestellt. Die Objekte der Knotenmenge werden als Kreise dargestellt und die Kanten als Linien, die jeweils zwei Knoten verbinden. In zwei der drei Darstellungen sind die Kanten so dargestellt, dass sie sich gegenseitig überschneiden. Es ist erkennbar, dass nicht jeder Knoten von einer oder mehr Kanten referenziert werden muss. Die übersichtliche Darstellung von Graphen ist ein eigenes Forschungsgebiet innerhalb der Graphentheorie. Die hier dargestellten Graphen sind planar (plättbar). Das bedeutet, dass sie sich im zweidimensionalen Raum ohne Überschneidung der Kanten darstellen lassen. Es gibt Graphen, bei denen dies nicht möglich ist. Diese Arbeit widmet sich Wäldern und Kaktusgraphen, diese sind immer planar.

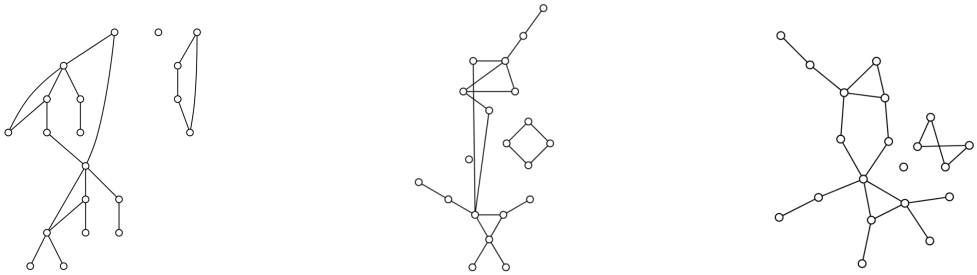

Abbildung 1.2: Beispielgraph in verschiedenen Darstellungen

Definition 1.1.47 (Adjazenz) *Sei ein Graph $G = (V, E)$ gegeben, dann sind zwei Knoten $u, v \in V$ adjazent oder auch benachbart zueinander, wenn $\exists e \in E : u, v \in e$.*

Definition 1.1.48 (Inzidenz) *Sei ein Graph $G = (V, E)$ gegeben, dann sei eine Kante $e \in E$ zu einem Knoten $v \in V$ inzident, wenn gelte $v \in e$.*

Knoten, die zu einer Kante inzident sind, werden als Endknoten dieser Kante bezeichnet.

Notation 1.1.6 (Nachbarschaft) *Sei ein Graph $G = (V, E)$ gegeben, dann sei mit der Abbildung $n : V \to 2^{[E]}$ die Nachbarschaft eines Knoten gegeben, also alle zu gegebenem Knoten adjazenten Knoten.*

Folgende Darstellung zeigt drei grüne Knoten und deren Nachbarschaft in Form von roten Knoten.

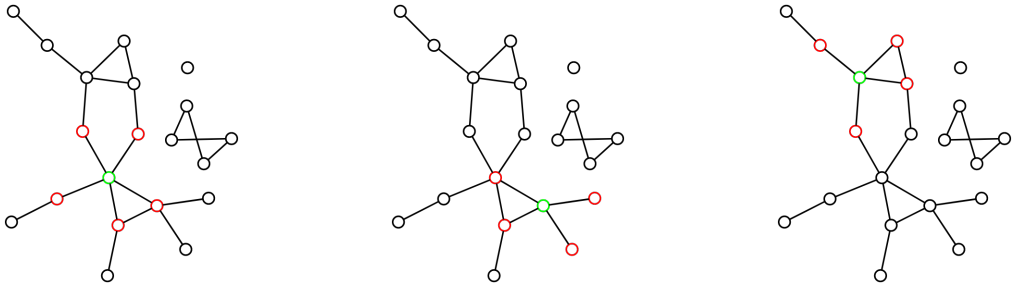

Abbildung 1.3: Nachbarschaft

Definition 1.1.49 (Grad) *Der Grad (engl. degree) eines Knoten ist durch eine Abbildung $deg : V \to \mathbb{N}$ gegeben, welche jedem Knoten die Anzahl der Kanten zuordnet, die inzident zu diesem Knoten sind.*

Definition 1.1.50 (Teilgraph) *Für einen beliebigen Graphen $G = (V, E)$ stellt $G' = (V', E')$ nur dann einen Teilgraph von G dar, wenn gilt, dass $V' \subseteq V$ und $E' \subseteq E$.*

Definition 1.1.51 (Pfad) *Sei ein Pfad f definiert als eine geordnete Kantenmenge $f = (e_1, e_2, ..., e_m)$, mit $m \geq 2$, wobei*

$$\forall e_i \in f : e_i = (v_i, v_{i+1})$$

und

$$\forall (i, j) \in \{(i, j) \in \mathbb{N}^2 | 1 \leq i < j \leq m \wedge i \neq j\} : v_i \neq v_j$$

Folgende Darstellung zeigt verschiedene Beispielpfade mit rot gefärbten Kanten und Knoten.

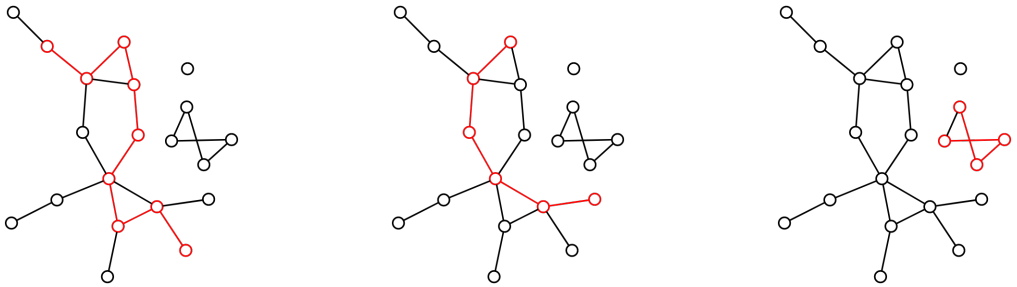

Abbildung 1.4: Pfade

Die Distanz zwischen zwei Knoten sei gegeben als die Anzahl an Kanten im kürzesten Pfad zwischen den beiden Knoten. Wenn die Knoten nicht derselben Komponente angehören ist die Distanz mit ∞ gegeben.

Definition 1.1.52 (Kreis) *Ein Kreis k sei definiert über die geordnete Kantenmenge $E(k) = (e_1, e_2, ..., e_m)$, mit $m \geq 2$, wobei $\forall e_i \in E(k) : e_i = (v_i, v_{(i+1)\%m})$ und*

$$\forall (i,j) \in \{(i,j) \in \mathbb{N}^2 | 1 \leq i < j \leq m\} : v_i \neq v_j$$

, sowie die geordnete Knotenmenge $V(k) = (v_1, v_2, ..., v_m)$, wobei

$$\forall v_i \in V(k) : \exists e_i \in E(k) : e_i = (v_i, v_{i+1})$$

Folgende Darstellung zeigt verschiedene Kreise mit rot gefärbten Kanten und Knoten.

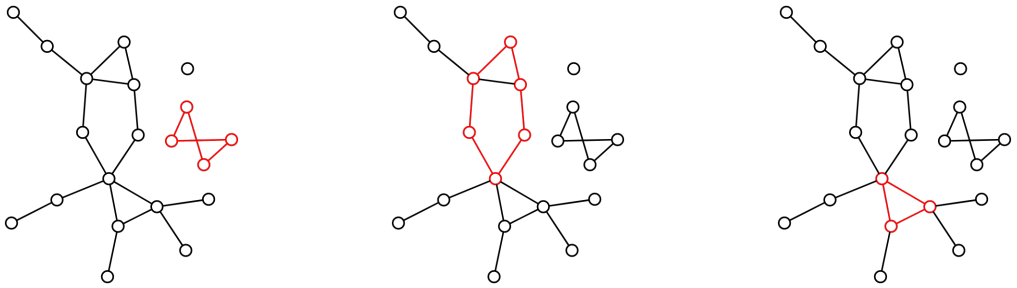

Abbildung 1.5: Kreise

Definition 1.1.53 (Zusammenhangskomponente) *Sei die Abbildung $F : V \times V \to 2^{[E]}$ gegeben, die jedem Knotenpaar die Menge aller Pfade zuordnet, die zwischen ihnen bestehen, sodass zwei Knoten $v_1, v_2 \in V$ nur dann in derselben Zusammenhangskomponente enthalten sind, wenn $F(v_1, v_2) \neq \emptyset$.*

Im Beispielgraph (Darstellung 1.2) sind drei Zusammenhangskomponenten zu sehen, eine bestehend aus einem einzelnen Knoten, eine bestehend aus vier Knoten und eine bestehend aus den restlichen Knoten.

Definition 1.1.54 (Induzierter Teilgraph) *Sei ein Graph $G = (V, E)$ gegeben, sowie eine Teilmenge $V' \subseteq V$, dann sei der Graph*

$$G[V'] = \left\{ V', E \cap \binom{V'}{2} \right\}$$

der durch V' induzierte Graph über G.

Folgende Darstellung zeigt verschiedene induzierte Graphen. Die Knotenmenge V' wird explizit angegeben und alle Kanten, von denen beide Endknoten in V' enthalten sind, sind auch im induzierten Graph enthalten. Die enthaltenen Knoten und Kanten werden mit durchgezogenen Linien dargestellt. Alle im induzierten Graphen nicht enthaltenen Knoten und Kanten werden mit unterbrochenen Linien dargestellt.

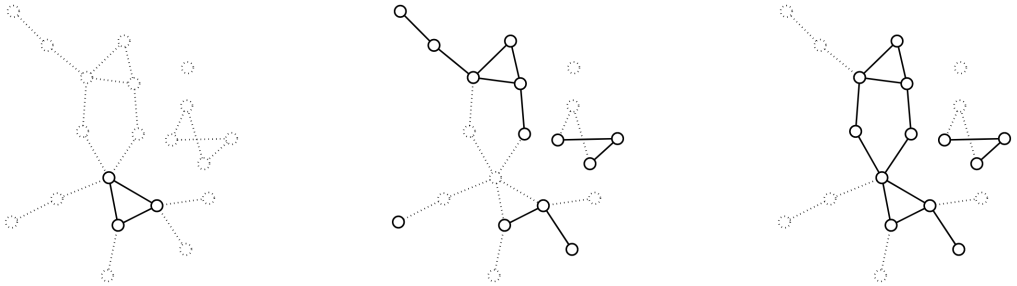

Abbildung 1.6: Induzierte Graphen

Definition 1.1.55 (Baum) *Ein Graph $G = (V, E)$ ist nur dann ein Baum, wenn er nur eine Zusammenhangskomponente und keine Kreise beinhaltet.*

Definition 1.1.56 (Wald) *Sei ein Graph $G = (V, E)$ gegeben. G ist nur dann ein Wald, wenn jede Zusammenhangskomponente einen Baum darstellt.*

Definition 1.1.57 (Kaktusgraph) *Sei ein zusammenhängender Graph $K = (V, E)$ gegeben. K ist nur dann ein Kaktus, wenn alle Kreise paarweise höchstens einen Knoten gemeinsam haben.*

Definition 1.1.58 (Gefärbte Elemente) *Eine Menge M wird als gefärbt bezeichnet, wenn es $k \in \mathbb{N}_{>0}$ gibt, die als Anzahl der Farben bezeichnet wird, zusammen mit einer Abbildung $c : M \to [k]$, die als Färbung bezeichnet wird, für die gilt, dass $\forall i \in [k] : \exists m \in M : c(m) = i$.*

Die Färbung der Elemente $m \in M$ impliziert eine Partition über der Menge M, mit $P = \{C_i \subseteq M | \forall m \in C_i : c(m) = i\}$. Die Anzahl der verschiedenen Farben gleicht dann der Kardinalität der Partition P, also $k = |P|$.

Definition 1.1.59 (Mengenfärbung) *Sei S eine Menge gefärbter Elemente, wobei die Färbung der Elemente mit $c : S \to [k]$ gegeben sei. Die Färbung der Menge S sei gegeben durch den Vektor $C(S) \in \mathbb{N}^k_{>0}$, wobei jedes Element des Vektors die Summe der Elemente dieser Farbe in S angibt, also gilt*

$$C(S)[i] = |\{s \in S | c(s) = i\}|$$

Inhaltsverzeichnis

1.1.5 Dynamische Programme für Bäume

Die im Hauptteil dieser Arbeit vorgestellten Algorithmen basieren auf dynamischen Programmen für Bäume. Zum Verständnis wird in diesem Abschnitt daher das dynamische Programm zur Lösung des Problem der minimalen Knotenüberdeckung für Bäume vorgestellt. Ein dynamisches Programm kann als eine Form von Rekursion betrachtet werden, bei welchem Zwischenergebnisse gespeichert verwendet werden, um die Laufzeit auf Kosten des Speicherbedarfs zu verringern.

Definition 1.1.60 (Knotenüberdeckungsproblem) *Sei ein Graph $G = (V, E)$ gegeben. Eine Knotenüberdeckung sei definiert als eine Menge $V' \subseteq V$, für die gilt, dass von jeder Kante in E mindestens ein Endknoten enthalten ist, also*

$$\forall e = (u, v) \in E : u \in V' \vee v \in V'$$

oder äquivalent

$$\forall e \in E : e \cap V' \neq \emptyset$$

Definition 1.1.61 (Minimale Knotenüberdeckung) *Sei ein Graph $G = (V, E)$ gegeben, zusammen mit einer Abbildung $w : V \to \mathbb{R}$. Eine minimale Knotenüberdeckung ist eine Knotenüberdeckung, in welcher es das Gewicht*

$$w(V') = \Sigma_{v \in V'} (w(v))$$

zu minimieren gilt.

Das Problem der Minimalen Knotenüberdeckung ist NP-vollständig für generelle Graphen (für den Beweis siehe [38]).

Lemma 1.1.1 *Für Eingabegraphen in Form von Bäumen ist das Problem über dynamische Programmierung in polynomieller Zeit $O(|E|)$ lösbar.*

Beim dynamischen Programm zur Lösung des Problems wird eine Spezialisierung des Problems auf allen Teilbäumen gelöst. Dazu wird der Baum in Teilbäume aufgeteilt, die folgend definiert werden.

Definition 1.1.62 (Teilbaum T_v) *Sei ein Baum $T = (V, E)$ gegeben, mit einem Knoten $r \in V$ als Wurzel von T. Sei die geordnete Menge der Kinder von $v \in V$ gegeben durch*

$$U(v) = \{u \in V | \exists e \in E : u, v \in e \wedge d(u) > d(v)\})$$

, wobei die Abbildung $d : V \to \mathbb{N}$ die Distanz eines Knoten zur Wurzel r angibt. Dann sei für jeden Knoten $v \in V$, mit $v \neq r$ ein Teilbaum

$$T_v = T\left[\{v\} \cup \left(\bigcup_{u \in U(v)} (T_u)\right)\right]$$

definiert, wobei $T[S \subseteq V]$ den induzierten Graphen von S angebe.

Geordnete Mengen werden hier weiterhin mit geschweiften Klammern definiert, weil die Angabe von Eigenschaften in Tupeln nicht etabliert ist.
Um nicht ständig die formale Definition angeben zu müssen, wird in dieser Arbeit teilweise vom Teilbaum "unter" gegebenem Knoten gesprochen, sodass mit dem Teilbaum unter v der Teilbaum T_v gemeint ist.

Für den Beweis des Lemmas 1.1.1 werden folgend zwei Spezialisierungen des Knotenüberdeckungsproblems gegeben.

Definition 1.1.63 (Spezialisierung Knotenüberdeckungsproblem) *Bei der ersten Spezialisierung wird als Parameter ein Knoten übergeben, der in der Knotenmenge der Knotenüberdeckung enthalten sein muss. Es sei mit $\tau : v \in V \mapsto \mathbb{R}$ das Gewicht gegeben, das eine minimale Knotenüberdeckung über dem Teilbaum T_v gegeben, das den angegebenen Knoten in der Knotenüberdeckung beinhaltet.*

Die zweite Spezialisierung verlangt nach einer Knotenüberdeckung, in welcher der angegebene Knoten nicht enthalten ist. Mit $\tau' : v \in V \mapsto \mathbb{R}$ sei das Gewicht einer Knotenüberdeckung über dem Teilbaum T_v gegeben, in der der Knoten v nicht enthalten ist.

Mit Hilfe dieser Spezialisierungen kann nun der Beweis zu Lemma 1.1.1 geführt werden.

Beweis: [Lemma 1.1.1] Sei der Baum $G = (V, E)$ gegeben, zusammen mit dem Wurzelknoten $r \in V$. Für jeden Knoten $v \in V$ wird das Gewicht $\tau(v)$ und $\tau'(v)$ bestimmt, sodass für jeden Teilbaum T_v das Gewicht mit bzw. ohne die Wurzel in der Knotenüberdeckung gegeben ist.

Wenn ein Teilbaum T_v betrachtet wird, gibt es nur zwei Möglichkeiten: entweder wird der Knoten v in der Lösungsmenge V' beinhaltet, oder nicht. Wenn v nicht enthalten ist, dann müssen automatisch alle Kindknoten $u \in U(v)$ enthalten sein. Wenn ein Kindknoten $u' \in U(v)$ nicht in der Knotenüberdeckung enthalten wäre, dann würde die Kante (u', v) nicht abgedeckt.

Wenn v in der Knotenüberdeckung enthalten ist, sind automatisch alle Kanten von v zu dessen Kindern $u \in U(v)$ abgedeckt. In diesem Fall können die Kinder $u \in U(v)$ in der Knotenüberdeckung enthalten sein, müssen aber nicht. Ob sie in die Lösungsmenge aufgenommen werden, wird über das Gewicht der Knotenmenge $\tau(u)$ bzw. $\tau'(u)$ entschieden. Das Gewicht der beiden Lösungen wird für jeden Knoten wie folgt hergeleitet

$$\tau(v) = \Sigma_{u \in U(v)} \tau'(u))$$

und

$$\tau'(v) = w(v) + \Sigma_{u \in U(v)} \left(\min\left(\tau(u), \tau'(u)\right)\right)$$

Die Bestimmung von $\tau(v)$ und $\tau'(v)$ basiert somit auf den Werten $\tau(u)$ und $\tau'(u)$ mit $u \in U(v)$. Demnach kann die Lösung nur für solche Knoten hergeleitet werden, deren Kinder bereits bearbeitet wurden.

In Bäumen gibt es der Definition nach keine Kreise, weshalb es in Bäumen mit $|V| \geq 2$ mindestens einen Knoten geben muss, der nur einen Nachbarn hat. Alle Knoten $b \in V$, mit $|n(b)| \leq 1$ werden als Blätter bezeichnet. Für Blätter ist die Herleitung der Lösungen trivial, da die Teilbäume T_b nur das Blatt selbst beinhalten. Die Gewichte sind wie folgt definiert $\tau(b) = 0$ und $\tau'(b) = w(b)$. Die Blätter stellen in diesem Fall die Rekursionsanker dar. Nach den Blättern werden die Lösungen der Teilbäume der Elternknoten berechnet und dann die derer Elternknoten. Zuletzt wird die Lösung des Wurzelknoten r berechnet. Die Lösung einer der beiden Spezialisierungen des Problems stellt die Lösung des generellen Problems dar. Mit $\min\left(\tau(r), \tau'(r)\right)$ steht somit das Gewicht der minimalen Knotenüberdeckung für den Eingabegraphen G fest.

Für jeden Knoten $v \in V$ muss für jeden Kindknoten $u \in U(v)$ das Minimum zwischen $\tau(u)$ und $\tau'(u)$ bestimmt werden. Die Minimierung über zwei Werte stellt eine elementare Operation dar und ist somit in $O(1)$ ausführbar. Diese Operation muss für jedes Knotenpaar (v, u) ausgeführt werden. Somit wird dieser Vergleich für jede Kante genau einmal ausgeführt, womit die Komplexität des Algorithmus in $O(|E|)$ liegt.

Wenn bei der Berechnung von $\tau(v)$ und $\tau'(v)$ für jede Kante notiert wird, welcher Wert $\tau(u)$ oder $\tau'(u)$ mit $u \in U(v)$ verwendet wird, kann für die Kinder hergeleitet werden, ob diese in der Knotenüberdeckung enthalten sind. Auf diese Weise kann die Lösungsmenge der Knotenüberdeckung rekonstruiert werden und das mit einem Aufwand von $O(|E|)$. □

Inhaltsverzeichnis

1.1.6 Problemdefinition

In diesem Kapitel werden die Problemstellungen des Correlation Clusterings und des Fair Correlation Clusterings genauer definiert. Die Definitionen folgen denen von Casel et. al. [12].

Inhaltsverzeichnis

1.1.6.1 Correlation Clustering

Definition 1.1.64 (Correlation Clustering) *Sei ein Graph $G = (V, E)$ gegeben. Die Lösung für das Correlation Clustering ist definiert als eine Partition P über G, sodass die Summe der Kanten deren Endknoten in verschiedenen Clustern liegen zusammen mit der Summe der Knotenpaare, die in einem Cluster liegen, sich aber keine Kante teilen, minimiert wird. Es gilt demnach die folgende Kostenfunktion zu minimieren*

$$cost(G, P) = |\left\{\{u, v\} \in \binom{V}{2} \backslash E \mid P[u] = P[v]\right\}| + |\{\{u, v\} \in E \mid P[u] \neq P[v]\}|$$

, wobei $P[v]$ das Cluster angebe, in dem sich der Knoten v befindet.

Der erste Summand wird als intra-cluster Kosten bezeichnet (intra: lat. innerhalb), also die Anzahl der Knotenpaare innerhalb eines Clusters, die nicht adjazent zueinander sind.

$$\psi = |\left\{\{u, v\} \in \binom{V}{2} \backslash E \mid P[u] = P[v]\right\}|$$

Der zweite Summand wird als inter-cluster Kosten (inter: lat. zwischen) bezeichnet. Er gibt die Anzahl der Kanten an, deren Endknoten in verschiedenen Clustern liegen. Die Kanten, die zu den inter-cluster Kosten beitragen werden als geschnitten bezeichnet.

$$\chi = |\{\{u, v\} \in E \mid P[u] \neq P[v]\}|$$

Es sei zu beachten, dass es sich bei der gegebenen Definition um ungewichtete vollständige Graphen handelt. In vielen Publikationen werden die Kanten mit '+' und '-' kategorisiert, um "Ähnlichkeit" und "Unähnlichkeit" anzugeben. In der hier verwendeten Definition wird die Existenz eine Kante zwischen Knotenpaaren zur Angabe von Ähnlichkeit verwendet. Alle Knotenpaare, die nicht Endknoten einer gemeinsamen Kante sind, werden als nicht ähnlich zueinander interpretiert. [12]

Das Problem des Correlation Clusterings lässt sich auch mit gewichteten Kanten ausführen, dabei werden statt der Anzahl der geschnittenen Kanten, die Gewichte der Kanten aufsummiert. [19]

Inhaltsverzeichnis

1.1.6.2 Fair Correlation Clustering

Das Fair Correlation Clustering stellt eine Generalisierung des Correlation Clusterings dar, das auf gefärbten Graphen ausgeführt wird. Bei diesem Problem wird die Anforderung gestellt, dass das Verhältnis zwischen den Farben im Graphen auf jedes Cluster der Lösungsmenge übertragen wird. Eine Teilmenge S einer gefärbten Menge V wird als fair bezeichnet, wenn das Verhältnis der Farben in V auf S übertragen wird.

Definition 1.1.65 (Faire Teilmenge) *Sei V eine Menge gefärbter Elemente, mit $k \in \mathbb{N}_{>0}$ Farben. Sei die Färbung eines Elements aus V durch die Abbildung $c : V \to [k]$ gegeben, mit $[k] = \{1, 2, ..., k\}$. Die Teilmengen V_i seien definiert als die Teilmenge aller Elemente $v \in V$, welche die gleiche Farbe i aufweisen, also $V_i = \{u \in V | c(u) = i\}$. Sei die Teilmenge $S_i \subseteq S$ analog zu V_i definiert als $S_i = \{u \in S | c(u) = i\}$. Jede Teilmenge S für die gilt $\frac{|S_i \cap V_i|}{|S_i|} = \frac{|V_i|}{V}$, ist eine faire Teilmenge von V.*

Mit Hilfe der fairen Teilmenge kann das Problem des Fair Correlation Clusterings wie folgt definiert werden.

Definition 1.1.66 (Fair Correlation Clustering) *Sei $G = (V, E)$ ein gefärbter Graph, mit der Färbung $c : V \to [k]$, mit $k \in \mathbb{N}_{>0}$. Jede Partition über der Knotenmenge V, deren Cluster faire Teilmengen von V darstellen, wird als faires Clustering bezeichnet. Beim Problem des Fair Correlation Clusterings wird nach einem fairen Clustering über V mit minimalen Kosten gesucht.*

Das Fair Correlation Clustering stellt eine Generalisierung des Correlation Cluster dar und kann somit nicht algorithmisch einfacher sein als das Correlation Clusterings selbst, welches NP-schwierig ist.

Inhaltsverzeichnis

1.2 Beitrag

Der Beitrag dieser Arbeit zum Wissenschaftlichen Diskurs besteht in der Entwicklung zweier Algorithmen zur Lösung des Fair Correlation Clusterings beschränkt auf Eingaben in Form von Kaktusgraphen. Einer der beiden Algorithmen ist auf Kaktusgraphen mit zwei Farben im Verhältnis von 2:1 beschränkt, während der andere beliebig viele Farben in beliebigem Verhältnis verarbeiten kann. Darüber hinaus wird die Komplexitätsgrenze der Algorithmen bestimmt.

Als Basis der Algorithmen werden die von Casel et. al. in [12] vorgestellten Algorithmen verwendet. Bei diesen handelt es sich um dynamische Programme, welche nur Eingabeinstanzen des Fair Correlation Clustering in Form von Wäldern verarbeiten. Auch bei diesen verarbeitet der eine Algorithmus nur Eingaben mit zwei Farben im Verhältnis 2:1 und der andere Algorithmus beliebig viele Farben in beliebigem Verhältnis. Der Algorithmus mit stärkeren Beschränkungen ist in polynomieller Zeit ausführbar, während der andere nicht effizient ist.

Die Algorithmen werden so aufgearbeitet, dass das dynamische Programm zur Verarbeitung von Kaktusgraphen implementiert werden kann, welches von Buchin in [10] zur Lösung eines anderen Partitionierungsproblems verwendet wird.

Als Ergebnis der Komplexitätsanalyse stellt sich heraus, dass die Problemdefinitionen mit Beschränkung auf Kaktusgraphen in denselben Komplexitätsklassen liegen, wie die mit Beschränkung auf Wälder. Das Fair Correlation Clustering für Kaktusgraphen mit zwei Farben im Verhältnis von 2:1 kann in polynomiell Zeit berechnet werden und Kaktusgraphen mit beliebig vielen Graphen mit beliebigem Verhältnis nicht. Damit wird die Menge der bekannten effizient lösbaren Probleminstanzen erweitert. Als Nebenprodukt der Umformulierung der Algorithmen aus [12] werden auch für diese neue Komplexitätsgrenzen erkennbar, welche keinen Einfluss auf die Komplexitätsklassen haben.

Inhaltsverzeichnis

1.3 Verwandte Arbeiten

In diesem Kapitel werden die Ergebnisse verwandter Arbeiten zusammengefasst. Im ersten Unterkapitel 1.3.1 wird der Forschungsstand zu Correlation Clustering vorgestellt, mit Fokus auf die verschiedenen Ansätze zum Umgang mit der NP-Vollständigkeit. Im darauf folgenden Unterkapitel 1.3.2 werden die unterschiedlichen Definitionen der algorithmischen Fairness vorgestellt. Im letzten Unterkapitel 1.3.3 wird das Fair Correlation Clustering und die Komplexitätsgrenzen verschiedener Eingabeformen vorgestellt.

Inhaltsverzeichnis

1.3.1 Correlation Clustering

Der Name Correlation Clustering wurde erstmals von Bansal, Blum und Chawla in [6] verwendet. Ähnliche Problemdefinitionen werden bereits seit den 1960er-Jahren behandelt. Das Problem des Correlation Clustering ist ebenfalls unter dem Namen Cluster Editing bekannt. [12]

Die mathematische Definition für das Correlation Clustering wird in Kapitel 1.1.6.1 gegeben.

In [6] stellen Bansal et. al. das Correlation Clustering für vollständige ungewichtete Graphen vor. Jeder Kante wird ein Label mit Wert "+" oder "-" zugeordnet. Die Label geben an, ob die Endknoten sich ähneln oder unterscheiden. Es werden zwei unterschiedliche Zielsetzung beschrieben: das Min-Disagreement, bei welchem versucht wird, die Anzahl an Endknoten von "+"-Kanten zu minimieren, die sich in unterschiedlichen Clustern befinden, und das Max-Agreement, bei welchem das Ziel darin besteht, die Anzahl der Endknoten von "+"-Kanten in identischen Clustern zu maximieren. Anders ausgedrückt stellt das Min-Disagreement die Minimierung der inter-cluster Kosten und das Max-Agreement die Minimierung der intra-cluster Kosten dar.

Über die Problemdefinition hinaus wird in [6] bewiesen, dass das Correlation Clustering NP-schwierig ist. Dazu wird eine Reduktion des X3C-Problems auf das Correlation Clustering vorgestellt. Das X3C-Problem (exact 3 cover) ist eine NP-schwierige Spezialisierung des NP-vollständigen Problems der exakten Überdeckung. Die exakte Überdeckung ist eines der 21 klassischen NP-vollständigen Probleme, die in [28] definiert werden. Das Problem gibt eine Menge V vor sowie eine Menge an Teilmengen $S \subseteq 2^{[V]}$, für die wiederum eine Teilmenge $T \subseteq S$ gesucht wird, für die gilt, dass alle Elemente in T paarweise disjunkt sind, also $\forall(t_1, t_2) \in \{(t_1, t_2) \in T^2 | t_1 \neq t_2\} : t_i \cap t_j = \emptyset$ und $\bigcup_{t \in T}(t) = \bigcup_{s \in S}(s)$. Die Spezialisierung X3C verlangt, dass für jede Teilmenge in $s \in S$ gelte $|s| = 3$. Für die Reduktion selbst sei auf [6] verwiesen.

Durch die algorithmische Komplexität des Problems widmen sich die meisten Arbeiten zu Correlation Clustering entweder Approximationsalgorithmen für das Correlation Clustering selbst, exakten Berechnungen zu Spezialisierungen des Correlation Clustering oder Approximationsalgorithmen für Spezialisierungen.

Das Min-Disagreement und das Max-Agreement beschreiben zueinander duale Probleme. Die Ergebnisse der beiden dualen Ansätze sind in Hinblick auf die Optimalität äquivalent zueinander, weil aus der Minimierung des einen automatisch die Maximierung des anderen hervorgeht und umgekehrt. Für Minimierung und Maximierung wurden jedoch unterschiedliche Komplexitätsgrenzen ermittelt. [6][7][39]

Folgend werden die Arbeiten zu Approximationsalgorithmen für das Max-Agreement zusam-

mengefasst. Danach folgen die Ergebnisse zu Min-Disagreement (die vorliegende Arbeit löst das Fair Correlation Clustering über das Min-Disagreement, also die Minimierung der inter-cluster Kosten).

Für die Maximierung ist eine triviale Approximation mit einer Approximationsgüte von $0,5$ möglich (die Definition des Begriffs Approximationsgüte ist in Kapitel 1.1.3 gegeben).

Zur Berechnung der trivialen Approximation werden entweder alle Knoten zusammen in genau ein Cluster oder jeder Knoten einzeln in ein eigenes Cluster gruppiert. Aus diesen beiden Alternativen wird die Lösung danach ausgewählt, welche die geringeren Kosten aufweist. Dieser Approximationsalgorithmus funktioniert nicht nur mit Eingaben in Form von vollständigen ungewichteten Graphen, sondern auch mit generellen gewichteten Graphen.

Die in [6] gegebene Definition bezieht sich auf ungewichtete Graphen. Die gegebene Definition kann jedoch als Spezialfall gewichteter Graphen interpretiert werden, bei welcher die Gewichte auf die beiden Werte $\{+1, -1\}$ beschränkt werden. Dementsprechend kann das Correlation Clustering auch für gewichtete Graphen mit beliebigen Kantengewichten definiert werden. Da es sich um eine Spezialisierung handelt kann die in [6] gegebene Problemstellung auch von Algorithmen für generelle gewichtete Graphen gelöst werden. Daher werden in vielen Publikationen auch Algorithmen zur Lösung des Correlation Clusterings für gewichtete Graphen erwähnt. Bei gewichteten Graphen wird die Höhe der Kantengewichte als Maß für Ähnlichkeit interpretiert und die inter- und intra-cluster Kosten werden durch die Summe der positiv und negativ gewichteten Kanten gebildet (siehe Definition in Kapitel 1.1.6).

Bansal et. al. geben in [6] neben der Definition des Correlation Clusterings und dem Beweis für dessen NP-Schwierigkeit auch ein $(\epsilon \cdot |V|^2)$-PTAS-Algorithmus für das Maximierungsproblem. Der Algorithmus basiert darauf, Dreiecke zu identifizieren, in denen zwei "+"-Kanten und eine "-"-Kante vorhanden sind. Dieser Ansatz basiert auf der Eigenschaft, dass Eingabegraphen, welche die Transitivität erfüllen ein perfektes Clustering ermöglichen, also eines mit inter- und intra-cluster Kosten von jeweils Null. Für solche Eingabegraphen ist die optimale Lösung gegeben durch ein Clustering, in dem jede Zusammenhangskomponente ein Cluster darstellt (die Definition für Zusammenhangskomponenten wird in Kapitel 1.1.4 gegeben). Dieser Algorithmus ist angelehnt an einen Algorithmus zur Lösung des MAXCUT-Problems auf dichten oder vollständigen Graphen.

Darüber hinaus wird ein Approximationsalgorithmus mit konstanter Approximationsgüte für das Min-Disagreement gegeben. [39]

Laut Becker [7] geben Charikar, Guruswami und Wirth in [13] einen $0,7664$-Approximationsalgorithmus für generelle Eingabegraphen. Laut Cambus, Choo, Miikonen und Uitto [11] wird der beste Approximationsalgorithmus für das Maximierungsproblem von Swamy in [39] vorgestellt. Dieser hat eine Approximationsgüte von $0,7666$ und funktioniert für generelle gewichtete Eingabegraphen. Laut [7] liegt die beste Approximationsgüte für das Maximierungsproblem für generelle ungewichtete Graphen bei konstanten $0,7776$, der zugehörige Algorithmus wird in [39] vorgestellt. Die Spezialisierung ermöglicht also eine genauere Approximation.

Unabhängig voneinander beweisen Charikar et. al. in [13] parallel zu Demaine, Emanuel, Fiat und Immorlica in [18], dass das Min-Disagreement APX-schwierig ist. [30][39]

Obendrein geben Charikar et. al. einen 4-Approximationsalgorithmus für das Minimierungsproblem für Eingaben in Form von vollständigen Graphen. Charikar et. al. und Demaine et. al. geben parallel einen $O(log(n))$-Approximationsalgorithmus für gewichtete Graphen. [7]

Demaine und Immorlica betrachten in [19] eine Variante, bei der die Kanten neben dem Label $\{''+'', ''-''\}$ zusätzlich positive Gewichte aufweisen. Diese Definition lässt sich vereinfachen, indem die Label als Faktoren $\{+1, -1\}$ mit den Gewichten multipliziert werden; danach können die Label

vernachlässigt werden und es kann rein mit den Gewichten gearbeitet werden.

Laut Casel, Friedrich, Schirneck und Wietheger [12] hat der aktuell beste Approximationsalgorithmus für das Minimierungsproblem eine Approximationsgüte von $1,994 + \epsilon$, vorgestellt von Cohen-Addad, Lee und Newman in [15]. In [13] wurde bewiesen, dass das Problem der Minimierung APX-schwierig ist, es also keinen PTAS-Algorithmus mit beliebig guter Approximationsgüte geben kann.

In [27] betrachten Kalhan, Makarychev und Zhou das Correlation Clustering mit local Objectives (dt. lokale Zielsetzungen). Bei diesem wird die Kostenfunktion durch die sogenannten l_q Normen definiert. Zu dieser Spezialisierung behandeln sie das MIN-Disagreement für vollständige Graphen. Dazu wird ein Disagreement-Vektor $x \in \mathbb{N}^n$ definiert, dessen i-tes Element die Anzahl der Disagreements am i-ten Knoten angibt. Mit diesem Vektor wird dann die l_q Norm gebildet und als Kostenfunktion verwendet, wobei die l_q definiert ist als $||x||_q = (\Sigma_u(x_u^q))^{\frac{1}{q}}$. Für dieses Problem wurde ein 5-Approximationsalgorithmus angegeben und für generelle gewichtete Graphen ein $O(n^{\frac{1}{2}-\frac{1}{2q}} \cdot log^{\frac{1}{2}-\frac{1}{2q}}(n))$-Approximationsalgorithmus.

In [1] wird Correlation Clustering als die natürlichste Form des Clusteringproblems beschrieben. Laut [12] gibt es jedoch eine noch generellere Darstellung des Correlation Clustering, bei welchem jeder Kante zwei Werte zugeordnet werden, einer als Maß für die Ähnlichkeit und einer als Maß für die Unterschiedlichkeit eines Knotenpaares.

Der Vorteil des Correlation Clustering gegenüber älteren Clusteringproblemen besteht darin, dass potentielles Bias auf die Definition der Ähnlichkeit zwischen Datenobjekten beschränkt wird. Ausgenommen davon sind unvollständige Eingabegraphen, bei denen die Metrik der Ähnlichkeit nicht für jedes Knotenpaar berechnet wird.[7][39]

Inhaltsverzeichnis

1.3.2 Algorithmische Fairness

Dänzer beschreibt in [17], dass Intuitives Verständnis von Fairness haben. Die genaue Definition des Begriffes ist in den Geisteswissenschaften viel diskutiert. Auch die mathematische Definition von Fairness ist daher gar nicht so offensichtlich.

Die Frage nach einer mathematischen Definition für Fairness entstammt Konflikten bei der Anwendung statistischer Analysen mit Gesetzen gegen Diskriminierung. In den Vereinigten Staaten werden zunehmend Algorithmen zur Entscheidungsfindung verwendet, um Personengruppen direkt und indirekt Ressourcen zuzuordnen. Damit die Anwendung der Algorithmen nicht mit dem Gesetz in Konflikt gerät, dürfen seit der Verabschiedung des Gesetzes zum "disparate impact" bestimmte personenbezogene Merkmale in solchen Anwendungsfällen keinen Einfluss auf die Entscheidungsfindung haben. Unter den geschützten Merkmalen sind unter anderem die ethnische Zugehörigkeit, das Geschlecht, die Religion und die Nationalität. [4][41]

Die Literatur zum Thema algorithmische Fairness verzeichnet in den letzten Jahren einen starken Zuwachs, speziell im Kontext des maschinellen Lernens. Das Ziel der Forschung besteht darin Kriterien und Algorithmen zu entwickeln, die gewährleisten, dass Objekte mit vorgegebenen Merkmalen in keinem Cluster unter- oder überrepräsentiert werden. [4][35]

Pessach und Shmueli unterscheiden in [35] zwei Arten von unfairer Benachteiligung. Das ist zum einen das "disparate treatment" (dt. unterschiedliche Behandlung), welche die absichtliche Be-

nachteiligung beschreibt und zum anderen der "disparate impact" (dt. unterschiedliche Auswirkung), welche die nicht vorhergesehene Benachteiligung beschreibt, die bspw. durch die Unterrepräsentation im Datensatz hervorgerufen werden kann. Eine gängige Strategie zur Bekämpfung von absichtlicher Benachteiligung liegt darin, Merkmale zu definieren, die geschützt werden sollen und diese aus den Daten komplett zu entfernen. In diesem Fall kann jedoch die nicht beabsichtigte Benachteiligung immer noch greifen und ggf. noch für nicht geschützte Merkmale verstärkt werden.

Pessach und Shmueli geben folgende Definitionen für das Maß von Fairness:

Definition 1.3.1 (Disparate Impact) *Sei S das zu schützende Attribut, wobei Datenobjekte mit $S = 1$ die privilegierte Gruppe und solche mit $S = 0$ die unterprivilegierte Gruppe darstelle. Sei $\widehat{Y} = 1$ eine als positiv betrachtete Ausgabe (bspw. eine Zusage zu einem Job, oder die Einteilung in eine Gruppe mit niedrigerer Gefahrenstufe). Sei mit*

$$P[\widehat{Y} = 1|S \neq 1]$$

die Wahrscheinlichkeit gegeben, dass ein Datenobjekt mit $S \neq 1$ in die positive Klasse $\widehat{Y} = 1$ eingeordnet wird, bzw. eine positive Ausgabe gegeben wird. Dabei gelte $0 \leq P[\widehat{Y} = 1|S \neq 1] \leq 1$. Dann gilt ein Clustering als fair, wenn die Wahrscheinlichkeit, eines positiven Ausgangs, für Objekte mit dem geschützten Attribut im Verhältnis zur Wahrscheinlichkeit eines positiven Ausgangs für Objekte ohne dieses Attribut eine untere Grenze hat, die mit $\epsilon \geq 0$ vorgegeben werden kann, also

$$\frac{P[\widehat{Y} = 1|S \neq 1]}{P[\widehat{Y} = 1|S = 1]} \geq 1 - \epsilon$$

Das gesetzlich in den Vereinigten Staaten vorgegebene Verhältnis liegt bei 80%, sodass $\epsilon \leq 0,2$. [24]

Definition 1.3.2 (Demographic Parity) *Die Demographic Parity (dt. demographische Parität, demographische Gleichheit) ist analog zum Disparate Impact definiert, mit dem Unterschied, dass statt des Verhältnisses der Betrag der Differenz als Maß der Fairness genutzt wird*

$$|P[\widehat{Y} = 1|S = 1] - P[\widehat{Y} = 1|S \neq 1]| \leq \epsilon$$

Im Gegensatz zu den bisherigen Maßen für Fairness muss für die folgenden beiden Definitionen die tatsächliche Klassifizierung bekannt sein.

In der Statistik wird mit dem Alphafehler oder auch *Fehler 1. Art* eine positive Prognose bezeichnet, die eigentlich hätte negativ ausfallen müssen.

Definition 1.3.3 (Equalized Odds) *Nach dem Maß der Equalized Odds (dt. gleichgesetzte Wahrscheinlichkeiten) gilt eine Klassifizierung als fair, wenn der Betrag der Differenz zwischen der Wahrscheinlichkeit eines Alphafehlers für Objekte mit $S = 1$ und der Wahrscheinlichkeit eines Alphafehlers für Objekte mit $S \neq 1$ nach oben beschränkt sind*

$$|P[\widehat{Y} = 1|S = 1, Y = 0] - P[\widehat{Y} = 1|S \neq 1, Y = 0]| \leq \epsilon$$

, sowie der Betrag der Differenz zwischen der Wahrscheinlichkeit einer korrekten positiven Klassifizierung für Objekte mit $S = 1$ und solche mit $S \neq 1$

$$|P[\widehat{Y} = 1|S = 1, Y = 1] - P[\widehat{Y} = 1|S \neq 1, Y = 1]| \leq \epsilon$$

, mit $\epsilon \geq 0$, wobei Y den tatsächlichen Wert angebe, den es zu prognostizieren gilt.

Definition 1.3.4 (Equal Opportunity) *Sei S ein Merkmal und Objekte mit $S = 1$ privilegiert sowie Objekte mit $S \neq 1$ nicht privilegiert. Beim Equal Oppurtunity werden Klassifikationen als fair bezeichnet, wenn die zweite Formel des Equalized Odds erfüllt wird, also die Differenz zwischen der Wahrscheinlichkeit einer korrekten positiven Ausgabe für ein Objekt mit $S = 1$ und solchen mit $S \neq 1$ nach oben begrenzt ist*

$$|P[\widehat{Y} = 1|S = 1, Y = 1] - P[\widehat{Y} = 1|S \neq 1, Y = 1]| \leq \epsilon$$

, mit $\epsilon \geq 0$

Die bisherigen Maße für Fairness in Klassifikationen beziehen sich ausschließlich auf die Verhältnisse bzw. Differenzen zwischen Objekten mit und ohne das zu schützende Merkmal.

Definition 1.3.5 (Individual Fairness) *Sei die Abbildung $d : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ gegeben, welche die Ähnlichkeit zweier Objekte in Form einer Distanz angibt, sodass zueinander nahe Objekte als ähnlich zueinander definiert seien. Sei $S^{(\cdot)}$ das sensible Attribut und $X^{(\cdot)}$ die restlichen Attribute, wobei $\cdot$ durch einen Wert $i, j \in \mathbb{N}$ ersetzt wird, um ein bestimmtes Objekt des Datensatzes zu adressieren. Die Klassifikation gelte genau dann als fair, wenn der Betrag der Differenz der Wahrscheinlichkeiten zweier sehr ähnlicher Objekte in derselben Gruppe klassifiziert zu werden, nach oben beschränkt werden kann*

$$|P(\widehat{Y}^{(i)} = y|X^{(i)}, S^{(i)}) - P(\widehat{Y}^{(j)} = y|X^{(j)}, S^{(j)})| \leq \epsilon, \text{ wenn } d(i, j) \cong 0$$

, mit $\epsilon \geq 0$.

Die gegebenen Definitionen setzen voraus, dass der Anwender Merkmale vorgibt, für die alle Objekte, die dieses Merkmal aufweisen, in den Clustern weder über- noch unterrepräsentiert werden sollen. Laut [2] ist nachweisbar, dass der Schutz eines Merkmals zur erhöhten Diskriminierung anderer Merkmale führt. Das bedeutet, dass durch den Ausschluss von Merkmalen, die als schützenswert gelten, nicht nur kein Garant dafür gegeben werden kann, dass nicht trotzdem aufgrund korrelierender Eigenschaften diskriminiert wird, sondern gerade durch deren Ausschluss nicht geschützte Gruppen aktiv benachteiligt werden. [12][2]

Algorithmische Fairness nach dem Gesetz des desparate impact ist somit weniger fair. Es gibt jedoch Szenarien in denen algorithmische Fairness ihr Ziel erreichen kann. Algorithmischer Bias kann vor allem dann entstehen, wenn nur wenige Objekte mit dem gleichen Merkmal im Datensatz vorhanden sind, sodass wenige Objekte einen unverhältnismäßig großen Einfluss auf die Gruppierung aller Objekte mit diesem Merkmal haben können. Wenn also bspw. in einer bestimmten Personengruppe nur zwei Personen mit einem bestimmten Merkmal erfasst wurden und eine dieser Personen als gefährlich gilt, würde dadurch ohne Intervention abgeleitet werden, dass Personen mit diesen Merkmalen mit 50%-iger Wahrscheinlichkeit gefährlich sind. Schützenswerte Merkmale müssen demnach in vorliegenden Datensätzen identifiziert werden, indem deren Unterrepräsentation nachgewiesen wird. [12][2]

Laut [35] muss bei allen Clusteringproblemen stets ein Kompromiss zwischen Fairness und Präzision geschlossen werden. Dieser Kompromiss wird in den Studien [32] und [16] behandelt.

Es können nicht mehrere Maße für Fairness parallel maximiert werden. Die Definition für algorithmische Fairness muss daher vom Anwender gewählt werden. [35]

Über die Definitionen von Fairness hinaus gibt [35] auch noch Methoden zur Steigerung von Fairness in maschinellem Lernen an. Diese werden in drei Gruppen eingeteilt: pre-process (vor der Ausführung), in-process (während der Ausführung), und post-process (nach der Ausführung). Methoden der Gruppe pre-process verändern die Daten vor der Verarbeitung durch einen Klassifizierungsalgorithmus. Das umfasst das Ausschließen von Datenobjekten, das Ausschließen von

Merkmalen, und das Verändern von Werten, um die Daten einheitlicher erscheinen zu lassen. Dazu können bspw. die Merkmale so manipuliert werden, dass Datenobjekten zufällig in privilegierte und unterprivilegierte Gruppen eingeordnet werden. Methoden der Gruppe in-process beziehen sich auf die Anpassung der Lernalgorithmen. Dazu werden bspw. Kosten in Abhängigkeit des Fairnessmaßes verwendet, welche es zu optimieren gilt. Methoden der Gruppe post-process umschreiben die nachträgliche Manipulation der Clusterings. Dazu werden nachträglich so viele zufällige Datenobjekte in andere Gruppen eingeteilt, bis die Anforderungen für Fairness erfüllt sind.

Die Forschung zum Thema algorithmische Fairness im Zusammenhang von Clusteringalgorithmen wird erstmals von Chierichetti, Kumar, Lattanzi und Vassilvitskii in [14] behandelt. Dort wird die Definition fairer Teilmengen und fairer Clusterings, sowie Algorithmen zur Approximation fairer Lösungen des fair k-median- und des fair k-center-Clustering vorgestellt, zweier Clusteringalgorithmen, bei denen die Anzahl der Cluster im Ergebnis durch Parameter vorgegeben wird. [4][35]

In [14] werden die Eingaben für das k-median- und das k-center-Clustering zuerst in sogenannte Fairlets unterteilt, faire Teilmengen, die als geeignet betrachtet werden, das Clusteringproblem zu lösen. Nachfolgende Arbeiten haben die Ergebnisse aus [14] auf andere Clusteringprobleme übertragen, wie das k-means-Clustering. [3][9][8][5][29]

Das Fair Correlation Clustering nutzt faire Teilmengen, welche der Definition des Disparate Impact gleichen, mit $\epsilon = 0$ (siehe Kapitel 1.1.6.2). [12]

Inhaltsverzeichnis

1.3.3 Fair Correlation Clustering

Beim Fair Correlation Clustering handelt es sich um eine Generalisierung des Correlation Clusterings, bei dem die Knotenmenge gefärbt ist und jedes Cluster das Verhältnis der Farben aus der Knotenmenge bewahren muss. Die mathematische Definition des Fair Correlation Clustering wird in Kapitel 1.1.6 gegeben.

Das Correlation Clustering stellt demnach eine Spezialisierung des Fair Correlation Clusterings dar, bei welchem alle Knoten dieselbe Farbe haben. Für das Fair Correlation Clustering kann es daher keinen effizienteren Algorithmus geben, als für das Correlation Clustering. Das Fair Correlation Clustering ist somit mindestens NP-schwierig. Die meisten Arbeiten zu Fair Correlation Clustering beschränken sich deshalb auf die Verbesserung von Approximationsalgorithmen oder exakte Lösungen für Spezialisierungen des Problems, wie auch beim Correlation Clustering. Folgend werden zunächst ausgewählte wissenschaftliche Beiträge zur Lösung auf generellen Graphen vorgestellt, bevor Ergebnisse zu Algorithmen und Komplexitätsgrenzen bezüglich Eingabegraphen in Form von Wäldern präsentiert werden.

Inhaltsverzeichnis

1.3.3.1 Fair Correlation Clustering für generelle Graphen

Laut Casel et. al. [12] wird das Thema Fair Correlation Clustering erstmals parallel in den beiden Publikationen von Ahmadi et. al. [1] und Ahmadian, Epasto, Kumar und Mahdian [4] behandelt. Deren Ergebnisse beziehen sich auf vollständige ungewichtete Graphen.

In Ahmadian et. al. [4] und [3] wird die Fairness über einen Faktor α definiert, welcher für

jede Farbe einheitlich vorgibt, wie viele Knoten in jedem Cluster enthalten sein müssen. Diese Definition des Fair Correlation Clustering deckt sich nicht mit der Definition von Casel et. al. [12] und somit auch nicht mit der in dieser Arbeit verwendeten Definition. Für Eingaben mit k Farben und $\alpha = \frac{1}{k}$ stellt die Definition aus [1] einen Sonderfall der Definition dar, die in [12] verwendet wird. Es handelt sich um den Spezialfall, dass das Verhältnis c_i der Farben im Eingabegraphen mit $\forall i \in [k] : c_i = 1$ gegeben ist, jeder Farbe also gleich viele Knoten zugeordnet werden. In [1] wird für diesen Fall ein $16,48 \cdot k^2$-Algorithmus und für den Spezialfall $k = 2$ sogar ein Algorithmus mit einer Approximationsgüte von 256 vorgestellt.

In [1] wird ein $O(c^2)$-Approximationsalgorithmus für Eingaben mit zwei Farben und einem Verhältnis von $1 : c$ vorgestellt. Darüber hinaus wird für Eingaben genereller Graphen mit zwei Farben im Verhältnis 1 : 1 ein Algorithmus mit einer Approximationsgüte von $3\beta + 4$ vorgestellt. Das Ergebnis dieses Approximationsalgorithmus basiert auf dem Ergebnis einer Approximation für das Correlation Clustering, aus dem dann ein Fair Correlation Clusterings erstellt wird. β stellt die Approximationsgüte des verwendeten Algorithmus zur Approximation des Correlation Clusterings dar. Für Eingaben mit beliebig vielen Farben und als einziger Beschränkung für die Verhältnisse $c_1 = 1$ sowie $\forall i \in [k] : c_i \in \mathbb{N}$, wird ein $O(k^2 \cdot max_{1 \leq i \leq k} c_i)$-Approximationsalgorithmus vorgestellt, sowie ein $O(k^2 \cdot max_{2 \leq i \leq k} q_i)$-Approximationsalgorithmus für relaxed Fairness.

Nach [12] ist die zeitlich als nächstes publizierte Arbeit zu dem Thema die von Friggstad et. al. [25]. In dieser wird ein $6,18$-Approximationsalgorithmus für Eingabegraphen mit zwei Farben in einem Verhältnis von 1 : 1 vorgestellt. In [2] wurden die erwähnten Approximationsgüten aus [1] unter Verwendung von relaxed Fairness auf $O(\frac{1}{\epsilon \cdot min_i \alpha_i})$ verbessert, wobei die Anforderung der Fairness in jedem Cluster um maximal $max\{1, \epsilon \cdot |C| \cdot max_i(\alpha_i)\}$ Knoten überschritten werden darf. Dabei wird $\epsilon > 0$ als Parameter vorgegeben. Für Eingaben mit $\forall i \in [k] : \alpha_i = \frac{1}{2}$ erreicht der vorgestellte Algorithmus eine Approximationsgüte von $4 + \frac{1}{\epsilon}$, was im Vergleich zu den vorherigen 256 eine deutliche Verbesserung für Werte $\epsilon > \frac{1}{252}$ darstellt.

Im Bestreben die Probleme algorithmisch leichter definieren zu können, wurde die relaxed (dt. entspannte, gelockerte) Fairness erstmals von Ahmadian et. al. in [3] eingeführt. [1]

Definition 1.3.6 (Relaxed Fairness) *Sei V eine Menge gefärbter Elemente, mit $k \in \mathbb{N}_{>0}$ Farben. Sei die Färbung eines Elements aus V durch die Abbildung $c : V \to [k]$ gegeben, mit $[k] = \{1, 2, ..., k\}$. Seien mit $p_i, q_i \in \mathbb{Q}$ für jede Farbe eine untere und obere Grenze gegeben, mit $i \in [k]$, für die gelte $0 < p_i \leq \frac{|V_i|}{|V|} \leq q_i < 1$, wobei $V_i = \{u \in V | c(u) = i\}$. Dann wird eine Teilmenge $S \subseteq V$ nur dann als relaxed fair bezeichnet, wenn die untere und obere Grenze eingehalten werden, sodass $p_i \leq \frac{|S_i \cap V_i|}{|S_i|} = \frac{|V_i|}{V} \leq q_i$.*

Casel et. al. [12] haben jedoch gezeigt, dass die algorithmische Komplexität des Fair Correlation Clusterings nicht von der exakten Fairness herrührt, sondern hauptsächlich von der Anzahl an Farben und deren Verhältnis zueinander abhängt. Casel et. al. [12] und Ahmadi, Galhotra, Saha und Schwartz [1] haben nachgewiesen, dass das Problem des Fair Correlation Clustering für generelle Graphen bereits für zwei Farben mit einem Verhältnis 1:p, mit $p > 2$ als NP-schwierig gilt, sowie für Farben $k \geq 3$.

Inhaltsverzeichnis

1.3.3.2 Fair Correlation Clustering für Wälder

Casel et. al. [12] beschäftigen sich hauptsächlich mit der Herleitung von Komplexitätsgrenzen für eine Spezialisierung des Fair Correlation Clusterings, welches sich auf Eingaben in Form von Wäldern beschränkt. Wälder stellen eine der einfachsten Form von Graphen dar. Problemspezialisierungen, bei denen die Eingaben auf Wälder beschränkt werden, sind oftmals effizient lösbar, auch wenn das für generelle Eingaben ggf. nicht der Fall ist. Aus diesem Grund werden für viele Probleme zuerst Spezialisierungen mit Beschränkung der Eingaben auf Graphen in Form von Wälder behandelt. Meist ist es möglich aus deren Behandlung Einsichten zu erhalten, die bei der Lösung des generelleren Problems hilfreich sind. Folgende Ergebnisse zur algorithmischen Komplexität beziehen sich ausschließlich auf das Fair Correlation Clustering mit Eingabegraphen in Form von Wäldern.

In [12] wird die Abhängigkeit der Komplexität der Probleminstanzen verschiedener Spezialisierungen von bestimmten Eigenschaften bewiesen. Darunter sind neben der Anzahl der Farben und deren Verhältnissen auch der Grad und der Durchmesser eines Graphen.

Den einfachsten Fall stellen Eingaben mit zwei Farben im Verhältnis 1:1 dar. Casel et. al. [12] zeigen, dass sich diese Spezialisierung auf das Problem des maximalen Matchings auf bipartiten Graphen reduzieren lässt, womit eine Laufzeit $O(|V| + |E|) = O(n)$ möglich ist, mit $n = |V|$. In folgender Tabelle aus [12] werden die Ergebnisse bezüglich der algorithmischen Komplexität des Fair Correlation für Eingaben in Form von Wäldern mit zwei Farben in verschiedenen Verhältnissen angegeben. In der oberen Spalte sind die Verhältnisse der beiden Farben angegeben und darunter die Komplexität, die sich daraus ergibt. Dabei stellt $f(p)$ eine polynomielle Funktion dar. [12]

$1:1$	$1:2$	$1:(\frac{n}{p}-1)$
$O(n)$	$O(n^6)$	$O(n^{f(p)})$

Tabelle 1.3: Komplexitäten Fair Correlation Clustering für Wälder

In der vorliegenden Arbeit wird für das Verhältnis $1:2$ eine tiefere Komplexitätsgrenze von $O(n^3)$ hergeleitet (siehe Kapitel 2.1.1).

Über eine Reduktion des 3-Partition Problem auf das Fair Correlation Clustering beweisen Casel et. al., dass die Komplexität für Eingaben in Form von Wäldern mit einem Verhältnis von $1:c$, mit $c \geq 3$ bereits ab einem Durchmesser von ≥ 4 NP-schwierig ist. (Für den Beweis sei auf [12] verwiesen, mit dem Zusatz, dass für die Summe dreier Zahlen $a_i \in \mathbb{N}_{>0}$, mit $i \in \{1,2,3\}$ gilt $\Sigma_{i=1}^{3} a_i \geq 3$, woraus $c \geq 3$ hervorgeht.)

Casel et. al. beweisen ebenfalls durch die Verwendung der Reduktion des 3-Partition Problems, dass das Fair Correlation Clustering für Wälder mit zwei Farben in einem Verhältnis von $1:c$, mit $c \geq 3$ bereits ab einem Grad von ≥ 5 NP-schwierig ist.

Die folgenden Lemmata werden in [12] hergeleitet und zur Entwicklung der exakten Algorithmen verwendet. Es wird gezeigt, dass immer ein optimales faires Clustering existiert, bei dem alle Cluster die gleiche Größe aufweisen und diese Größe vom Verhältnis der Farben abhängig ist.

Lemma 1.3.1 *Sei ein Eingabegraph in Form eines gefärbten Waldes $F = (V, E)$ gegeben, mit $\Sigma_{i=1}^{k} c_i \geq 3$, wobei $c_1, c_2, ..., c_k$ die Verhältnisse der k Farben seien und der größte gemeinsame Teiler aller c_i sei $ggt(c_1, c_2, ..., c_k) = 1$. Dann haben alle Cluster eines Fair Correlation Clusterings in F die Kardinalität $d = \Sigma_{i=1}^{k} c_i$.*

Beweis: Sei ein beliebiges Fair Clustering P von V gegeben, dann lassen sich die Cluster in faire Cluster der Größe d aufteilen, ohne die Kosten zu erhöhen. Jedes Cluster in P muss mindestens d Knoten enthalten, da mindestens c_i Knoten der Farbe i enthalten sein müssen. Sei ein Cluster S gegeben, mit $|S| > d$, dann muss $|S| = ad$, mit $a \in \mathbb{N}_{\geq 2}$, und von jeder Farbe i müssen genau $a \cdot c_i$ Knoten enthalten sein, damit das Cluster fair ist. Sei P' die Partition, die erzeugt wird, wenn S in S_1 und S_2 aufgeteilt wird, wobei $S_1 \subset S$ eine beliebige faire echte Teilmenge von S sei, mit der Größe d, sodass von jeder Farbe i, c_i viele Knoten enthalten sind und $S_2 = S \backslash S_1$. Seien ψ die intra-cluster Kosten von P über $F[S]$, wobei $F[S]$ der induzierte Graph über der Kantenmenge S über dem Graphen F darstelle. Für die intra-cluster Kosten gilt,

$$cost(F[S], P) \geq \psi \geq \frac{ad(ad-1)}{2} - (ad-1) = \frac{a^2d^2 - 3ad + 2}{2}$$

Das Cluster hat die Kardinalität ad und da es sich um einen Wald handelt, beinhaltet es maximal $ad-1$ Kanten. Im Extremfall schneidet P' alle $ad-1$ Kanten. Die Kosten der neuen Cluster S_1 und S_2 sind geringer als die von S, weil die Kosten der Partition P' wie folgt begrenzt sind

$$\begin{aligned} cost(F[S], P') &= \chi + \psi \\ &\leq ad - 1 + \tfrac{d(d-1)}{2} + \tfrac{(a-1)d\cdot((a-1)d-1)}{2} \\ &= \tfrac{2d^2 + a^2d^2 - 2ad^2 + ad - 2}{2} \end{aligned}$$

Die Differenz zwischen den Kosten der beiden Partitionen liegt bei

$$cost(F[S], P) - cost(F[S], P') \geq \frac{2ad^2 - 2d^2 - 4ad + 4)}{2} = ad(d-2) - d^2 + 2$$

Der Zugewinn in den intra-cluster Kosten bei der Trennung überwiegt ab einer Größe von $d > 3$. Die Kosten sind abhängig von a und d, der Zugewinn wächst mit a, sodass die untere Grenze mit $a = 2$ gegeben ist. Die untere Grenze ergibt sich somit mit $cost(F[S], P) - cost(F[S], P') \geq d^2 - 4d + 2$. Damit hängt die untere Grenze der Differenz der Kosten nur von d ab. Jedes Cluster mit einer Größe von mehr als d, mit $d > 3$ ist suboptimal.

Es bleibt nur noch für den Fall $d = 3$ zu beweisen, dass auch hier die Auftrennung in Cluster der Größe d optimal ist. Mit $d = 3$ ergibt die untere Grenze $cost(F[S], P) - cost(F[S], P') \geq -1$. Die Kosten $cost(F[S], P')$ können verringert werden, indem darauf geachtet wird, bei der Aufteilung von S in S_1 und S_2 mindestens eine Kante aus S auch in S_1 oder S_2 zu beinhalten. Dadurch trägt diese Kante nicht zu den inter-cluster Kosten bei und zwei Knoten, die ansonsten zu den intra-cluster Kosten beitragen würden, werden in zwei verschiedene Cluster aufgeteilt. Dadurch erhöht sich die Differenz der Kosten von -1 um mindestens 2. Die einzigen beiden Farbverhältnisse die mit $d = 3$ möglich sind, sind $1:1:1$ und $1:2$. Sei eine Kante $e = \{u, v\} \in S$ gegeben, mit $e = \{u, v\}$, mit $c(u) \neq c(v)$, wobei die Abbildung $c : V \to [k]$ die Farbe der Knoten angebe. Wenn das Cluster S in die fairen Teilmengen $\{u, v, w\}$ und $S \backslash \{u, v, w\}$ getrennt wird, dann sind die Kosten der Partition P' damit geringer als die von P. Wenn keine solche Kante $\{u, v\}$ existiert, dann ist $F[S]$ nicht zusammenhängend. Dies impliziert für einen Wald, bei einem Verhältnis von 1:1:1, dass es höchstens $3a - 3$ Kanten zwischen den a Knoten jeder Farbe, mit maximal $a - 1$ Kanten inzident zu jedem Knoten gibt. Für das Verhältnis 1:2 würde es maximal $3a - 2$ Kanten in S geben. Selbst wenn all diese Kanten für die Trennung von S geschnitten würden, würde mit $a = 2$ höchstens eine Kante in P' geschnitten wodurch die Kosten um mindestens eins geringer wären als mit P. □

Für bipartite Graphen, und damit insbesondere auch für Bäume, haben die intra-cluster Kosten keinen Einfluss auf die Kosten des Clusterings. Dies wird in folgendem Lemma formalisiert.

Lemma 1.3.2 *Die Kosten eines Fair Correlation Clusterings für bipartite Graphen sind unabhängig von den intra-cluster Kosten.*

Beweis: Die Kosten einer Partition seien mit $cost(P) = \psi + \chi$ gegeben. Wenn eine Kardinalität von d für jedes Cluster angenommen wird, dann sind in jedem der $\frac{|V|}{d}$ Cluster $\frac{d(d-1)}{2}$ Knotenpaare enthalten, von denen jedes Paar intra-cluster Kosten von maximal 1 erzeugt. Für die intra-cluster Kosten gilt somit $\psi = \frac{|V|}{d} \cdot \frac{d(d-1)}{2}$. Eine Kante kann nur entweder zu den intra-cluster Kosten oder den inter-cluster Kosten beisteuern, wodurch für die intra-cluster Kosten in Abhängigkeit der inter-cluster Kosten gilt, dass $\psi = \frac{|V|}{d} \cdot \frac{d(d-1)}{2} - (|E| - \chi)$. Daraus ergibt sich $cost(P) = \chi + \psi = \frac{(d-1)\cdot|V|}{2} - |E| + 2\chi$. Wenn der betrachtete Graph ein Baum ist, dann gilt $|E| = |V| - 1$ und somit $cost(P) = \chi + \psi = \frac{(d-3)\cdot|V|}{2} + 2\chi + 1$, wobei P die Partition der optimalen Lösung sei. □

Für die exakte Lösung von Eingaben mit beliebig vielen Farben in beliebigem Verhältnis wird in [12] eine Komplexität von

$$O(n^{2setvars+stmax+2} \cdot setvars^{setmax})$$

hergeleitet, wobei $setmax = \Sigma_{i=1}^{k} c_i$ und $setvars = \prod_{i=1}^{k}(c_i + 1)$. Auch für diesen Algorithmus wird eine genauere Komplexitätsgrenze hergeleitet.

Inhaltsverzeichnis

Kapitel 2

Hauptteil

In diesem Kapitel werden die beiden Algorithmen zur Lösung des Fair Correlation Clusterings beschränkt auf Kaktusgraphen beschrieben. Die Algorithmen behandeln Kaktusgraphen mit zwei Farben im Verhältnis 2:1 und beliebig vielen Farben in beliebigem Verhältnis.

Als erstes werden in Kapitel 2.1 die Algorithmen aus [12] in überarbeiteter Form vorgestellt, sowohl für das Verhältnis 2:1 in Kapitel 2.1.1, als auch für beliebige Verhältnisse in Kapitel 2.1.2. Die Überarbeitung der Definitionen ist notwendig, um das dynamische Programm zur Verarbeitung von Kaktusgraphen einzubinden, wie in [10] gegeben. Durch die Überarbeitung werden neue tiefere Komplexitätsgrenzen für die Algorithmen aus [12] hergeleitet, was sich mit der Aussage von Casel et. al. in [12] deckt, dass es wahrscheinlich noch genauere Grenzen geben müsse.

Danach folgen in Kapitel 2.2 die beiden neuen Algorithmen für Kaktusgraphen, erst für das Verhältnis 2:1 in Kapitel 2.2.1, dann in Kapitel 2.2.2 für beliebige Verhältnisse. Für diese werden Komplexitätsgrenzen hergeleitet und damit nachgewiesen, dass das Problem für die verschiedenen Vorgaben bezüglich der Färbungen denselben Komplexitätsklassen angehört wie die Problemdefinitionen, die auf Wälder beschränkt sind.

Inhaltsverzeichnis

2.1 Fair Correlation Clustering für Eingabegraphen in Form von Wäldern

In diesem Unterkapitel werden die beiden Algorithmen aus [12] in überarbeiteter Formulierung vorgestellt. Die Überarbeitung der Definitionen dient dazu, die Komplexität des erweiterten Algorithmus herleiten zu können, welcher im nächsten Kapitel vorgestellt wird. Die Algorithmen werden nicht verändert. Es wurden lediglich an einigen Stellen Aspekte definiert, die in [12] keine genaue Definition verlangen, für die Erweiterung auf Kaktusgraphen aber eine Rolle spielen. Für alle Definitionen, die ohne Änderungen übernommen werden, wird dies explizit angegeben. Der direkte Vergleich jeder Definition wird nicht angegeben.

Der ausschlaggebende Unterschied, welcher die Herleitung der genaueren Komplexitätsgrenze erlaubt, liegt in der Definition der Mengen $B(v, h)$ und $A(x)$, sowie die Abbildung $\Delta_v^h(i)$ in Abhängigkeit von i, also der Berechnung der Teilergebnisse von T_v^i. Eine ähnliche Auftrennung wird in [10] für ein anderes Partitionsproblem gegeben, um die Behandlung von Kaktusgraphen zu ermöglichen. Dabei

wird eine zweite Rekursionsebene in das dynamische Programm eingeführt. Statt nur der Rekursion von der Wurzel hin zu den Blättern, wird zusätzlich eine Rekursion bei der Berechnung der Kinder einer Wurzel eingeführt. Dieses Vorgehen ist für die Verarbeitung von Kaktusgraphen unabdinglich.

Zuerst wird in 2.1.1 der Algorithmus zur Lösung des Fair Correlation Clustering in Wäldern mit zwei Farben im Verhältnis 1:2 vorgestellt. In dem darauf folgenden Unterkapitel 2.1.2 wird der Algorithmus für Wälder mit beliebig vielen Farben in beliebigem Verhältnis vorgestellt.

Das Symbol Δ wird eigentlich der Mengenoperation zur Bildung der symmetrischen Differenz zugeordnet (siehe Definition 1.1.13 in Kapitel 1.1.1). Die Definition von Identifikatoren besagt, dass Symbole, die mit Operationen assoziiert werden, nicht als Identifikatoren zulässig seien (siehe Definition 1.1.1 in Kapitel 1.1.1). Dieses Symbol wird in [12] als Identifikator verwendet und für den einfachen Vergleich hier übernommen.

Inhaltsverzeichnis

2.1.1 Wälder mit zwei Farben im Verhältnis 1:2

Der Eingabegraph sei ein gefärbter Wald $F = (V, E)$ mit $k = 2$ Farben im Verhältnis $c_1 : c_2$, mit $c_1 = 1$ und $c_2 = 2$. Zur Verdeutlichung werden die Farben blau und rot angenommen. Der Algorithmus arbeitet in zwei Schritten. Im ersten Schritt wird ein dynamisches Programm ausgeführt, welches über der Knotenmenge V eine Partition mit so vielen zusammenhängenden fairen Teilmengen minimaler Kardinalität bildet, wie möglich. Im zweiten Schritt werden alle übrigen Knoten zu nicht zusammenhängenden fairen Teilmengen vereint.

Folgend abgebildeter Graph dient als Beispiel, an welchem die verschiedenen Definitionen verdeutlicht werden. Der Knoten mit der Nummer 1 sei als Wurzel gegeben.

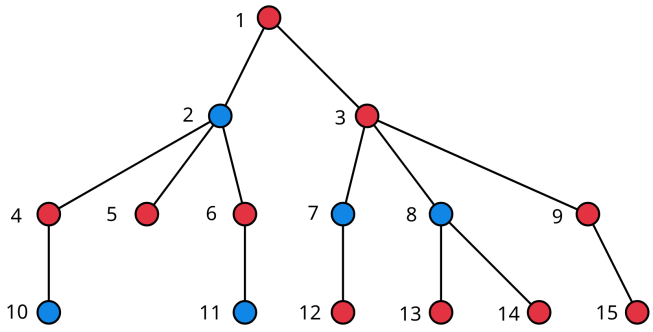

Abbildung 2.1: Eingabegraph in Form eine Baumes

Nach Lemma 1.3.1 gibt es mindestens eine optimale Lösung für das Fair Correlation Clustering, in welchem die Lösung eine Partition darstellt, in der alle Knoten in Cluster mit einem blauen und zwei roten Knoten eingeteilt werden. Cluster mit Knoten dieser Farben werden folgend als brr-

Cluster bezeichnet. Im vorgestellten Algorithmus wird im ersten Schritt eine solche Partition berechnet.

Definition 2.1.1 (Splitting) *Sei ein gefärbter Graph $G = (V, E)$ mit $k \in \mathbb{N}_{>0}$ Farben im Verhältnis $c_1 : c_2 : ... : c_k$, mit $c_1, c_2, ..., c_k \in \mathbb{N}_{>0}$ gegeben. Die Verhältnisse seien so angegeben, dass deren größter gemeinsamer Teiler $ggt(c_1, c_2, ..., c_k) = 1$ sei. Ein Splitting über V stellt eine Partition dar, in der alle Cluster zusammenhängend sind und möglichst viele Cluster fair sind und eine Kardinalität von maximal $d = \Sigma_{i=1}^{k} c_i$ aufweisen. Alle Cluster, die nicht fair sind, seien so groß wie möglich aber kleiner als d.*

Während die Lösung des Correlation Clusterings als Partition über der Knotenmenge definiert wird, kann das Splitting als Partition über der Kantenmenge definiert werden, da es sich bei jedem Cluster per Definition um eine Zusammenhangskomponente handelt.

Beispiel 2.1.1 *In folgender Abbildung wird ein Splitting dargestellt, mit folgenden brr-Clustern $\{1, 2, 5\}$, $\{3, 7, 12\}$ und $\{8, 13, 14\}$. Alle anderen Cluster werden als möglichst große Zusammenhangskomponenten dargestellt. Für gegebenen Eingabegraphen gibt es keine Partition, in der mehr als drei zusammenhängende brr-Cluster möglich wären. Es wären aber einige andere Splittings denkbar. Im Cluster $\{1, 2, 5\}$ könnten statt dem Knoten 5 alternativ entweder der Knoten 4 oder der Knoten 6 enthalten sein. Das Cluster $\{3, 7, 12\}$ könnte alternativ durch die Cluster $\{3, 7, 9\}$, oder $\{1, 3, 7\}$ ersetzt werden. Das Cluster $\{8, 13, 14\}$ muss in jedem Splitting enthalten sein, da ansonsten mit Knoten 8 kein brr-Cluster gebildet werden kann oder der Knoten 7 in keinem zusammenhängenden brr-Cluster enthalten wäre. Unter dem beschriebenen Splitting sind schematisch drei weitere Splittings angegeben.*

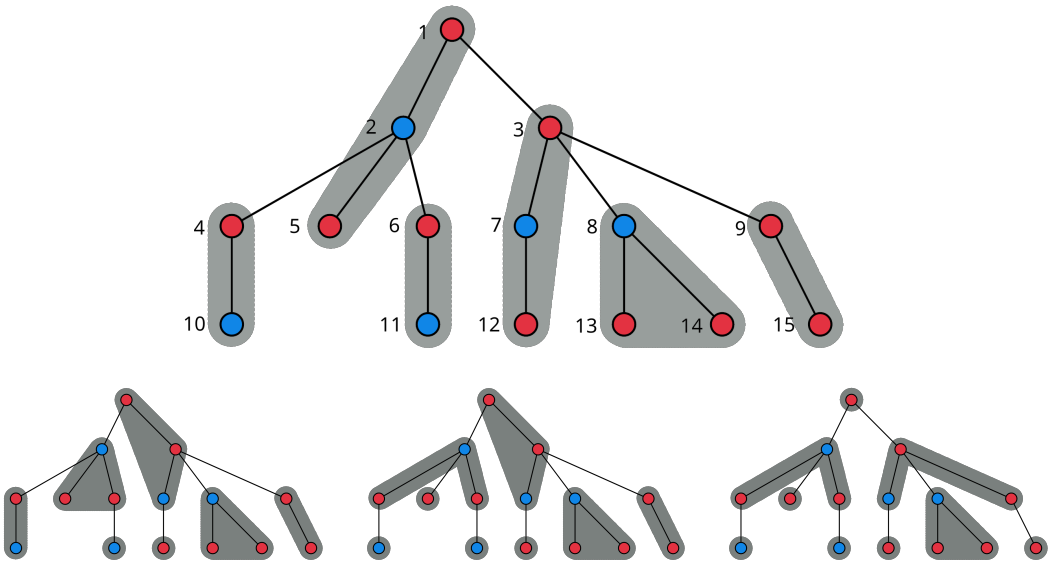

Abbildung 2.2: Beispiel Splitting

Folgendes Lemma ist in dieser Form auch in [12] gegeben. Weil in dessen Formulierung das Splitting verwendet wird, wird die Definition nicht im Kapitel 1.3.3 sondern hier angegeben.

Lemma 2.1.1 *Sei $F = (V, E)$ ein gefärbter Wald mit zwei Farben blau und rot im Verhältnis 1:2, sowie ein Splitting P für F gegeben, dann kann mit $\max\left(0, \frac{x}{2}\right)$ zusätzlichen Kantenschnitten eine Partition aus dem Splitting P mit ausschließlich brr-Clustern gebildet werden, wobei $x = \#br - \#r$ und $\#br$ die Anzahl der br-Komponenten und $\#r$ die Anzahl der r-Komponenten sei. Die entstehende Partition kann in $O(n)$ berechnet werden, wobei $n = |V|$.*

Beweis: Laut Lemma 1.3.1 gibt es mindestens eine optimale Lösung, in welcher alle Cluster brr-Komponenten darstellen. Im Splitting P haben alle Cluster die Kardinalität von maximal d. Jedes Cluster mit d Knoten ist fair und stellt somit ein brr-Cluster dar. Alle anderen Cluster beinhalten weniger als 3 Knoten und können somit nur b-, br-, r- oder rr-Cluster darstellen. Es werden zuerst alle br- mit allen r-Komponenten vereint. Danach werden alle b- mit allen rr-Komponenten vereint. Danach werden alle übrigen rr-Komponenten durch den Schnitt einer Kante in zwei r-Komponenten aufgeteilt und mit den übrigen br-Komponenten vereint.

Es gibt maximal n Cluster, die maximal drei Knoten beinhalten. Die Farbe eines Clusters kann somit in konstanter Zeit $O(1)$ bestimmt werden. Die Identifikation der Cluster, die geschnitten werden müssen, oder direkt vereint werden können, liegt somit in $O(n)$. Es gibt maximal $\frac{n}{3}$ rr-Cluster, maximal $\frac{n}{3}$ br-Cluster und maximal $\frac{2 \cdot n}{3}$ r-Cluster. Es müssen somit maximal $\frac{n}{3} = O(n)$ Schnitt-Operationen ausgeführt werden und danach maximal $\frac{2 \cdot n}{3} = O(n)$ Vereinigungen, was zu $O(3n) = O(n)$ führt. □

Beispiel 2.1.2 *Im Splitting aus Abbildung 2.2 können die beiden Cluster $\{4, 10\}$ und $\{6, 11\}$ mit zwei roten Knoten zu den nicht zusammenhängenden brr-Clustern $\{4, 10, 9\}$ und $\{6, 11, 15\}$ vereint werden, dazu muss das rr-Cluster $\{9, 15\}$ getrennt werden und somit genau eine Kante. Mit $\#br = 2$ und $\#r = 1$ ergibt sich $1 = \#br - \#r$, dabei sei $\#br$ die Anzahl der br-Cluster und $\#r$ die Anzahl der r-Cluster.*

Lemma 2.1.2 *Seien die Mengen $A, B \subset \mathbb{R}$ gegeben, dann kann keine Summe $a + b$, mit $a \in A$ und $b \in B$, kleiner sein als die, die sich durch die beiden Minima beider Mengen bilden lässt, es gilt also*

$$min_{a \in A}(a) + min_{b \in B}(b) \leq min_{a \in A, b \in B}(a + b)$$

Beweis: Sei $a = \min_{e \in A}(e)$ und $b \neq \min_{e \in B}(e)$, dann ist jede Summe $a + b > a + c$, wenn $c < b$, weil

$$\begin{array}{lcl} a + b & > & a + c \quad | - a \\ b & > & c \end{array}$$

□

Lemma 2.1.3 *Seien $n \in \mathbb{N}_{\geq 2}$ viele Mengen $A_i \in \mathbb{R}^g$ mit $g \in \mathbb{N}_{>0}$ und das kartesische Produkt dieser Mengen gegeben mit $B = A_1 \times A_2 \times ... \times A_n$, sowie die Summe aller Elemente eines Tupels gegeben durch die Abbildung $f : B \to \mathbb{R}$, mit $b \in B$, definiert als $f(b) = \Sigma_{i=1}^{n} b[i]$, dann gilt für das Minimum der Summen aller Tupel*

$$min_{b \in B}(f(b)) = min_{i \in [g]}(A_1[i]) + min_{i \in [g]}(A_2[i]) + ... + min_{i \in [g]}(A_n[i])$$

, sowie

$$\forall l \in [n-1] : min_{b \in B}(f(b)) = min_{b \in B}(\Sigma_{i=1}^{l-1} b[i]) + min_{i \in [g]}(A_l[i]) + min_{b \in B}(\Sigma_{i=l+1}^{n} b[i])$$

Beweis: Der beschriebene Umstand lässt sich durch Induktion von Lemma 4 beweisen. Sei $n = 2$, dann lässt sich die Aussage

$$min_{b \in B}(f(b)) = min_{i \in g}(A_1[i]) + ... + min_{i \in g}(A_n[i])$$

auf die Aussage

$$min_{b \in B}(f(b)) = min_{i \in g}(A_1[i]) + min_{i \in g}(A_2[i])$$

reduzieren. Sei

$$c = min_{i \in g}(A_1[i]) + min_{i \in g}(A_2[i])$$

, dann lässt sich die Aussage auf $n = 3$ erweitern, mit

$$min_{b \in B}(f(b)) = c + min_{i \in g}(A_3[i])$$

und somit per Induktion auf alle $n \in \mathbb{N}_{\geq 2}$. Diese Aussage lässt sich dann über das Kommutativgesetz auf

$$min_{b \in B}(f(b)) = min_{b \in B}(\Sigma_{i=1}^{l-1} b[i]) + min_{i,j \in g}(A_l[i] + A_{l+1}[j]) + min_{b \in B}(\Sigma_{i=l+2}^{n} b[i])$$

verallgemeinern, für alle $l \in [n-1]$. □

Die folgende Definition ist auf ähnliche Weise aber nicht formal in [12] gegeben.

Definition 2.1.2 (Kopf) *Sei ein Baum $T = (V, E)$ gegeben, mit einem Knoten $r \in V$ als Wurzel und eine Partition P über T, sowie ein Teilbaum T_v und die Kante $e \in E$, mit $e = (u, v)$ und $d(u) < d(v)$, mit $u, v \in V$, wobei $v \in U(u)$, dann sei der Kopf von T_v definiert als*

$$k = \begin{cases} p \cap T_v & , \text{ wenn } \exists p \in P : e \in p \\ \emptyset & , \text{ wenn } \neg(\exists p \in P : e \in p) \end{cases}$$

, für $v = r$ sei $k = \emptyset$.

Anders ausgedrückt bezeichnet der Kopf eines Teilbaumes T_v die Schnittmenge des Teilbaums mit dem Cluster, in dem sich die Wurzel v des Teilbaums und dessen Elternknoten u gemeinsam befinden, mit $v \in U(u)$.

Zur Lösung des Fair Correlation Clusterings wird ein dynamisches Programm für Bäume angewandt, um ein optimales Splitting zu bestimmen. Wie auch beim dynamischen Programm zur Lösung des Knotenüberdeckungsproblems (siehe Kapitel 1.1.5) werden Spezialisierungen des Problems erstellt und diese für jeden Teilbaum T_v gelöst. Basierend auf den Lösungen der Kindknoten $u \in U(v)$ wird durch Minimierung über die Ergebnisse ein Ergebnis für T_v bestimmt, indem die inter-cluster Kosten summiert werden. So wie auch für das Knotenüberdeckungsproblem werden für jeden Teilbaum die Partitionen der Spezialisierungen vermerkt, sondern nur deren Werte, in diesem Fall die inter-cluster Kosten, weil nach Lemma 1.3.2 der Wert der Kostenfunktion für Wälder unabhängig von den intra-cluster Kosten ist. Nach Berechnung der inter-cluster Kosten ist eine Rekonstruktion nötig ist, um die Partition zu erhalten. Im Gegensatz zum Knotenüberdeckungsproblem werden nicht nur die Lösungen für genau zwei Spezialisierungen berechnet.

Lemma 2.1.4 *Sei ein gefärbter Wald F gegeben, mit $k = 2$ Farben und einem Farbverhältnis von 1:2, dann lässt sich ein Fair Correlation Clusterings für F in $O(n^3)$ bestimmen.*

Beweis: Sei ein gefärbter Baum $T = (V, E)$ mit zwei Farben im Verhältnis 1:2 gegeben, sowie ein beliebiger Knoten $r \in V$ als Wurzel. Für den Baum T wird mittels eines dynamischen Programms ein Splitting mit minimalen inter-cluster Kosten erstellt. Zur Lösung dieses Problems wird eine Spezialisierung definiert, die für alle Teilbäume T_v gelöst wird, mit $v \in V$. Bei der Spezialisierung wird ein Splitting mit minimalen inter-cluster Kosten gesucht, bei dem der Kopf die Färbung $h \in \{\emptyset, b, br, r, rr\}$ aufweist und die Differenz zwischen der Anzahl an br- und r-Clustern $x = \#br - \#r$ entspricht. Für jeden Teilbaum werden zu jedem Element $(h, x) \in \{\emptyset, b, br, r, rr\} \times l$, mit $l = \{x \in \mathbb{Z} | -n \leq x \leq \frac{n}{3}\}$ die minimalen inter-cluster Kosten berechnet, die ein Splitting mit diesen Anforderungen erzielen könnte. Folgende Abbildung gebe die berechneten inter-cluster Kosten für jeden Teilbaum T_v, gegeben durch $v \in V$, in Abhängigkeit der Anforderungen $x \in l$ und $h \in \{\emptyset, b, br, r, rr\}$ an.

$$\Delta : V \times \{\emptyset, b, br, r, rr\} \times l \to \mathbb{N} \tag{2.1}$$

In [12] werden die inter-cluster Kosten in einem Vektor angegeben. Diese Darstellung hat die gleiche Funktion wie die Abbildung 2.1, mit dem Unterschied, dass die Differenz $x \in l$ als Position des Elements im Vektor angegeben wird, statt als Parameter für die Abbildung. Sie wird folgend angegeben und für den einfacheren Vergleich mit [12] auch in dieser Arbeit verwendet.

$$\Delta : V \times \{\emptyset, b, br, r, rr\} \to \mathbb{N}^l$$

Da die Werte $x \in l$ auch negativ sein können, wird die Abbildung $m : l \to [|l|]$ verwendet, um die Werte aus l auf positive Werte abzubilden, sodass diese als Index zur Adressierung von Elementen des Vektors genutzt werden können. Die minimalen inter-cluster Kosten für Splittings des TeilbaumsT_v, mit h und x seien daher mit $\Delta(v, h)[m(x)]$ gegeben. Die Abbildung $m(x)$ wird folgend nicht mehr explizit angegeben. Es sei $\Delta(v, h)[x] = \Delta_v^h[x]$. Mit $S(v, h, x)$ sei ein Splitting über T_v mit gegebenen Anforderungen gegeben, welches die inter-cluster Kosten $\Delta_v^h[x]$ erzielt.

Der Kopf des Teilbaums T_v wird durch die Vereinigung der Köpfe der Teilbäume T_u, der Kinder $u \in U(v)$ mit dem Knoten v gebildet. Dabei müssen die Farben der Köpfe der Teilbäume T_u zusammen mit der Farbe des Knoten v die Farbe h ergeben. Ob gewählte Köpfe zusammen mit v ein Cluster der geforderten Farbe bilden können wird über die Summe der Vektoren der Mengenfärbung überprüft (siehe Definition 1.1.59). Alle Kombinationen an Köpfen, die mit $C(v)$ in Summe $C(h)$ erzielen seien durch die folgende Abbildung gegeben.

$$B : v \times h \mapsto \{\emptyset, b, br, r, rr\}^{|U(v)|}$$

Die Elemente dieser Menge $H \in B(v, h)$ stellen Tupel dar, mit $|U(v)|$ Färbungen, welche jedem Kind der geordneten Menge $U(v)$ jeweils eine Kopffärbung zuordnen. Die Abbildung sei wie folgt definiert.

$$B(v,h) = \begin{cases} H \in \{\emptyset, b, br, r, rr\}^{|U(v)|} & | \quad C(\{v\}) + \Sigma_{i=1}^{|U(v)|} C(H[i]) = C(h') \quad \text{, wenn } h = \emptyset \\ H \in \{\emptyset, b, r\}^{|U(v)|} & | \quad C(\{v\}) + \Sigma_{i=1}^{|U(v)|} C(H[i]) = C(h) \quad \text{, wenn } h \neq \emptyset \end{cases} \tag{2.2}$$

Dabei wird großzügig davon ausgegangen, dass sich die Abbildung der Färbungen auch auf Eingaben $h \in \{\emptyset, b, br, r, rr\}$ anwenden ließe, sodass $C(h \in \{\emptyset, b, br, r, rr\}) \in \{(0, 0), (1, 0), (1, 1), (0, 1), (0, 2)\}$.

Die Mengenfärbung $h = \emptyset$ stellt einen Sonderfall dar, weil in diesem Fall die Kante zwischen der Wurzel v und ihrem Elternknoten als getrennt interpretiert wird, wodurch der Kopf die leere Menge darstellt (siehe Definition 2.1.2). Mit $S(v, \emptyset, x)$ sei das Splitting für $\Delta_v^{\emptyset}[x]$ gegeben und demnach mit $S(v, \emptyset, x)[v]$ das Cluster, in dem sich der Knoten v befindet. Mit h' sei die Färbung des Clusters $S(v, \emptyset, x)[v]$ gegeben. Dieses Cluster kann im Gegensatz zu einem Kopf die Färbungen $h' \in \{b, br, brr, rr, r\}$ annehmen.

Für jede Färbung $h \in \{\emptyset, b, br, r, rr\}$ werden die minimalen inter-cluster Kosten für jede Differenz $x \in l$ berechnet. So wie die Abbildung B alle Kombinationen an Kopffärbungen angibt, die zur Bildung des Kopfes verwendet werden können, gibt die Abbildung A alle Differenzen für die Splittings der Kinder an, die summiert die geforderte Differenz $x = \#br - \#r$ ergeben. Die Abbildung sei definiert als

$$A : x \times y \mapsto \{M \in l^y | \Sigma_{i=1}^{y} M[i] = x\}$$

, sodass $A(x, |U(v)|)$ die Menge aller Vektoren $[l]^{|U(v)|}$ darstellt, deren Elemente summiert x ergeben.

Mit Hilfe der Abbildungen A und B lassen sich die Lösungen für $\Delta_v^h[x]$, für alle $h \in \{\emptyset, b, br, r, rr\}$ wie folgt rekursiv berechnen.

$$\Delta_v^h[x] = \min_{H \in B(v,h)} \left(\min_{M \in A(x,|U(v)|)} \left(\Sigma_{i=1}^{|U(v)|} \left(\Delta_{U(v)[i]}^{H[i]}[M[i]] \right) \right) \right) \tag{2.3}$$

Die Rekursion besteht im Zugriff auf die Ergebnisse der Kindknoten mit $\Delta_{U(v)[i]}^{H[i]}$. Die Abbildung durchläuft alle Kombinationen $H \in B(v, h)$. Die Kombination H wird verwendet, um den Köpfen der Kinder von v jeweils eine Färbung zuzuweisen. Dadurch ist gegeben, dass die Splittings über den Teilbäumen der Kinder ausgesucht werden, deren Köpfe kombiniert mit v die Färbung h ergeben. Mit fester Kombination $H \in B(v, h)$ werden die Werte aller Tupel $M \in A(x, |U(v)|)$ durchlaufen. Die inter-cluster Kosten für gegebene h und x mit $h \in \{b, br, r, rr\}$ entsprechen der Summe der Kosten für die Kinder. Aus allen Summen wird jeweils das Minimum übernommen.

Für $h = \emptyset$ wird die die Kante zwischen der Wurzel und ihrem Elternknoten als geschnitten betrachtet und trägt daher zu den inter-cluster Kosten bei. In diesem Fall wird die Summe der inter-cluster Kosten um Eins inkrementiert. Wenn dazu das Cluster $S(v, \emptyset, x)[v]$ ein br- oder r-Cluster darstellt, dann trägt dieses Cluster zur Differenz $x = \#br - \#r$ bei.

Die inter-cluster Kosten für $h' = br$ werden somit nicht an Position x im Vektor notiert, sondern an Position $x + 1$. Analoges gilt für $h' = r$. Um das in der Notation wider zu spiegeln, sei die Funktion $a : \{b, br, brr, rr, r\} \times \{\emptyset, b, br, r, rr\} \to \{-1, 0, 1\}$ gegeben, die dazu verwendet wird, den Index entsprechend zu verringern oder zu erhöhen. Sie sei definiert als

$$a(h', h) = \begin{cases} 0 & \text{, wenn} \quad h \neq \emptyset \\ 0 & \text{, wenn} \quad h = \emptyset \wedge h' \notin \{br, r\} \\ 1 & \text{, wenn} \quad h = \emptyset \wedge h' = br \\ -1 & \text{, wenn} \quad h = \emptyset \wedge h' = r \end{cases} \tag{2.4}$$

Dazu wird die Abbildung $b : h \to \mathbb{N}$ definiert, welche dazu dient in Abhängigkeit von h die inter-cluster Kosten um Eins zu erhöhen. Sie sei gegeben als

$$b(h) = \begin{cases} 0 & \text{, wenn} \quad h \neq \emptyset \\ 1 & \text{, wenn} \quad h = \emptyset \end{cases} \tag{2.5}$$

Die Abbildung 2.3 kann wie folgt abgeändert werden, um auch den Fall $h = \emptyset$ abzudecken.

$$\Delta_v^h[x - a(h', h)] = \min_{H \in B(v,\emptyset)} \left(\min_{M \in A(x,|U(v)|)} \left(\Sigma_{i=1}^{|U(v)|} \left(\Delta_{U(v)[i]}^{H[i]}[M[i]] \right) + b(h) \right) \right)$$

Zur vereinfachten Darstellung werden die Abbildungen 2.4 und 2.5 ab hier nicht mehr explizit angegeben, ähnlich wie die Abbildung $m(x)$.

Die Rekursionsanker sind durch die Blätter des Baumes gegeben. Wenn $U(v) = \emptyset$, v also ein Blatt darstellt, sei Δ_v^h wie folgt definiert.

$$\Delta_v^h = \begin{cases} \forall x \in (l \backslash \{0\}): & \Delta_v^h[x] = \infty & \Delta_v^h[0] = 0 & \text{wenn } c(v) = h = r \\ \forall x \in (l \backslash \{-1\}): & \Delta_v^h[x] = \infty & \Delta_v^h[-1] = 1 & \text{wenn } c(v) = r \wedge h = \emptyset \\ \forall x \in (l \backslash \{0\}): & \Delta_v^h[x] = \infty & \Delta_v^h[0] = 0 & \text{wenn } c(v) = h = b \\ \forall x \in (l \backslash \{0\}): & \Delta_v^h[x] = \infty & \Delta_v^h[0] = 1 & \text{wenn } c(v) = b \wedge h = \emptyset \end{cases} \tag{2.6}$$

, dabei gebe $\Delta_v^h[x] = \infty$ an, dass keine Partition über T_v möglich ist, welche die durch h und x gegebenen Anforderungen erfüllt.

In gegebener Form liegt die Kardinalität der Menge $A(x, y)$, mit $y = O(n)$ bei $|A(x, y)| = O(l^n)$. Die Elemente dieser Menge können demnach zur Berechnung der Abbildung 2.3 nicht in polynomieller Zeit durchlaufen werden. Laut Lemma 2.1.3 kann die Anzahl der Kombinationen beschränkt werden, indem die Lösungen rekursiv für alle Teilbäume T_v^i bestimmt werden, mit $2 \leq i \leq |U(v)|$. Dabei werden immer nur zwei Vektoren in jedem Schritt betrachtet.

T_v^i sei der Teilbaum, der sich mit v und den Teilbäumen seiner ersten i Kinder bilden lässt, sodass

$$T_v^i = T\left[\{v\} \cup \left(\bigcup_{j=1}^{i} T_{U(v)[j]}\right)\right]$$

, wobei $T[S \subseteq V]$ den induzierten Graphen angibt. Es gilt $T_v^i \subseteq T_v$ und $T_v^{|U(v)|} = T_v$. Die Lösung für den Teilbaum T_v^i lässt sich aus den Lösungen der beiden Teilbäume T_v^{i-1} und $T_{U(v)[i]}$ bestimmen. Für den Teilbaum T_v^i seien die minimalen inter-cluster Kosten mit $\Delta_v^h(i)$ gegeben. Es gelte $\Delta_v^h(|U(v)|) = \Delta_v^h$, sodass das Ergebnis der Abbildung 2.3 alternativ durch folgende rekursive Abbildung berechnet werden kann

$$\Delta_v^h(i)[x] = \min_{H \in B(v,h)} \left(\min_{M \in A(x,2)} \left(\Delta_v^{H[i-1]}(i-1)[M[1]] + \Delta_u^{H[i]}[M[2]] \right) \right) \tag{2.7}$$

, für alle $1 \leq i \leq |U(v)|$, mit $u = U(v)[i]$.

Der Fall $i = 0$ stellt den Rekursionsanker dar. Auch in diesem Fall stellen Blattknoten die Rekursionsanker dar. Die inter-cluster Kosten $\Delta_v^{h \in \{b,br,r,rr\}}(0)$ werden daher mit der Abbildung 2.6 initialisiert. Das bedeutet, dass auch für Knoten v mit $|U(v)| > 0$ der Ausgangspunkt der eines Blatt ist. Damit kommt ein zusätzlicher Rekursionsschritt pro Knoten hinzu, der aber keinen Einfluss auf die Komplexität hat, da er von anderen Aspekten überschattet wird. Zudem wird er auch durch den entstehenden Zugewinn durch die Reduktion der Mengen 2.8 und 2.9 überschattet.

Durch die rekursive Verarbeitung von T_v^i mit $T_{U(v)[i+1]}$ wird die Menge an Kombinationen in $A(x, |U(v)|)$ auf zwei Elemente begrenzt, sodass stattdessen

$$A(x, 2) = A(x) = \{(a_1, a_2) \in l^2 | a_1 + a_2 = x\} \tag{2.8}$$

verwendet werden kann. Mit $|U(v)| = O(n)$ und $l = O(n)$ senkt das die Anzahl an Kombinationen von $|A(x, |U(v)|)| \leq l^{|U(v)|}$ auf $|A(x)| = O(n)$. Analoges gilt für die Menge $B(v, h)$, für diese gilt nach Definition der Abbildung 2.2 $|B(v, h)| = O(n^2)$, weil das größte Cluster, in dem v enthalten sein kann, mit der Kopffärbung $h = \emptyset$ mit der Mengenfärbung $h' = brr$ ist. Es können demnach nur zwei weitere Kindknoten in dem gleichen Cluster enthalten sein, sodass für jede Kombination $H \in B(v, \emptyset)$ höchstens ein Wertepaar $(j, k) \in \mathbb{N}_{>0}^2$ existiert, sodass mit $j \neq k$ gilt $H[j] \neq \emptyset \wedge H[k] \neq \emptyset$. Für die Abbildung 2.7

kann statt der Menge aus Abbildung 2.2 folgende Definition für $B(v,h)$ verwendet werden.

$$B(v,h) = \begin{cases} (h_1,h_2) \in \{\emptyset, b, br, r, rr\}^2 & | \quad C(v)+C(h_1)+C(h_2) = C(h') \quad \text{, wenn } h = \emptyset \\ (h_1,h_2) \in \{\emptyset, b, r\}^2 & | \quad C(v)+C(h_1)+C(h_2) = C(h) \quad \text{, wenn } h \neq \emptyset \end{cases} \tag{2.9}$$

Damit reduziert sich die Anzahl der Kombinationen auf $|B(h)| \leq 14 = O(1)$. Die möglichen Wertemengen sind folgend gelistet.

$$\begin{aligned} B(v,\emptyset) &= \{(\emptyset,r),(r,\emptyset),(rr,\emptyset),(\emptyset,rr),(r,r),(\emptyset,\emptyset)\} \\ B(v,b) &= \{(\emptyset,\emptyset)\} \\ B(v,br) &= \{(r,\emptyset),(\emptyset,r)\} \qquad \text{mit c(v)=b} \\ B(v,r) &= \{\} \\ B(v,rr) &= \{\} \end{aligned}$$

$$\begin{aligned} B(v,\emptyset) &= \{(b,r),(r,b),(br,\emptyset),(\emptyset,br),(b,\emptyset),(\emptyset,b),(r,\emptyset),(\emptyset,r),(\emptyset,\emptyset)\} \\ B(v,b) &= \{\} \\ B(v,br) &= \{(b,\emptyset),(\emptyset,b)\} \qquad \text{mit c(v)=r} \\ B(v,r) &= \{(\emptyset,\emptyset)\} \\ B(v,rr) &= \{(r,\emptyset),(\emptyset,r)\} \end{aligned}$$

Mit den beiden Abbildungen 2.8 und 2.9 kann eine vereinfachte Notation zu Abbildung 2.7 wie folgt definiert werden.

$$\Delta_v^h(i)[x] = \min_{(h_1,h_2)\in B(v,h)} \left(\min_{(a_1,a_2)\in A(x)} \left(\Delta_v^{h_1}(i-1)[a_1] + \Delta_u^{h_2}[a_2] \right) \right) \tag{2.10}$$

In der Abbildung 2.10 werden in der Minimierung im Innern der Klammern $|A(x)| = O(n)$ Durchläufe berechnet. Die Äußere Klammer zur Minimierung über alle $H \in B(v,h)$ wird $|B(v,h)| = O(1)$ oft durchlaufen. Die Berechnung der minimalen inter-cluster Kosten für eine bestimmte Differenz x hat eine Laufzeit von $O(n)$. Mit $x \in l$ und $|l| = O(n)$ liegt die Laufzeit zur Bestimmung aller Elemente im Vektor $\Delta_v^h(i)$ damit bei $O(n^2)$. Die Abbildung 2.10 muss für alle $O(n)$ Knoten und alle $|U(v)| = O(n)$ Kindknoten ausgeführt werden, was in $O(n^2)$ liegt. Jedoch existiert für jedes dieser Knotenpaare (u,v), mit $u \in U(v)$ genau eine Kante $e = (u,v) \in E$, was mit $|E| \leq n-1$ zu einer Laufzeit von insgesamt $O(n^3)$ führt.

Zur Rekonstruktion der Partition als Lösung des Fair Correlation Clusteringproblems werden bei der Ausführung des dynamischen Programms für jeden Wert $\Delta_v^h[x]$ die Kombination H und die Kombination an Indizes M gespeichert, die zur Herleitung des minimalen Werts verwendet wurden. Diese Informationen können im Vektor gespeichert werden, also $\Delta_v^h[x] = (\chi,\ M,\ H)$ oder alternativ auf den Kanten zwischen Wurzel und Kind. Nach Berechnung von $\Delta_r^\emptyset$ kann das Minimum $min_{x\in l}\left(\Delta_r^\emptyset[x] + max(0,\ \frac{x}{2})\right)$ berechnet werden. Wenn zu jedem Wert $\Delta_v^h[x]$ notiert wird, mit welchen Elementen H und M diese erzielt wurden, kann rekursiv die Partition als Lösung der ersten Phase hergeleitet werden.

Nach der Rekonstruktion wird im zweiten Schritt aus allen Clustern, die nicht brr-Komponenten darstellen durch den in Lemma 2.1.1 beschriebenen Vorgang brr-Komponenten erstellt, was eine Laufzeit von $O(n)$ aufweist. Die Laufzeit der Rekonstruktion und der zweiten Phase werden von der Laufzeit zur Berechnung der Vektoren überschattet.

Der Algorithmus lässt sich auf Wälder verallgemeinern. Seien die Wurzeln der einzelnen Bäume mit r_i gegeben, dann werde auf jedem einzelnen Baum der Vektor $\Delta^h_{r_i}$ berechnet und danach über alle Vektoren eine zusätzliche Minimierung für $\Delta^\emptyset_{r_i}$ ausgeführt. Dieser zusätzliche Schritt hat keinen Einfluss auf die asymptotische Laufzeit des Algorithmus, weil $O(n^3 + n^2) = O(n^3)$. □

In folgendem Beispiel wird anhand des Eingabegraphen aus Darstellung 2.1 gezeigt, wie der Algorithmus zum vorhergehenden Beweis zu Lemma 2.1.4 funktioniert.

Beispiel 2.1.3 (Dynamisches Programm zur Berechnung eines optimalen Splittings)
Als erstes werden die Blätter betrachtet, weil sie die Rekursionsanker darstellen und der Rest des Algorithmus somit auf deren Lösungen aufbaut.

In der gegebenen Eingabe aus Bild 2.1 sind die Blätter, also alle Knoten $v \in V$ mit $U(v) = \emptyset$, durch die Knoten $5, 10, 11, 12, 13, 14$ und 15 gegeben. Deren Vektoren Δ^h_v können demnach mit den Werten aus Abbildung 2.6 initialisiert werden. Sei mit $V[i]$ der i-te Knoten entsprechend der vorgegebenen Nummerierung gegeben. Die Vektoren für die Blattknoten $\Delta^h_{V[10]}$ und $\Delta^h_{V[11]}$ gleichen sich und haben die Form

$$\Delta^{h=\emptyset}_{V[10]} = \begin{pmatrix} \infty \\ ... \\ 1 \\ ... \\ \infty \end{pmatrix} \quad \Delta^{h=b}_{V[10]} = \begin{pmatrix} \infty \\ ... \\ 0 \\ ... \\ \infty \end{pmatrix} \quad \Delta^{h=br}_{V[10]} = \Delta^{h=r}_{V[11]} = \Delta^{h=rr}_{V[11]} \begin{pmatrix} \infty \\ ... \\ \infty \end{pmatrix}$$

, wobei sich nur die Werte $\Delta^\emptyset_{V[10]}[0]$ und $\Delta^b_{V[10]}[0]$ von ∞ unterscheiden. Es sei daran erinnert, dass die Abbildung $m(x)$ in $\Delta^h_v[m(x)]$ nicht explizit angegeben wird. In der folgenden Darstellung wird das Splitting $S(V[11], \emptyset, 0)$ mit inter-cluster Kosten von $\Delta^\emptyset_{V[11]}[0] = 1$ auf der linken Seite und das Splitting $S(V[11], b, 0)$ mit inter-cluster Kosten von $\Delta^b_{V[11]}[0] = 0$ auf der rechten Seite dargestellt. Für $h = \emptyset$ ist das einzige Cluster im Splitting $S(V[11], \emptyset, 0)$ das Cluster, in dem sich der Wurzelknoten $V[11]$ befindet, welches mit $S(V[11], \emptyset, 0)[V[11]]$ gegeben sei. Dieses Cluster wird als abgeschlossen betrachtet, was bedeutet, dass dieses Cluster nicht mit dem Knoten $V[6]$ vereint werden kann. Dadurch wird die Kante $(V[6], V[11])$ durch die Grenze des Clusters geschnitten, was zu inter-cluster Kosten von 1 führt. Weil kein br- oder r-Cluster, sondern nur ein b-Cluster im Splitting enthalten sind, ergibt sich die Differenz $x = \#br - \#r = 0$. Zusammen führt das zu $\Delta^\emptyset_{V[11]}[0] = 1$.

Das Cluster $S(V[11], b, 0)[V[11]]$ ist dagegen so definiert, dass es den Knoten $V[6]$ enthält. Dies wird in der Darstellung 2.3 durch Verblassen der Clustergrenzen nach oben angedeutet. In diesem Fall bildet der Kopf aber ebenfalls kein br- oder r-Cluster, weil der Kopf als die Schnittmenge des Clusters mit dem betrachteten Teilbaum definiert ist. Es wird auch kein b-Cluster gebildet, weil das Cluster mindestens den Knoten $V[6]$ enthalten muss. Dieses Vorgehen wird folgend noch in Darstellung 2.5 anhand von Clustern verdeutlicht, die mit $h = \emptyset$ in die Differenz $x = \#br - \#r$ einbezogen würden, aber mit $h = br$ nicht. Alle anderen Splittings über dem Teilbaum sind nicht möglich und erzeugen daher inter-cluster Kosten von ∞.

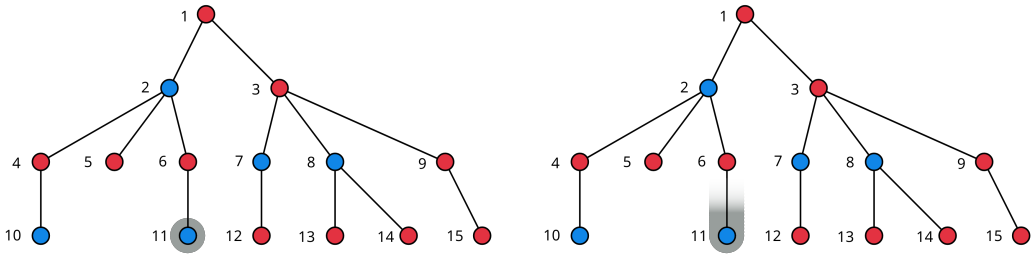

Abbildung 2.3: Köpfe der blauen Blätter

Analog ergeben sich die Vektoren für die Knoten $5, 12, 13, 14$ *und* 15 *wie folgt.*

$$\Delta^{h=\emptyset}_{V[12]} = \begin{pmatrix} \infty \\ \dots \\ 1 \\ \dots \\ \infty \end{pmatrix} \quad \Delta^{h=r}_{V[12]} = \begin{pmatrix} \infty \\ \dots \\ 0 \\ \dots \\ \infty \end{pmatrix} \quad \Delta^{h=b}_{V[11]} = \Delta^{h=br}_{V[12]} = \Delta^{h=rr}_{V[12]} = \begin{pmatrix} \infty \\ \dots \\ \infty \end{pmatrix}$$

, wobei sich nur die Werte $\Delta^{\emptyset}_{V[12]}[-1]$ *und* $\Delta^{r}_{V[12]}[0]$ *von* ∞ *unterscheiden. Analog zur Darstellung 2.3 werden folgend in Darstellung 2.4 die beiden möglichen Splittings über dem Teilbaum* T^h_v*, mit* $v = V[13]$ *und* $h \in \{\emptyset, r\}$ *angegeben.*

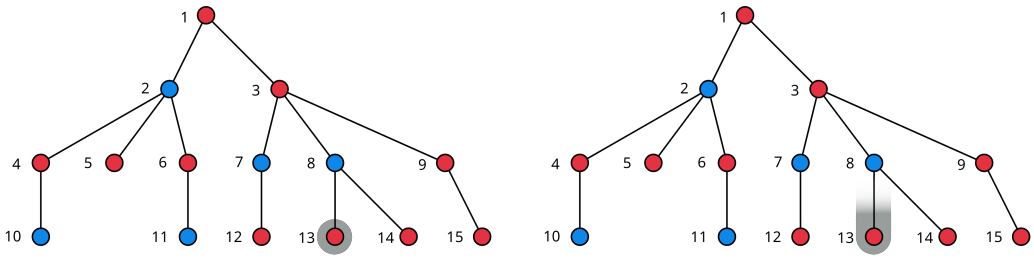

Abbildung 2.4: Köpfe der roten Blätter

In Darstellung 2.5 wird der Teilbaum $T_{V[3]}$ *angegeben, sowie alle Splittings, die über dem Teilbaum* T_v*, mit* $v = V[8]$ *möglich sind, mit* $h \in \{b, br\}$ *und* $x \in \{-2, -1\}$*. Alle nicht angegebenen Splittings ergeben inter-cluster Kosten des Werts* ∞ *und sind somit über* T_v *nicht erstellbar. Das sind bspw. alle Köpfe ohne blauen Knoten, also solche mit* $h \in \{r, rr\}$*, da mit* $C(\{v\}) = b$ *keine solche Kopffärbung möglich ist. Die Differenz* x *ergibt sich für Kopffärbungen* $h \in \{b, br, r, rr\}$ *ausschließlich durch die Summen aus den Differenzen der verwendeten Ergebnisse der Kindknoten. Nur für* $h = \emptyset$ *kann sich die Differenz um maximal Eins von der Summe unterscheiden, wie in Darstellung 2.6 noch gezeigt wird. In den* Δ*-Vektoren der Kindknoten gibt es nur zwei Elemente, die sich von* ∞ *unterscheiden. Diese sind an den Positionen* 0 *und* -1*, sodass* $\Delta^h_u[0] \neq \infty$ *und* $\Delta^h_u[-1] \neq \infty$*. Die Minimierung über allen Kombinationen in* $A(x, |U(V[8])|$*, mit*

$$\Delta^h_{V[8]}(i)[x] = \min_{H \in B(V[8],h)} \left(\min_{M \in A(x,2)} \left(\Delta^{H[i-1]}_v (i-1)[M[1]] + \Delta^{H[i]}_u [M[2]] \right) \right)$$

ergibt demnach nur für die Elemente der Positionen $x \in \{-2, -1\}$ *inter-cluster Kosten* $\Delta_{V[8]}^{h \in \{b,br\}}[x] \neq \infty$.

Die beiden Splittings auf der rechten Seite der Darstellung 2.5 erzeugen eine Differenz von $x = -1$*, weil jeweils ein r-Cluster gebildet wird, nämlich die Cluster* $\{V[13]\}$ *und* $\{V[14]\}$*. Unter den beiden Clustern wird in* Δ_v^{br} *nicht unterschieden, da sie in diesem Fall beide die vorgegebenen Anforderungen* $h = br$ *und* $x = -1$ *erfüllen und die inter-cluster Kosten sich gleichen, mit* $\Delta_v^{br}[-1] = 1$*. Der Kopf wird bei allen Splittings mit* $h \neq \emptyset$ *nicht in die Differenz einbezogen, weil in diesem Fall bekannt ist, dass das Cluster durch den Knoten* $u = V[3]$ *erweitert wird. Ansonsten würde sich eine Differenz von* $x = 0$ *ergeben.*

Das Splitting auf der linken Seite zeigt den Fall $h = b$*, mit* $x = -2$*. Es werden genau zwei r-Cluster erstellt und die Kante* $(V[3], V[8])$ *wird nicht geschnitten, dafür aber die Kanten* $(V[8], V[13])$ *und* $(V[8], V[14]$*. Somit ergeben sich inter-cluster Kosten von* $\Delta_v^b[-2] = 2$*.*

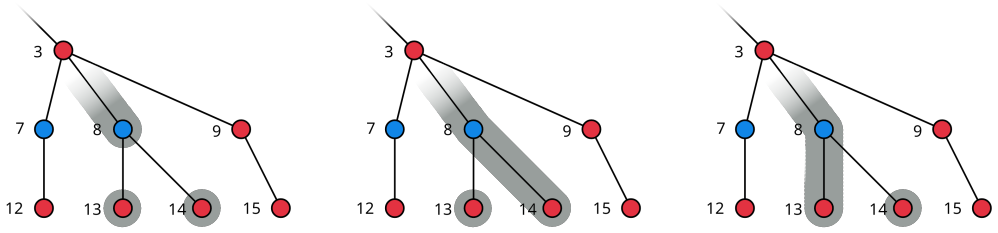

Abbildung 2.5: Offene Köpfe des Teilbaums $T_{V[8]}$

In Darstellung 2.6 wird der Teilbaum $T_{V[3]}$ *angezeigt, sowie die Splittings über dem Teilbaum* T_v*, mit* $v = V[8]$*, mit Kopffärbung* $h = \emptyset$*. Mit* $h = \emptyset$ *wird in jedem Splitting mindestens eine Kante geschnitten, die Kante* $(V[3], V[8])$*. Analog zur Darstellung 2.5 werden nur die Splittings angegeben, die realisierbar sind, also inter-cluster Kosten aufweisen, die sich von* ∞ *unterscheiden.*

Die Darstellung ganz links entspricht dem Splitting $S(v, \emptyset, -2)$ *und weist inter-cluster Kosten von* $\Delta_v^{\emptyset}[-2] = 3$ *auf. Die Differenz ergibt sich daraus, dass* $V[13]$ *und* $V[14]$ *jeweils alleine in einem Cluster liegen und somit* $\#br = 0$ *und* $\#r = 2$*. Die inter-cluster Kosten ergeben sich daraus, dass zusätzlich zu* $(V[3], V[8])$ *die Kanten* $(V[8], V[13])$ *und* $(V[8], V[14]$ *geschnitten werden.*

Die beiden Darstellungen in der Mitte entsprechen den Splittings $S(v, \emptyset, 0)$*. Durch das Schneiden der Kante* $(V[3], V[8])$ *wird das br-Cluster als abgeschlossen betrachtet und in die Differenz* $x = \#br - \#r$ *einbezogen. Wie für* $\Delta_v^{h \in \{b,br\}}[-1]$ *wird auch hier nicht zwischen den beiden Splittings unterschieden, sodass bei der Rekonstruktion des Ergebnisses zufällig eine von beiden gewählt werden muss. In der Darstellung auf der rechten Seite ist das Splitting mit* $h' = brr$ *gegeben. Dies ist die einzige Möglichkeit eine Differenz von* $x = \#br - \#r = 0$ *zu erzielen. Dazu werden die Splittings* $S(V[13], r, 0)$ *und* $S(V[14], r, 0)$ *verwendet. Mit* $\Delta_{V[13]}^r[0] = \Delta_{V[14]}^r[0] = 0$ *ergeben sich durch die Trennung der Kante* $(V[3], V[8])$*) inter-cluster Kosten von* $\Delta_v^{\emptyset}[0] = 1$*.*

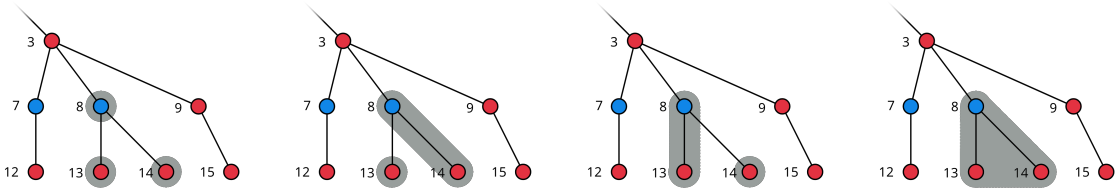

Abbildung 2.6: Abgeschlossene Köpfe des Teilbaums $T_{V[8]}$

Buchin et. al. [10] nutzen folgende Darstellung zur Verdeutlichung des Teilbaums T_v^i. Hierbei wird die Wurzel mit v bezeichnet und die Kinder von v mit $v_j \in U(v)$, wobei $1 \leq j \leq |U(v)|$. Die in T_v^i enthaltenen Teilbäume T_{v_j} sind umrandet. Die Teilbäume der Kindknoten T_{v_j} werden als Dreiecke angedeutet. Die Teilbäume $1 \leq j \leq i$ sind in T_v^i enthalten die Teilbäume mit $i < j \leq k$ sind nicht enthalten.

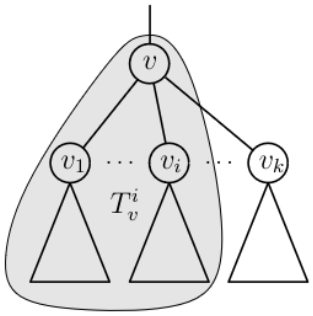

Abbildung 2.7: Schematischer Teilbaum

Die hergeleitete Laufzeit weicht von der in [12] gegebenen ab, allerdings wird auch darauf hingewiesen, dass für den polynomiellen Algorithmus keine genaue Eingrenzung der Komplexität vorgenommen wurde, weil alle polynomiellen Laufzeiten als effizient gelten (siehe Kapitel 1.1.3) und eine genauere Grenze für die Anwendung auf Bäumen nicht nötig ist.

Inhaltsverzeichnis

2.1.2 Wälder mit beliebig vielen Farben in beliebigem Verhältnis

Der zweite Algorithmus, der in [12] vorgestellt wird, behandelt Bäume mit beliebig vielen Farben in beliebigem Verhältnis zueinander. Der Algorithmus stellt eine Generalisierung des Algorithmus aus Kapitel 2.1.1 dar. Er wird folgend zusammengefasst und auf Basis der genaueren Laufzeit des vorherigen Kapitels, wird auch für diesen eine genauere Komplexitätsgrenze hergeleitet. Der Algorithmus basiert stark auf dem aus dem Kapitel 2.1.1 und wird daher in Anlehnung an diesen erklärt, um Wiederholungen zu vermeiden.

Definition 2.1.3 (Färbung einer Partition) *Sei V eine Menge gefärbter Elemente, mit n Elementen und k Farben, sei P eine Partition über V und $\mathcal{A}$ die geordnete Menge aller möglichen Mengenfärbungen $\mathcal{A} = \{C(D)|D \subseteq V\}$, dann ist die Färbung von P gegeben durch*

$$C_p(P)[i] = |\{p \in P|C(p) = \mathcal{A}[i]\}|$$

Die Definition der Färbung von Partitionen sowie das folgende Lemma und dessen Beweis sind in dieser Form auch in [12] gegeben. Beim Vergleich sei zu beachten, dass für die Partitionsfärbung in [12] ein leicht unterschiedlicher Bezeichner gewählt wurde.

Lemma 2.1.5 *Sei U eine Menge gefärbter Elemente, mit n Elementen und $k \in \mathbb{N}_{\geq 2}$ Farben und seien die Werte $d_1, d_2, ..., d_k \in ([n] \cup 0)$ gegeben. Sei S die Menge aller möglichen Partitionen der Menge U, für die gilt, dass für jede Farbe i höchstens d_i Elemente dieser Farbe in jedem Cluster enthalten sind, sodass für alle Cluster c jeder Partition s in S gelte*

$$\forall s \in S : \forall c \in s : \forall i \in [k] : C(c)[i] \leq d_i$$

Sei $\mathcal{C}$ die geordnete Menge aller Färbungen der Partitionen in S, also $\mathcal{C} = \{s \in S|C_p(s)\}$, dann gilt $|\mathcal{C}| \leq (n+1)^{setvars-1}$, wobei $setvars = \Pi_{i=1}^{k}(d_i + 1)$.

Beweis: Die Färbung einer Teilmenge $u \subseteq U$ ist durch einen Vektor mit k Elementen gegeben, weil für jede der k Farben die Anzahl der Knoten in u angegeben wird. Die Werte jedes Elements werden durch d_i begrenzt, sodass eine Mengenfärbung, statt $([n] \cup \{0\})^k$, nur *setvars* verschiedene Werte annehmen kann. Die Färbung einer Partition über U ist demnach ein Vektor mit setvars vielen Elementen. Jedes Element des Vektors kann einen Wert der Menge $\{[n] \cup \{0\}\}$ annehmen, weil maximal n Cluster mit der gleichen Mengenfärbung in der Partition enthalten sein können. Demnach gibt es höchstens $|([n] \cup \{0\})|^{setvars}$ viele verschiedene Partitionsfärbungen. Da die Färbung des letzten Clusters von den Farben und der Anzahl an übrigen Knoten abhängt, gilt $|\mathcal{C}| \leq (n+1)^{setvars-1}$. □

Der Algorithmus zur Lösung des Fair Correlation Clusterings mit beliebig vielen Farben in beliebigem Verhältnis unterscheidet sich von dem für zwei Farben im Verhältnis 2:1 dadurch, dass die Kopffärbung jede mögliche Mengenfärbung annehmen kann, die unter der Vorgabe der Fairness nach Lemma 1.3.1 möglich ist. Zudem entfällt die Angabe der Differenz x komplett. Stattdessen werden die Positionen der Elemente in den Vektoren Δ_v^h dazu verwendet, die Partitionsfärbung über dem Teilbaum anzugeben. Da eine Partitionsfärbung einen Vektor darstellt, wird dessen Position in der geordneten Menge $\mathcal{C}$ als Index verwendet.

Lemma 2.1.6 *Das Fair Correlation Clustering lässt sich auf Wäldern $F = (V, E)$ mit $k \in \mathbf{N}_{\geq 2}$ Farben und den Verhältnissen $c_i \in [n]$, mit $ggt(c_1, c_2, ..., c_k) = 1$ in einer Laufzeit von $O((n+1)^{5setvars-5} \cdot n^2)$ lösen, mit $d = \Sigma_{i=1}^{k} c_i$.*

Beweis: Laut Lemma 1.3.1 kann ein Cluster eines Splittings maximal $d = \Pi_{i=1}^{k} c_i$ verschiedene Färbungen annehmen. Laut Lemma 2.1.5 gibt es demnach $|\mathcal{C}| \leq (n+1)^{setvars-1}$ verschiedene Partitionsfärbungen über $|V|$, mit $setvars = \Pi_{i=1}^{k}(c_i + 1)$, wobei $\mathcal{C}$ die geordnete Menge aller Partitionsfärbungen sei. Für jeden Teilbaum T_v wird das Minimum der inter-cluster Kosten aller Splittings bestimmt, welche die vorgegebene Kopffärbung und die vorgegebene Partitionsfärbung erzielen. Die Menge der Kopffärbungen sei mit $L = \{h \in \mathbb{N}^k | \forall i \in [k] : h[i] \leq c_i\}$ gegeben. Die Menge L beinhaltet alle Mengenfärbungen, die höchstens d viele Knoten beinhalten und mit maximaler Kardinalität eine faire Teilmenge implizieren. Die inter-cluster Kosten werden - wie auch im vorherigen Kapitel - in einem Vektor gegeben. Die Abbildung

$$\Delta : V \times L \to ([n] \cup \{0\})^{|\mathcal{C}|}$$

gibt für gegebenen Teilbaum T_v und Kopffärbung die minimalen inter-cluster Kosten aller Splittings für alle möglichen Partitionsfärbungen und ordnet diese entsprechend der Ordnung aus $\mathcal{C}$ an.

Die Abbildung $m : \mathcal{C} \to [|\mathcal{C}|]$ bilde die Vektoren der Partitionsfärbungen auf positive Werte ab, sodass die Partitionsfärbungen zur Adressierung der Elemente im Vektor Δ_v^h verwendet werden können.

Es gilt für alle $h \in L$, $C \in \mathcal{C}$, alle $v \in V$ und $2 \leq i \leq |U(v)|$

$$\Delta_v^h(i)[m(C - a(h))] = \min_{(h_1,h_2) \in B(v,h)} \left(\min_{(c_1,c_2) \in \mathcal{C}^2} \left(\Delta_v^{h_1}(i-1)[c_1] + \Delta_{U(v)[i]}^{h_2}[c_2] \right) + b(h) \right)$$

Es sei $h = C_\emptyset$ das Pendant zu $h = \emptyset$ aus dem Beweis zu Lemma 2.1.4. Das bedeutet, dass in Splittings mit der Kopffärbung $C_\emptyset$ die Kante über der Wurzel als geschnitten betrachtet wird. Das zieht wiederum nach sich, dass Köpfe mit Färbung $h \neq C_\emptyset$ keinen Einfluss auf die Partitionsfärbung $C \in \mathcal{C}$ haben, weil das Cluster über den betrachteten Teilbaum hinausragt und die Knoten des Clusters, die sich außerhalb des Teilbaums befinden noch nicht festgelegt sind.

Köpfe mit Färbung $h \neq C_\emptyset$ haben dagegen einen Einfluss auf die Partitionsfärbung. Der Einfluss besteht darin, dass die Anzahl der Cluster mit der Mengenfärbung h' (definiert wie im Beweis zu Lemma 2.1.4) um Eins inkrementiert wird. Die Köpfe der Kindknoten, die dazu mit dem Wurzelknoten v vereint werden, weisen die Mengenfärbung $C_\emptyset$ auf. Sie haben somit keinen Einfluss auf die Partitionsfärbung und müssen daher nicht subtrahiert werden.

Analog zur Abbildung 2.4 sei die Abbildung $a : L \cup C_\emptyset \to \mathbb{N}^{|\mathcal{C}|}$ gegeben, die zur relativen Veränderung des Index verwendet wird, wenn ein neues Cluster in die Partitionsfärbung des Splittings aufgenommen wird.

$$a(h) = \begin{cases} e \in \mathbb{N}^{|\mathcal{C}|} \quad | \quad \forall i \in [|e|] : e[i] = 0 & \text{, wenn } h \neq C_\emptyset \\ e \in \mathbb{N}^{|\mathcal{C}|} \quad | \quad \forall i \in [|e|] : \mathcal{C}[i] \neq C(h) \Rightarrow e[i] = 0 \wedge \mathcal{C}[i] = C(h) \Rightarrow e[i] = 1 & \text{, wenn } h = C_\emptyset \end{cases} \tag{2.11}$$

Die Abbildung $b : L \cup C_\emptyset \to \{0, 1\}$ sei definiert als

$$b(h) = \begin{cases} 1 & \text{, wenn } h = C_\emptyset \\ 0 & \text{, wenn } h \neq C_\emptyset \end{cases} \tag{2.12}$$

Wie auch Abbildung 2.4 werden die Abbildungen 2.11 und 2.12 folgend nicht mehr explizit angegeben.

Die Werte für $\Delta_v^h(0)$ werden als Blätter initialisiert.

Die Werte Δ_v^h werden für Blätter, also Knoten mit $|U(v)| = 0$, wie folgt initialisiert.

$$\Delta_v^h = \begin{cases} \forall C \in \mathcal{C} \backslash \{C_p(\{v\})\} : \Delta_v^h[C] = \infty & \\ \Delta_v^h[C_p(\{v\})] = 0 & \text{, wenn } h = C_p(\{v\}) \\ \Delta_v^h[C_p(\{v\})] = 1 & \text{, wenn } h = C_\emptyset \end{cases}$$

Die Kombinationen der Köpfe, die vereint mit dem Wurzelknoten v ein Cluster der Färbung h ergeben, sei durch die folgende Abbildung gegeben

$$B : v \times h \mapsto \{H \in \mathcal{C}^{|U(v)|} | C(\{v\}) + \Sigma_{i=1}^{|U(v)|} H[i] = h\}$$

, sodass jedes Element $H \in B(v,h)$ ein Tupel darstellt, das dem Kopf jedes Kindknoten $u \in U(v)$ eine Mengenfärbungen zuordnet. Es gelte

$$|B(v,h)| \leq |\mathcal{C}|^{d-1} \leq (n+1)^{setvars-1^{d-1}}$$

, weil höchstens $d-1$ Knoten aus den $O(n)$ vielen Kindern entnommen werden müssen. Für die schrittweise Berechnung von $\Delta_v^h(i)$ für den Teilbaum T_v^i unter Verarbeitung von T_v^{i-1} mit $T_{U(v)[i]}$ wird $B(v,h)$ wie folgt angepasst werden.

$$B(v,h) = \{(h_1,h_2) \in \mathcal{C}^2 | C(\{v\}) + h_1 + h_2 = h\} \tag{2.13}$$

Für diese Menge liegt die Anzahl an Kombinationen bei

$$|B(v,h)| \leq |\mathcal{C}^2| \leq \left((n+1)^{setvars-1}\right)^2 = (n+1)^{2\cdot(setvars-1)}$$

Hier wird im Gegensatz zu vorigem Kapitel nicht die Abbildung $C(H[i])$ verwendet, weil sowohl h, als auch die Elemente des Tupels H bereits als Färbungsvektoren definiert sind.

Wie auch im Beweis zu Lemma 2.1.4 entsteht auch hier der Zugewinn bezüglich der Komplexität durch die rekursive Behandlung der Teilbäume T_v^{i-1} mit $T_{U(v)}$ Die Komplexität des Algorithmus ergibt sich wie folgt. Es sei $|V| = n$. Bei der Abbildung $\Delta_v^h(i)$ werden alle Elemente $(h_1,h_2) \in B(v,h)$ durchlaufen, sowie alle Elemente $(c_1,c_2) \in \mathcal{C}^2$ mit $|B(v,h)| \leq |\mathcal{C}^2|$ und $|\mathcal{C}| = O((n+1)^{setvars-1})$. Das bedeutet, dass die Berechnung von $\Delta_v^h(i)$ in $O((n+1)^{4setvars-4})$ liegt. Die Abbildung wird für alle $h \in \mathcal{C}$ bestimmt und für alle $O(n)$ Kanten in E. Die Abbildung $m(C)$ hat eine Laufzeit von $O(n)$ und muss in jedem Minimierungsschritt zweimal angewendet werden. Somit ergibt sich eine Laufzeit von

$$O((n+1)^{4setvars-4} \cdot (n+1)^{setvars-1} \cdot n \cdot n) = O((n+1)^{5setvars-5} \cdot n^2)$$

In der zweiten Phase muss, im Gegensatz zur Probleminstanz mit zwei Farben im Verhältnis 2:1, keine weitere Kante getrennt werden, wenn der Vektor Δ_r^h korrekt bestimmt wurde. Sei angenommen es gäbe ein Splitting S mit Färbung C_S, das durch zusätzliches Trennen von a Kanten ein Splitting S' mit Farbe $C_{S'}$ hervorruft, sodass die inter-cluster Kosten von S' geringer seien als die von S. In diesem Fall muss es einen Eintrag $\Delta_r^h[C_{S'}] \leq \Delta_r^h[C_S] + a$ geben, sodass S' in der Lösung bereits berücksichtigt werden würde und beim Vergleich gegenüber S präferiert werden müsste. Die nachträglichen Schnitte wären damit überflüssig.

In der zweiten Phase bleibt daher nur, die Cluster, die keine faire Teilmenge darstellen, zu fairen Clustern zu vereinen. Die Correlation Clustering Kosten sind nach Lemma 1.3.2 unabhängig von intracluster Kosten. Durch Vereinigung zweier Cluster C_1, C_2 eines Splittings zu einem Cluster $C' = C_1 \cup C_2$ können die inter-cluster Kosten der so entstehenden Partition nicht höher sein als die des Splittings. Die inter-cluster Kosten können lediglich sinken. Das ist aber nur dann der Fall, wenn die beiden Cluster C_1, C_2 über eine Kante miteinander verbunden sind, in diesem Fall würde die entstehende Partition einem Splitting gleichen, welches demnach in der ersten Phase berücksichtigt wurde. Damit müsste das Splitting in Δ_r^h enthalten sein und beim Vergleich der Kosten präferiert werden.

In der zweiten Phase wird daher das Splitting mit den geringsten inter-cluster Kosten nach Δ ausgewählt. Alle Cluster mit einer Kardinalität von weniger als d werden mit beliebigen anderen zu fairen Clustern der Größe d vereint. Wenn nur Splittings mit inter-cluster Kosten von ∞ vorhanden sind, gibt es keine Lösung für gegebenen Graphen. Die Rekonstruktion des Splitting, sowie die Verallgemeinerung auf Wälder funktionieren analog zu denen aus Lemma 2.1.4 und haben keinen Einfluss auf die Komplexität.

Der Algorithmus lässt sich auf Wälder generalisieren, indem er für jeden Baum und anschließend nochmals mit den Wurzeln aller Bäume als Kinder eine letzte Minimierung mit der Kopffärbung $h = C_\emptyset$ ausgeführt wird. □

Inhaltsverzeichnis

2.2 Fair Correlation Clustering für Eingabegraphen in Form von Kakteen

In diesem Abschnitt wird werden die beiden Algorithmen so angepasst, dass statt nur Wälder auch Kaktusgraphen als Eingabe verarbeitet werden können. Es wird anschließend gezeigt, dass diese Problemdefinition Komplexitätsgrenzen nach dieser Anpassung in denselben Komplexitätsklassen enthalten sind.

Inhaltsverzeichnis

2.2.1 Kakteen mit zwei Farben im Verhältnis 1:2

In der Publikation [10] wird ein Grundgerüst zur Anwendung dynamischer Programme auf Kaktusgraphen vorgestellt. Dieses Grundgerüst wird in diesem Kapitel verwendet, um den in Kapitel 2.1.1 vorgestellten Algorithmus zur Lösung des Fair Correlation Clusterings auf Bäumen mit zwei Farben im Verhältnis 1:2 auf Kaktusgraphen zu erweitern.

Lemma 2.2.1 *Sei ein Kaktusgraph $K = (V, E)$ gegeben, sowie die Menge aller Kreise $\mathcal{C}$ in K. Die Knoten $v \in V(C)$, mit $deg(v) = 2$ werden als C-Knoten bezeichnet. Alle Knoten $v \in V(C)$, mit $deg(v) \geq 3$ werden als H-Knoten bezeichnet. Aus einem solchen Kaktus lässt sich ein Baum $T = (V', E')$ bilden, indem alle Kanten $e \in E(C)$ jedes Kreises $C \in \mathcal{C}$ und alle C-Knoten gelöscht werden. Für jeden Kreis C werde ein Knoten c eingefügt, sodass $\forall C \in \mathcal{C} : \exists! c \in V'$, diese Knoten werden als Kreisknoten bezeichnet. Jeder H-Knoten v wird über eine neue Kante $e = (v, c)$ mit allen Kreisknoten verbunden, für die gilt $\forall v \in \{v \in C | deg(v) \geq 3\} : \exists! e = (v, c) \in E'$, wobei c der Kreisknoten sei der C ersetzt, die Menge dieser Kanten sei mit D gegeben, sodass $E' \backslash D \subseteq E$.*

Beweis: Es wird gezeigt, dass durch den Vorgang alle Kreise entfernt werden, keine neuen Kreise entstehen können und der Zusammenhang der Komponenten nicht beeinträchtigt wird, womit zwangsläufig ein Baum vorliegt. Aufgrund der Eigenschaft eines Kaktusgraphen, dass jede Kante nur in einem Kreis enthalten sein darf, werden durch Entfernen aller Kanten aus einem Kreis die Verbindungen zwischen allen H-Knoten desselben Kreises entfernt. Dadurch entsteht in T für jeden H-Knoten eines Kreises eine Zusammenhangskomponente. Diese Zusammenhangskomponenten werden über die Kanten der Kantenmenge D wieder miteinander verbunden. Der Teilgraph D stellt eine Sammlung von Sternen dar, mit jeweils einem Kreisknoten c in der Mitte. Somit werden keine neuen Kreise eingeführt. Jeder C-Knoten stellt nach Entfernen der Kanten eine Zusammenhangskomponente dar. Die C-Knoten werden alle entfernt. Damit ist T zusammenhängend und enthält keine Kreise. Sei C' die Menge aller Kreisknoten, dann gilt $V' \backslash C' \subseteq V$ und $E' \backslash D \subseteq E$. □

Beispiel 2.2.1 *Sei folgend dargestellter Kaktusgraph gegeben*

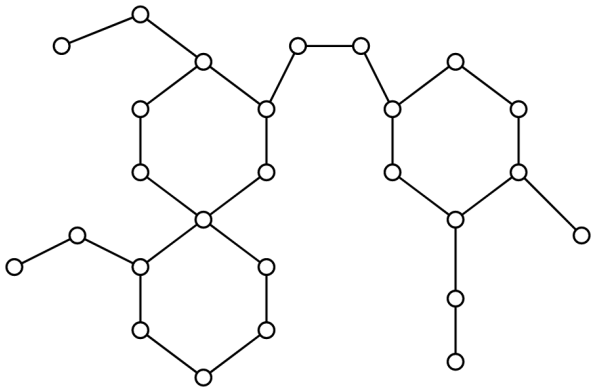

Abbildung 2.8: Beispiel Kaktusgraph

Dieser Kaktus hat drei Kreise, deren C-Knoten und Kanten in der nächsten Darstellung auf der linken Seite rot dargestellt werden. Die H-Knoten, also Knoten, die einem Kreis angehören und mehr als zwei Nachbarn haben, werden grün dargestellt.

In der rechten Darstellung werden die C-Knoten und die Kanten der Kreise, also alle Elemente, die bei der Transformation gelöscht werden, gestrichelt dargestellt. Zu jedem Kreis wird ein Kreisknoten eingefügt, der jeweils in der Mitte des Kreises zu sehen ist und blau markiert ist. Zwischen diesen neuen blauen Knoten und den grünen H-Knoten der jeweiligen Kreise werden Kanten eingefügt, welche im Beweis zu Lemma 2.2.1 als Sterne der Menge D bezeichnet werden.

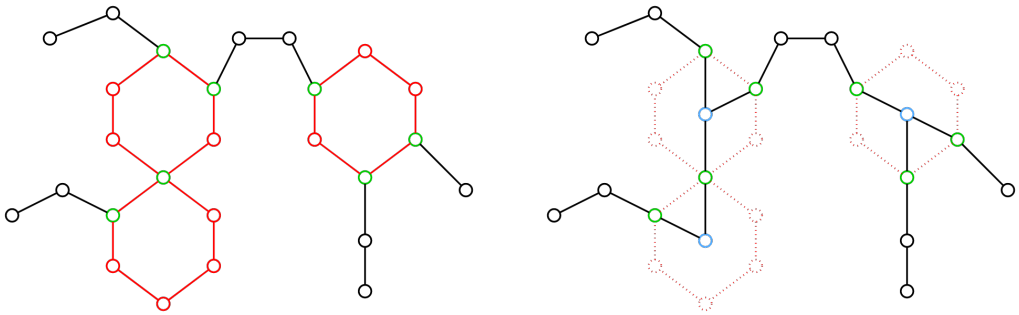

Abbildung 2.9: Transformation von Kaktus zu Baum

Die folgende Darstellung zeigt den Baum an, der sich durch die Transformation ergibt.

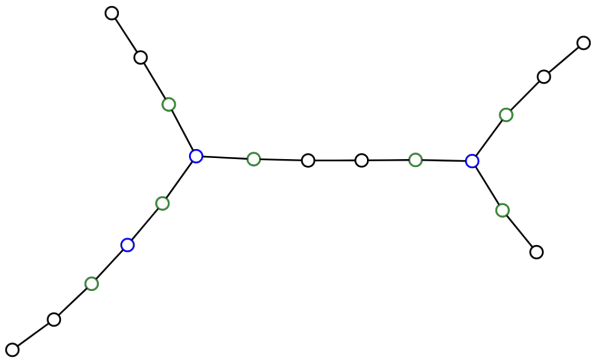

Abbildung 2.10: Baumrepräsentation eines Kaktusgraphen

Definition 2.2.1 (Konfiguration eines Kreises) *Sei ein Kaktusgraph $K = (V, E)$ gegeben. Die Menge an Kreisen in K sei gegeben durch $\mathcal{C}$. Sei $\forall C \in \mathcal{C} : |E(C)| = |V(C)| = m$, dann sei die Konfiguration C_j von C definiert über dessen geordnete Kantenmenge $E(C_j) = E(C) \backslash \{e_j\}$, mit $j \in [|m|]$ und die Knotenmenge $C_j = V(C)$.*

Folgende Darstellung zeigt die verschiedenen Konfiguration.

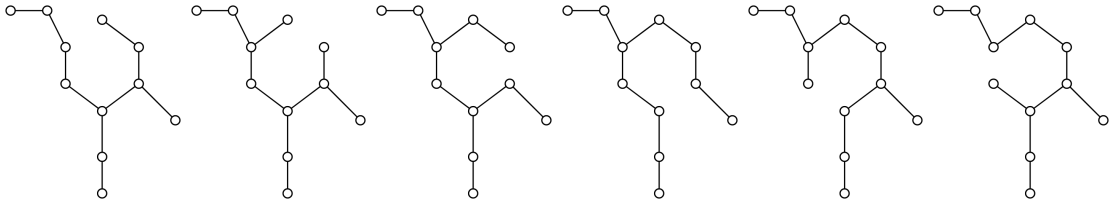

Abbildung 2.11: Konfigurationen

Lemma 2.2.2 *Jede Partition P über einem Kreis C lässt sich ebenfalls über einer der ersten $m-1$ Konfigurationen C_j bilden, sodass es keine Partitionen über der Konfiguration $C_{j=m}$ gibt, die nicht in einer der vorhergehenden Konfigurationen enthalten sei.*

Beweis: Sei w_1 die Wurzel des Kreises C und aller Konfigurationen C_j. Sei die Partition P gegeben und das Cluster $P[w_1]$ als die Vereinigung $P[w_1] = w_1 \cup P_L \cup P_R$ definiert, wobei $P_L = \{w_2, ..., w_i\}$ und $P_R = \{w_j, ..., w_m\}$, für $1 \leq i < j \leq m+1$, wobei für $i = 1$ die Menge P_L leer sei und Analoges für $j = m+1$ und P_R. Dann ist $P[w_1]$ das einzige Cluster in P, für $j = i+1$. Sei $j \neq i+1$, dann gibt es neben $P[w_1]$ noch mehr Cluster in P. Eine solche Partition kann entweder über der Konfiguration C_i oder C_{j-1} erstellt werden, in welcher die zusätzlichen Cluster alle entweder im linken oder rechten Teilbaum von C_i bzw. C_{j-1} enthalten wären. □
Bei folgendem Algorithmus müssen daher nur $m-1$ Konfigurationen durchlaufen werden.

Definition 2.2.2 (Teilkaktus K_{C_j}) *Sei ein Pfad f definiert als eine geordnete Kantenmenge*

$$f = (e_1, e_2, ..., e_m)$$

, mit $m \geq 2$, wobei $\forall e_i \in f : e_i = (v_i, v_{i+1})$ und

$$\forall (i,j) \in \{(i,j) \in \mathbb{N} | 1 \leq i < j \leq m\} : v_i \neq v_j$$

Sei die Abbildung $F : V \times V \to 2^{[E]}$ gegeben, die jedem Knotenpaar die Menge aller Pfade zuordnet, die zwischen ihnen bestehen, sodass zwei Knoten v_1, v_2 in derselben Zusammenhangskomponente enthalten sind, wenn $F(v_1, v_2) \neq \emptyset$.

Sei ein Kaktusgraph $K = (V, E)$ gegeben, mit einem H-Knoten r als Wurzel. Sei ein beliebiger Kreis $C \subseteq K$ gegeben, sowie dessen Wurzelknoten w_1, für den gelte $\forall w \in C : d(w_1) \leq d(w)$, wobei die Abbildung $d : V \to \mathbb{N}$ die Distanz eines Knotens zur Wurzel r angebe. Dann sei für C und r der Teilkaktus

$$K_C = \{e \in E | \forall v_1, v_2 \in e : F(v_1, w_1) \neq \emptyset \wedge F(v_2, w_1) \neq \emptyset \wedge d(w_1) \leq d(v_1) \wedge d(w_1) \leq d(v_2)\}$$

gegeben. Für jede Konfiguration C_j von C sei der Teilkaktus $K_{C_j} = K_C \backslash (w_j, w_{j+1})$ gegeben, mit $j \in [m-1]$.

Bei einem Teilkaktus handelt es sich somit um den Graphen, der entsteht, wenn zu gegebenem Kreis und Wurzel des Kreises nur alle Knoten im Teilgraphen enthalten sind, die "unter" der Wurzel aus Richtung der Wurzel r des Kaktusgraphen liegen, analog zum Teilbaum. Dazu wird die Kante entfernt, die von der Konfiguration mit j angegeben wird. Es sei zu beachten, dass alle anderen Kreise unverändert bleiben.

Folgend wird der Algorithmus selbst vorgestellt und dessen Komplexität hergeleitet. Die Behandlung von Kaktusgraphen basiert grob zusammengefasst darauf, dass der Kaktus über die vorgestellte Transformation in einen Baum überführt wird. Auf diesem Baum wird ein dynamisches Programm ausgeführt, dass die Kreisknoten überspringt und sie erst als Kindknoten behandelt. Wenn ein Kreisknoten als Kind auftritt, werden die Konfigurationen des repräsentierten Kreises betrachtet. Die Konfigurationen können wie Bäume behandelt werden, indem die Kindknoten, von Knoten des Kreises immer nach den Kindknoten behandelt werden, die außerhalb des Kreises liegen. Dadurch können diese vor der Behandlung des Kreisknoten an den entsprechenden H-Knoten bearbeitet werden, welche Kinder des Kreisknoten sind. Damit das funktioniert, ist die rekursive Behandlung der Teilbäume T_v^i von Nöten.

Theorem 2.2.3 *Das Fair Correlation Clustering lässt sich auf gefärbten Kaktusgraphen mit zwei Farben im Verhältnis 1:2 in $O(n^5)$ lösen.*

Beweis: Sei ein Kaktusgraph $K = (V, E)$ gegeben, sowie dessen Baumrepräsentation $T = (V', E')$, nach Lemma 2.2.1 mit einem beliebigen H-Knoten als Wurzel r, sowie die Menge aller Kreisknoten C'.

Zur Lösung des Fair Correlation Clusterings in Kaktusgraphen wird für die Knoten $v \in V' \backslash \{C'\}$ analog zum Algorithmus für Bäume vorgegangen, sodass die Lösung Δ_v^h rekursiv für alle Teilbäume $T_v^i \subseteq T$ berechnet wird. Die Kreisknoten $u \in C'$ werden übersprungen und gesondert behandelt, wenn sie als Kindknoten auftreten. Folgende Abbildung formuliert die Gleichung für die inter-cluster Kosten und die Unterscheidung zwischen Knoten $v \in V'$ und Kreisknoten $u \in C'$.

$$\Delta_v^h(i) = \min_{(h_1,h_2) \in B(v,h)} \begin{cases} \Delta_v^{h_1}(i-1) \odot \Delta_u^{h_2} & \text{, wenn } u \in V' \backslash C' \\ \Delta_v^{h_1}(i-1) \oplus \Delta_u^{h_2} & \text{, wenn } u \in C' \end{cases} \tag{2.14}$$

, für alle $2 \leq i \leq |U(v)|$, mit $u = U(v)[i]$ und $\Delta_v^h = \Delta_v^h(|U(v)|)$. Für Kindknoten, die keine Kreisknoten darstellen gleicht, das Vorgehen dem aus Abbildung 2.10. Die Operation $\odot$ stellt demnach die schrittweise Minimierung über alle $(a_1, a_2) \in A(x)$ dar, sodass

$$\left(\Delta_v^{h_1}(i-1) \odot \Delta_u^{h_2}\right)[x] = \min_{(a_1,a_2) \in A(x)} \left(\Delta_v^{h_1}(i-1)[a_1] + \Delta_u^{h_2}[a_2]\right) \tag{2.15}$$

, wobei $2 \leq i \leq |U(v)|$ und $A(x) = \{(a_1, a_2) \in l^2 | a_1 + a_2 = x\}$ für alle $x \in l$.

Wenn Kreisknoten auftreten, wird stattdessen die Operation $\oplus$ ausgeführt, welche wie folgt definiert sei.

$$\left(\Delta_v^{h_1}(i-1) \oplus \Delta_u^{h_2}\right)[x] = \min_{j \in [m-1]} \left(\Delta_v^{h_1}(i-1) \odot (\Delta_u^{h_2})_j\right)[x] \tag{2.16}$$

Wie zu erkennen ist, werden die beiden Vektoren $\Delta_v^{h_1}(i-1)$ und $(\Delta_u^{h_2})_j$ wieder mit der Operation $\odot$ verarbeitet, also mit der Vorschrift zur Behandlung von Bäumen. Diese Operation wird für alle Konfigurationen $j \in [m-1]$ ausgeführt und das Minimum übernommen. Die Konfiguration hat einen Einfluss auf den zweiten Operanden $(\Delta_v^{h_2})_j$.

$(\Delta_u^h)_j$ sei definiert als das Ergebnis der Abbildung 2.14 über dem Teilkaktus K_{C_j}. Dabei sei C_j die Konfiguration j für den Kreis C, der in V' durch den Kreisknoten u repräsentiert wird. Das bedeutet, dass der Kreis in Konfiguration j wie ein Baum behandelt wird, indem auf die Knoten und Kreise außerhalb von C nur über deren bereits berechneten inter-cluster Kosten zugegriffen wird.
Für alle Knoten der Menge $w \in \{w \in C_j | \exists w_1 \in C_j : d(w) > d(w_1)\}$ sei die Abbildung definiert als

$$(\Delta_w^h)_j(i) = \min_{(h_1,h_2) \in B(v,h)} \left((\Delta_{w_1}^{h_1})_j(i-1) \odot (\Delta_x^{h_2})_j\right) \tag{2.17}$$

und $x \in U_{C_j} \cup H_w$. Dabei wird der Anker $(\Delta_w^h)_j(0)$ definiert wie für Bäume auch mit Abbildung 2.6. Der Knoten w hat demnach höchstens einen Kindknoten, wenn es sich um einen C-Knoten handelt und zwei, wenn es sich um einen H-Knoten handelt, wobei die Repräsentation $H_w \in V'$ einer davon ist.

Für den Wurzelknoten w_1 des Kreises, ist der Knoten $H_{w_1} \in V'$ mit v in Abbildung 2.14 gegeben. Für diesen Knoten können jedoch ebenfalls maximal zwei Kindknoten auftreten, die beide im Kreis liegen. Die Formel gleicht der von Abbildung 2.17, mit dem Unterschied, dass $x \in U_{C_j}$, ohne H_{w_1}.

Die Anker $\Delta_w^h(0)$ sind wie im Beweis zu Lemma 2.1.4 durch die Werte für Blätter gegeben.

Laut Lemma 6 genügt es für $\Delta_v^{h_1}(i-1) \oplus \Delta_u^{h_2}$ nur die minimalen Werte aller Konfigurationen zu speichern. Die Minimierung in Abbildung 2.16 hat eine Laufzeit von $O(n^2)$, weil für jeweils einen festen Index $x \in l$, mit $l = O(n)$, über alle $j \in [m]$ Möglichkeiten, mit $m = O(n)$, der minimale Wert bestimmt wird.

Die Laufzeit der Operation $\oplus$ liegt in $O(n^4)$, weil für jede der j Konfigurationen für jeden der m Knoten maximal zwei Aufrufe der Operation $\odot$ ausgeführt werden und die Operation $\odot$ laut Theorem 2.1.4 eine Laufzeit von $O(n^2)$ aufweist. Danach wird für den Kreis mit einer Laufzeit von $O(n^2)$ das Minimum aller Differenzen x über allen j Konfigurationen ermittelt, was zu $O(n^4 + n^2) = O(n^4)$ führt. Für jede der $O(n)$ Kanten wird die Abbildung 2.14 bestimmt und entweder die Operation $\odot$ oder $\oplus$ ausgeführt, was zu einer Laufzeit von $O(n \cdot (n^4 + n^2)) = O(n^5)$ führt.

Für die Rekonstruktion des Splittings wird für jeden Wert $\Delta_v^h(i)$ die Konfiguration j gespeichert, mit welcher der Wert erzielt wurde. Die Laufzeit der Rekonstruktion wird von der Laufzeit zur Berechnung der Werte Δ_v^h überschattet.

Der Algorithmus lässt sich auf Kaktusgraphen mit mehreren Zusammenhangskomponenten verallgemeinern, indem für jede Zusammenhangskomponente eine Wurzel r_i gewählt und nach Berechnung aller $\Delta_{r_i}^{\emptyset}$ eine weitere Minimierung über $A(x)$ ausgeführt wird, mit $h_i = \emptyset$. □

Die Abbildung $\oplus$ in Abbildung 2.16 wird gegeben mit

$$\left(\Delta_v^{h_1}(i-1) \oplus \Delta_c^{h_2}\right)[x] = \left(\min_{j \in [m-1]} \left(\Delta_v^{h_1}(i-1) \odot (\Delta_c^{h_2})_j\right)[x]\right)$$

Diese Operation wird für den H-Knoten w_1 des Kreisknoten c ausgeführt. Der erste Operand $\Delta_v^{h_1}(i-1)$ repräsentiert den Teilbaum T_v^{i-1}, der die ersten i Kinder beinhaltet. Dies wird in folgender Darstellung aus [10] schematisch durch das Dreiecke dargestellt. Auf der linken Seite stellt der Knoten v den H-Knoten H_{w_1} und somit den Wurzelknoten des Kreises C dar, bzw. dessen Repräsentation in der Knotenmenge V'. Der Kreisknoten selbst ist in der Baumrepräsentation T Kind von H_{w_1}. Der Kreisknoten ist mit c markiert und größer dargestellt als die anderen Knoten, damit in ihm der zugehörige Kreis C dargestellt werden kann. Wie zu erkennen ist, sind alle anderen H-Knoten, welche nicht die Wurzel des Kreises darstellen, Kinder des Kreisknoten c. Diese werden somit vor dem Kreisknoten behandelt, sodass alle Nachbarn der H-Knoten, die sich außerhalb des Kreisknoten befinden, bereits bearbeitet sind. Die Teilbäume, die sich aus den H-Knoten H_{w_2}, H_{w_4} und H_{w_5} und deren Nachbarn außerhalb von C bilden lassen, werden unter den Knoten schematisch als Dreiecke dargestellt. Auf der rechten Seite werden die H-Knoten den Knoten im Kreis C zugeordnet. Ihre Relation wird durch doppelte Striche dargestellt, was hervorhebt, dass es sich dabei um eine Repräsentation der Knoten w_2, w_4 und w_5 des Kaktus K in dessen Baumrepräsentation T handelt und keine Kante in E' oder E.

Mit dem zweiten Operanden $(\Delta_c^{h_2})_j$ wird jede Konfiguration des Kreises C mit w_1 als Wurzel (statt mit dem H-Knoten v) wie ein Baum behandelt. Der Kreis wird in der rechten Darstellung in Form einer beispielhaften Konfiguration dargestellt, indem die Kante (w_4, w_5) ausgelassen wird. Der Knoten w_2 bildet zusammen mit H_{w_2} den Teilbaum $T_{w_2}^1$. Jeder H-Knoten wird als erstes Kind des Knoten behandelt, den er in T vertritt. Zusammen mit dem Knoten w_3 wird dann der Teilbaum $T_{w_2}^2$ gebildet. Auch wenn der Knoten w_2 im Kaktus K mehrere Nachbarn außerhalb des Kreises aufweist, muss der Teilbaum $T_{w_2}^1$ nur genau einmal berechnet werden, statt für jede Konfiguration einzeln. Auch im Fall, dass einer der Kindknoten eines Kreisknoten selbst Wurzel eines Kreises ist, müssen keine Permutationen zwischen den Kreisen berücksichtigt werden, weil die Ergebnisse dieser Kreise in $\Delta_{H_{w_2}}^h$ enthalten sind und somit, wie erwähnt, nur einmal berechnet werden müssen. Gleiches gilt für alle anderen H-Knoten, die Kinder von Kreisen darstellen. Mit der Lösung jeder Konfiguration werden im Knoten v für eine bestimmte Differenz x die inter-cluster Kosten $\Delta_v^i[x]$ gebildet und am Ende das Minimum übernommen.

Zuletzt sei erwähnt, dass für die Ordnung der Kinder eines Wurzelknotens w_1 keine bestimmte Ordnung verwendet wird, weil dieser Wurzel mehrerer Kreise sein kann und eine Ordnung nicht nötig ist.

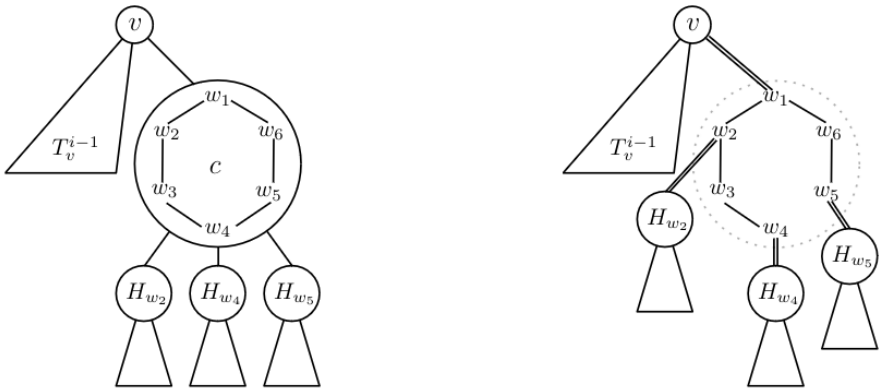

Abbildung 2.12: Bearbeitung von Kreisknoten in dynamischen Programmen

Inhaltsverzeichnis

2.2.2 Kakteen mit beliebig vielen Farben in beliebigem Verhältnis

Theorem 2.2.4 *Sei ein Kaktusgraph $K = (V, E)$ gegeben, sowie dessen Baumrepräsentation $T = (V', E')$, mit einem beliebigen H-Knoten als Wurzel r und der Menge an Kreisknoten C', mit $V' \backslash C' \subseteq V$. Dann lässt sich eine Lösung für das Fair Correlation Clustering für K in $O(n^{5setvars-2})$ berechnen.*

Beweis: Es gilt $|B(v,h)| = O(n^{2setvars-2})$ und $\mathcal{C} = O(n^{setvars-1})$. Der Umgang mit Kreisknoten sei analog zum Beweis zu Theorem 2.2.3. Für Ausdrücke, die hier nicht definiert werden, sei auf dessen Definitionen verwiesen. Die Partitionsfärbungen seien gegeben mit $\mathcal{C}$, sowie die Menge der zulässigen Teilmengenfärbungen für die Cluster mit $L = \{h \in \mathbb{N}^k | \forall i \in [k] : h[i] \leq c_i\}$. Für jeden Knoten $v \in V' \backslash C'$ und jede Kopffärbung $h \in L$ wird Δ_v^h wie folgt berechnet

$$\Delta_v^h(i) = \min_{(h_1,h_2) \in B(v,h)} \begin{cases} \Delta_v^{h_1}(i-1) \odot \Delta_u^{h_2}, & \text{wenn } u \in V' \backslash C' \\ \Delta_v^{h_1}(i-1) \oplus \Delta_u^{h_2}, & \text{wenn } u \in C' \end{cases} \tag{2.18}$$

, für alle $2 \leq i \leq |U(v)|$, mit $u = U(v)[i]$ und $\Delta_v^h(|U(v)|) = \Delta_v^h$. Die Operation $\odot$ sei analog zu Abbildung 2.15 definiert, mit dem Unterschied, dass statt über $A(x)$, über jedes Paar an Partitionsfärbungen minimiert wird

$$\left(\Delta_v^{h_1}(i-1) \odot \Delta_u^{h_2}\right)[x] = \min_{(c_1,c_2) \in \mathcal{C}^2} \left(\Delta_v^{h_1}(i-1)[c_1] + \Delta_u^{h_2}[c_2]\right) \tag{2.19}$$

, mit $x \in l$. Die Operation $\oplus$ sei analog zu Abbildung 2.16 definiert

$$\left(\Delta_v^{h_1}(i-1) \oplus \Delta_u^{h_2}\right)[x] = \min_{j \in [m-1]} \left(\Delta_v^{h_1}(i-1) \odot (\Delta_u^{h_2})_j\right)[x]$$

, wobei $(\Delta_w^h)_j$ definiert sei wie Abbildung 2.18 für den Teilkaktus K_{C_j}. Die Abbildung $(\Delta_w^h)_j$ entspreche der Definition 2.17, mit dem Unterschied, dass die Menge $B(v,h)$ aus Abbildung 2.13 verwendet wird.

Zur Laufzeit: Für jede Kante in E, mit $|E| = O(n)$, wird über alle Kopffärbungen $h \in \mathcal{C}$ für jedes Element in $B(v,h)$, mit $|B(v,h)| = O((n+1)^{2 \cdot setvars-2})$, entweder die Operation $\oplus$ oder $\odot$ ausgeführt. Das führt zu $O(n \cdot O((n+1)^{3 \cdot setvars-3})$ Aufrufen der $\oplus$ bzw. $\odot$.

Nach Theorem 2.2.3 hat die Operation $\odot$ eine Komplexität von $O(n^2)$, welche sich daraus ergibt, dass die Menge $A(x)$ eine Kardinalität von $|A(x)| = O(n)$ aufweist. Da in Abbildung 2.19 statt $A(x)$ die Elemente der Menge $\mathcal{C}^2$ durchlaufen werden, hat die Operation $\odot$ für Eingabegraphen mit Farben in beliebigem Verhältnis die Komplexität $O((n+1)^{2 \cdot setvars-2})$.

In der Operation $\oplus$ wird jede der j Konfigurationen durchlaufen und für jeden der m Knoten maximal zweimal für jede Kopffärbung $h \in \mathcal{C}$ die Abbildung $\odot$ für jede Kombination $(h_1, h_2) \in B(v,h)$ ausgeführt. Damit liegt die Laufzeit der Operation $\oplus$ in $O(n^2 \cdot (n+1)^{3 \cdot setvars-3})$.

Somit liegt die Laufzeit des Algorithmus in $O(n \cdot (n+1)^{3 \cdot setvars-3} \cdot n^2 \cdot (n+1)^{3 \cdot setvars-3})$ bzw. $O\left(n^3 \cdot (n+1)^{6 \cdot setvars-6}\right)$, weil im schlimmsten Fall ausschließlich die Operation $\oplus$ aufgeführt wird. □

Inhaltsverzeichnis

Kapitel 3

Fazit und Ausblick

In dieser Arbeit wurden zwei Algorithmen, die das Fair Correlation Clustering für Eingaben in Form von Wäldern lösen erfolgreich erweitert, sodass sie auch Eingaben in Form von Kakteen entgegennehmen.

Die Basis der Algorithmen ist [12] gegeben. Dort werden Eingaben in Form von Wäldern als Grenze des polynomiell lösbaren vorgestellt. Mit der Erweiterung konnte jedoch bewiesen werden, dass auch Eingaben in Form von Kaktusgraphen effizient lösbar sind. Als willkommenes Nebenprodukt der Arbeit konnten genauere niedrigere Laufzeiten für die ursprünglichen Algorithmen bestimmt werden. Das ist weniger überraschend, weil in [12] bereits erwähnt wurde, dass es wahrscheinlich niedrigere Grenzen gibt.

Für generelle Graphen ist das Problem des Correlation Clusterings NP-schwierig. Das Correlation Clustering stellt eine Spezialisierung des Fair Correlation Clustering dar, bei dem alle Knoten dieselbe Farbe aufweisen. Das Problem des Fair Correlation Clusterings ist eine Generalisierung des Correlation Clusterings und ist somit mindestens so schwierig wie das Correlation Clustering, also ebenfalls NP-schwierig. Selbst für sehr begrenzte Graphen, den Wäldern, ist das Problem des Fair Correlation Clustering bereits mit zwei Farben in einem Verhältnis $1 : p$ mit $p \geq 3$ nicht mehr effizient lösbar. Es konnte gezeigt werden, dass sich die Laufzeiten der beiden Algorithmen für Kakteen leicht erhöhen, weil diese genereller sind als Wälder, aber die Spezialisierung des Problems, beschränkt auf Kaktusgraphen, in denselben Komplexitätsklassen liegen, wie die beschränkt auf Wälder.

Die Schwierigkeit der Fair Correlation Clustering und des Correlation Clusterings schließen die Anwendung auf realistischere Eingabeinstanzen aus. Die Herleitung dieser Algorithmen werden daher keinen großen Einfluss auf die Clusteranalyse haben. Die Kenntnis über die genauen Grenzen ist dennoch aus zwei Aspekten wertvoll: Einerseits müssen in diese Richtung keine weiteren Versuche mit zu hohen Erwartungen unternommen werden. Andererseits könnten die Algorithmen im Falle eines unverhofften Durchbruchs eine Anlaufstelle bieten, um als Framework genutzt zu werden.

Über diese Erkenntnisse in Bezug zum Fair Correlation Clustering hinaus, wurde in Kapitel 1.3.2 ausgearbeitet, dass Fairness sich nicht mit statischen Gesetzen erzwingen lässt, ohne auf Kosten ungeschützter Gruppen zu funktionieren. Stattdessen müssen Gruppen, für die zu wenige Daten für eine unvoreingenommene statistische Auswertung vorliegen, für jeden Datensatz einzeln bestimmt werden.

Inhaltsverzeichnis

Literatur

[1] Saba Ahmadi u. a. *Fair Correlation Clustering*. arXiv:2002.03508 [cs, stat]. Feb. 2020. DOI: 10.48550/arXiv.2002.03508. URL: http://arxiv.org/abs/2002.03508 (besucht am 21. 11. 2023).

[2] Sara Ahmadian und Maryam Negahbani. *Improved Approximation for Fair Correlation Clustering*. arXiv:2206.05050 [cs]. Juni 2022. DOI: 10.48550/arXiv.2206.05050. URL: http://arxiv.org/abs/2206.05050 (besucht am 21. 11. 2023).

[3] Sara Ahmadian u. a. "Clustering without Over-Representation". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. arXiv:1905.12753 [cs]. Juli 2019, S. 267–275. DOI: 10.1145/3292500.3330987. URL: http://arxiv.org/abs/1905.12753 (besucht am 27. 11. 2023).

[4] Sara Ahmadian u. a. *Fair Correlation Clustering*. arXiv:2002.02274 [cs, stat]. März 2020. DOI: 10.48550/arXiv.2002.02274. URL: http://arxiv.org/abs/2002.02274 (besucht am 27. 11. 2023).

[5] Arturs Backurs u. a. *Scalable Fair Clustering*. arXiv:1902.03519 [cs]. Juni 2019. DOI: 10.48550/arXiv.1902.03519. URL: http://arxiv.org/abs/1902.03519 (besucht am 12. 12. 2023).

[6] N. Bansal, A. Blum und S. Chawla. "Correlation clustering". In: *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* ISSN: 0272-5428. Nov. 2002, S. 238–247. DOI: 10.1109/SFCS.2002.1181947. URL: https://ieeexplore.ieee.org/document/1181947 (besucht am 17. 12. 2023).

[7] Hila Becker. "A Survey of Correlation Clustering". en. In: (Mai 2005). URL: http://www.cs.columbia.edu/~hila/clustering.pdf (besucht am 27. 11. 2023).

[8] Suman K. Bera u. a. *Fair Algorithms for Clustering*. arXiv:1901.02393 [cs]. Juni 2019. DOI: 10.48550/arXiv.1901.02393. URL: http://arxiv.org/abs/1901.02393 (besucht am 12. 12. 2023).

[9] Ioana O. Bercea u. a. *On the cost of essentially fair clusterings*. arXiv:1811.10319 [cs]. Nov. 2018. DOI: 10.48550/arXiv.1811.10319. URL: http://arxiv.org/abs/1811.10319 (besucht am 12. 12. 2023).

[10] Maike Buchin und Leonie Selbach. *Partitioning algorithms for weighted trees and cactus graphs*. arXiv:2001.00204 [cs]. März 2022. DOI: 10.48550/arXiv.2001.00204. URL: http://arxiv.org/abs/2001.00204 (besucht am 25. 12. 2023).

[11] Mélanie Cambus u. a. *Massively Parallel Correlation Clustering in Bounded Arboricity Graphs*. arXiv:2102.11660 [cs]. Aug. 2021. DOI: 10.48550/arXiv.2102.11660. URL: http://arxiv.org/abs/2102.11660 (besucht am 04. 01. 2024).

[12] Katrin Casel u. a. *Fair Correlation Clustering in Forests*. arXiv:2302.11295 [cs]. Feb. 2023. DOI: 10.48550/arXiv.2302.11295. URL: http://arxiv.org/abs/2302.11295 (besucht am 21. 11. 2023).

[13] Moses Charikar, Venkatesan Guruswami und Anthony Wirth. “Clustering with Qualitative Information”. In: *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*. FOCS ’03. USA: IEEE Computer Society, Okt. 2003, S. 524. ISBN: 978-0-7695-2040-7. (Besucht am 17. 12. 2023).

[14] Flavio Chierichetti u. a. *Fair Clustering Through Fairlets*. en. arXiv:1802.05733 [cs, stat]. Feb. 2018. URL: http://arxiv.org/abs/1802.05733 (besucht am 26. 11. 2023).

[15] Vincent Cohen-Addad, Euiwoong Lee und Alantha Newman. “Correlation Clustering with Sherali-Adams”. In: *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. ISSN: 2575-8454. Okt. 2022, S. 651–661. DOI: 10.1109/FOCS54457.2022.00068. URL: https://ieeexplore.ieee.org/document/9996889 (besucht am 27. 11. 2023).

[16] Sam Corbett-Davies u. a. *Algorithmic decision making and the cost of fairness*. arXiv:1701.08230 [cs, stat]. Juni 2017. DOI: 10.1145/3097983.309809. URL: http://arxiv.org/abs/1701.08230 (besucht am 29. 11. 2023).

[17] Sonja Dänzer. “Fairness”. de. In: *Handbuch Gerechtigkeit*. Hrsg. von Anna Goppel, Corinna Mieth und Christian Neuhäuser. Stuttgart: J.B. Metzler, 2016, S. 168–173. ISBN: 978-3-476-05345-9. DOI: 10.1007/978-3-476-05345-9_27. URL: https://doi.org/10.1007/978-3-476-05345-9_27 (besucht am 27. 12. 2023).

[18] Erik Demaine und Nicole Immorlica. “Correlation Clustering with Partial Information”. In: Jan. 2003, S. 1–13. ISBN: 978-3-540-40770-6. DOI: 10.1007/978-3-540-45198-3_1.

[19] Erik D. Demaine u. a. “Correlation clustering in general weighted graphs”. en. In: *Theoretical Computer Science* 361.2-3 (Sep. 2006), S. 172–187. ISSN: 03043975. DOI: 10.1016/j.tcs.2006.05.008. URL: https://linkinghub.elsevier.com/retrieve/pii/S0304397506003227 (besucht am 04. 01. 2024).

[20] E. Diday und J. C. Simon. “Clustering Analysis”. en. In: *Digital Pattern Recognition*. Hrsg. von King Sun Fu. Communication and Cybernetics. Berlin, Heidelberg: Springer, 1976, S. 47–94. ISBN: 978-3-642-96303-2. DOI: 10.1007/978-3-642-96303-2_3. URL: https://doi.org/10.1007/978-3-642-96303-2_3 (besucht am 28. 11. 2023).

[21] Hartmut Ernst, Jochen Schmidt und Gerd Beneken. “Algorithmen – Berechenbarkeit und Komplexität”. de. In: *Grundkurs Informatik: Grundlagen und Konzepte für die erfolgreiche IT-Praxis - Eine umfassende, praxisorientierte Einführung*. Hrsg. von Hartmut Ernst, Jochen Schmidt und Gerd Beneken. Wiesbaden: Springer Fachmedien, 2016, S. 415–468. ISBN: 978-3-658-14634-4. DOI: 10.1007/978-3-658-14634-4_11. URL: https://doi.org/10.1007/978-3-658-14634-4_11 (besucht am 02. 12. 2023).

[22] Hartmut Ernst, Jochen Schmidt und Gerd Beneken. “Automatentheorie und formale Sprachen”. de. In: *Grundkurs Informatik: Grundlagen und Konzepte für die erfolgreiche IT-Praxis - Eine umfassende, praxisorientierte Einführung*. Hrsg. von Hartmut Ernst, Jochen Schmidt und Gerd Beneken. Wiesbaden: Springer Fachmedien, 2016, S. 371–414. ISBN: 978-3-658-14634-4. DOI: 10.1007/978-3-658-14634-4_10. URL: https://doi.org/10.1007/978-3-658-14634-4_10 (besucht am 30. 11. 2023).

[23] Vladimir Estivill-Castro. “Why so many clustering algorithms: a position paper”. In: *ACM SIGKDD Explorations Newsletter* 4.1 (Juni 2002), S. 65–75. ISSN: 1931-0145. DOI: 10.1145/568574.568575. URL: https://doi.org/10.1145/568574.568575 (besucht am 27. 12. 2023).

[24] Michael Feldman u. a. *Certifying and removing disparate impact*. arXiv:1412.3756 [cs, stat]. Juli 2015. DOI: 10.48550/arXiv.1412.3756. URL: http://arxiv.org/abs/1412.3756 (besucht am 28. 11. 2023).

[25] Zachary Friggstad und Ramin Mousavi. “Fair Correlation Clustering with Global and Local Guarantees”. In: *Algorithms and Data Structures: 17th International Symposium, WADS 2021, Virtual Event, August 9–11, 2021, Proceedings*. Berlin, Heidelberg: Springer-Verlag, Aug. 2021, S. 414–427. ISBN: 978-3-030-83507-1. DOI: 10.1007/978-3-030-83508-8_30. URL: https://doi.org/10.1007/978-3-030-83508-8_30 (besucht am 11. 12. 2023).

[26] *Gerrymandering: Wenn sich US-Politiker an die Macht tricksen — ZDFheute*. en. URL: https://zdfheute-stories-scroll.zdf.de/usa_wahl_gerrymandering/index.html (besucht am 25. 12. 2023).

[27] Sanchit Kalhan, Konstantin Makarychev und Timothy Zhou. *Improved algorithms for Correlation Clustering with local objectives*. arXiv:1902.10829 [cs]. Juni 2019. DOI: 10.48550/arXiv.1902.10829. URL: http://arxiv.org/abs/1902.10829 (besucht am 12. 12. 2023).

[28] Richard M. Karp. “Reducibility among Combinatorial Problems”. en. In: *Complexity of Computer Computations*. Hrsg. von Raymond E. Miller, James W. Thatcher und Jean D. Bohlinger. Boston, MA: Springer US, 1972, S. 85–103. ISBN: 978-1-4684-2003-6 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9. URL: http://link.springer.com/10.1007/978-1-4684-2001-2_9 (besucht am 11. 12. 2023).

[29] Matthäus Kleindessner, Pranjal Awasthi und Jamie Morgenstern. *Fair k-Center Clustering for Data Summarization*. arXiv:1901.08628 [cs, stat]. Mai 2019. DOI: 10.48550/arXiv.1901.08628. URL: http://arxiv.org/abs/1901.08628 (besucht am 12. 12. 2023).

[30] Shrinu Kushagra, Shai Ben-David und Ihab Ilyas. *Semi-supervised clustering for de-duplication*. en. arXiv:1810.04361 [cs, stat]. Okt. 2018. URL: http://arxiv.org/abs/1810.04361 (besucht am 11. 12. 2023).

[31] Minlei Liao u. a. “Cluster analysis and its application to healthcare claims data: a study of end-stage renal disease patients who initiated hemodialysis”. In: *BMC Nephrology* 17 (März 2016), S. 25. ISSN: 1471-2369. DOI: 10.1186/s12882-016-0238-2. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4776444/ (besucht am 29. 11. 2023).

[32] Zachary C. Lipton, Alexandra Chouldechova und Julian McAuley. *Does mitigating ML's impact disparity require treatment disparity?* arXiv:1711.07076 [cs, stat]. Jan. 2019. DOI: 10.48550/arXiv.1711.07076. URL: http://arxiv.org/abs/1711.07076 (besucht am 29. 11. 2023).

[33] Ulrike von Luxburg, Robert C. Williamson und Isabelle Guyon. “Clustering: Science or Art?” en. In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. ISSN: 1938-7228. JMLR Workshop und Conference Proceedings, Juni 2012, S. 65–79. URL: https://proceedings.mlr.press/v27/luxburg12a.html (besucht am 28. 11. 2023).

[34] Guillermo Owen und Bernard Grofman. “Optimal partisan gerrymandering”. In: *Political Geography Quarterly* 7.1 (Jan. 1988), S. 5–22. ISSN: 0260-9827. DOI: 10.1016/0260-9827(88)90032-8. URL: https://www.sciencedirect.com/science/article/pii/0260982788900328 (besucht am 25. 12. 2023).

[35] Dana Pessach und Erez Shmueli. *Algorithmic Fairness*. arXiv:2001.09784 [cs, stat]. Jan. 2020. DOI: 10.48550/arXiv.2001.09784. URL: http://arxiv.org/abs/2001.09784 (besucht am 10. 11. 2023).

[36] Jella Pfeiffer u. a. “Algorithmic Fairness in AI”. en. In: *Business & Information Systems Engineering* 65.2 (Apr. 2023), S. 209–222. ISSN: 1867-0202. DOI: 10.1007/s12599-023-00787-x. URL: https://doi.org/10.1007/s12599-023-00787-x (besucht am 10. 11. 2023).

[37] Walter Purkert und Hans Joachim Ilgauds. *Georg Cantor 1845-1918*. ger. Vita mathematica 1. Basel Boston Stuttgart: Birkhäuser, 1987. ISBN: 978-3-7643-1770-6.

[38] J. Rollin und A. Schulz. *Effiziente Algorithmen - Einführung in die Theorie der Algorithmen*. Deutsch. Bd. Kurseinheit 1: Einführung in die Theorie der Algorithmen. Effiziente Algorithmen 1. Hagen, Deutschland: FernUniversität in Hagen.

[39] Chaitanya Swamy. "Correlation Clustering: maximizing agreements via semidefinite programming". In: *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*. SODA '04. USA: Society for Industrial und Applied Mathematics, Jan. 2004, S. 526–527. ISBN: 978-0-89871-558-3. (Besucht am 17. 12. 2023).

[40] William Vickrey. "On the Prevention of Gerrymandering". In: *Political Science Quarterly* 76.1 (1961). Publisher: [Academy of Political Science, Wiley], S. 105–110. ISSN: 0032-3195. DOI: 10.2307/2145973. URL: https://www.jstor.org/stable/2145973 (besucht am 25. 12. 2023).

[41] Xiaomeng Wang, Yishi Zhang und Ruilin Zhu. "A brief review on algorithmic fairness". en. In: *Management System Engineering* 1.1 (Nov. 2022), S. 7. ISSN: 2731-5843. DOI: 10.1007/s44176-022-00006-z. URL: https://doi.org/10.1007/s44176-022-00006-z (besucht am 10. 11. 2023).

[42] Ingo Wegener. *Komplexitätstheorie*. Springer-Lehrbuch. Berlin, Heidelberg: Springer, 2003. ISBN: 978-3-540-00161-4 978-3-642-55548-0. DOI: 10.1007/978-3-642-55548-0. URL: http://link.springer.com/10.1007/978-3-642-55548-0 (besucht am 27. 12. 2023).

[43] Lulu Wen, Kaile Zhou und Shanlin Yang. "A shape-based clustering method for pattern recognition of residential electricity consumption". In: *Journal of Cleaner Production* 212 (März 2019), S. 475–488. ISSN: 0959-6526. DOI: 10.1016/j.jclepro.2018.12.067. URL: https://www.sciencedirect.com/science/article/pii/S0959652618337697 (besucht am 29. 11. 2023).

[44] Dongkuan Xu und Yingjie Tian. "A Comprehensive Survey of Clustering Algorithms". en. In: *Annals of Data Science* 2.2 (Juni 2015), S. 165–193. ISSN: 2198-5804, 2198-5812. DOI: 10.1007/s40745-015-0040-1. URL: http://link.springer.com/10.1007/s40745-015-0040-1 (besucht am 29. 11. 2023).