Clustern mit qualitativen Informationen

Jan Niehues

01.01.2007

Inhaltsverzeichnis

Kapitel 1

Einleitung

Diese Ausarbeitung beschäftigt sich mit dem Paper "Clustering with Qualitative Information" von Moses Charikar, Venkatesan Guruswami und Anthony Wirth [3]. Das Paper wurde zum "44th Annual IEEE Symposium on Foundations of Computer Science(FOCS'03)" veröffentlicht und befasst sich mit dem Problem Elemente zu clustern, für die paarweise Vergleiche existieren. Die Veröffentlichung baut auf einer Arbeit von Banasal et al. [4] auf und versucht einen besseren Überblick über die Approximierbarkeit der Probleme zu geben. Dazu werden zum einen Approximationsalgorithmen angegeben, zum anderen werden Aussagen über die Komplexität des Problems bewiesen.

1.1 Problem

Das Problem Elemente in verschiedene Gruppen einzuteilen taucht in vielen Bereichen auf und wurde daher gründlich untersucht. Allerdings wird das Problem dabei häufig so modelliert, dass für jeweils zwei Elemente eine Entfernung angeben wird, die angibt, wie verschieden die Elemente sind.
In dem bearbeitetem Paper wird das Problem jedoch anders modelliert. Es werden für jedes Paar von Elementen nur qualitative Informationen angegeben, d.h. es wird angegeben, ob sie übereinstimmen oder im Widerspruch zueienander stehen.
Das Ziel besteht dann darin, eine Clusterung zu finden, so dass möglichst zwei gleichartige Objekte immer in das dasselbe Cluster und zwei Objekte, die im Widerspruch zueinander stehen, in verschiedene Cluster eingeteilt werden. Falls so eine Einteilung existiert, ist das Problem einfach zu lösen.Falls die paarweisen Vergleiche jedoch nicht wiederspruchslos sind, ist das Problem algorithmisch interessant.

1.2 Frühere Ergebnisse

Basnal et al. haben in [4] gezeigt, dass die Minimierungsvariante des Problems (MinDisAgree) auf vollständigen Graphen NP-hart ist. Außerdem haben sie einen Approximationsalgorithmus mit konstantem Faktor angegeben. Allerdings ist dieser Approximationsfaktor mit einem Wert von 17433 sehr hoch. Ebenso wurde in obigen Paper nur gezeigt, dass das Problem APX-schwer ist. In [1] wurde unabhängig von dieser Veröffentlichung auch ein Approximationsalgorithmus mit Faktor $O(\log n)$ angegeben.
Für die Maximierungsvariante des Problems (MaxAgree) auf vollständigen Graphen haben die Autoren von [4] bereits gezeigt, dass es NP-hart ist und haben eine PTAS angegeben.
Für den Fall auf allgemeinen Graphen wurde bisher nur ein naiver Algorithmus angegeben.

Kapitel 2

Übersicht

Im Paper [3] werden beide Probleme betrachtet. Außerdem werden die Probleme sowohl auf vollständigen Graphen als auch auf allgemeinen Graphen betrachtet. Somit ergeben sich vier unterschiedliche Fälle des Problems. Für diese werden verschiedene Algorithmen angegeben und die Komplexität der Probleme wird getrennt betrachtet. Im Abschnitt 2.1 werden zunächst die Probleme formal beschrieben. Danach wird in diesem Kaptiel eine Übersicht über die Ergebnisse des Papers gegeben. Dabei werden zunächst die Approximationsalgorithmen für das MinDisAgree und danach für das MaxAgree-Problem betrachtet. Anschließend werden die Komplexitätsaussagen des Papers betrachtet.

2.1 Formale Beschreibung

Das Problem wird mittels eines Graphen modelliert, dessen Knoten den Elementen entsprechen. Die Kanten sind mit "+" oder "-" beschriftet, womit entsprechend eine Übereinstimmung oder ein Widerspruch zwischen den Elementen dargestellt wird. Falls für jedes Paar eine Bewertung vorliegt, erhält man somit einen vollständigen Graphen. In dem Paper wird aber auch der Fall behandelt in dem nicht alle Bewertungen vorliegen und man somit einen allgemeinen Graphen betrachten muss.
Das Ziel, die Elemente möglichst gut entsprechend der Informationen zu unterteilen, kann auf zwei verschiedene Arten formuliert werden. Es ergibt sich eine Maximierungs- und eine Minimierungsvariante, die im folgenden als MaxAgree und MinDisAgree bezeichnet wird. Beim MinDisAgree-Problem wird die Anzahl der Widersprüche, welche negativen Kanten im Cluster und positiven Kanten zwischen Clustern sind, minimiert. Analog wird beim MinDisAgree-Problem die Anzahl der Übereinstimmungen, weche positiven Kanten im Cluster und positiven Kanten zwischen Clustern sind, maximiert.

2.2 Algorithmen für MinDisAgree

Für das MinDisAgree-Problem werden Approximationsalgorithmen angegeben, die auf der Rundung der Lösung einer relaxierten Version eines ganzzahligen Linaren Programms (ILP) beruhen. Das ILP kann dann relaxiert werden und entsprechend kann in polynomialer Zeit dafür eine optimale Lösung gefunden werden. Im Paper wird dann für vollständige wie auch für allgemeine Graphen jeweils eine Rundungsstragie angegeben, um aus dieser Lösung eine gültige Lösung für das ursprüngliche Problem zu gewinnen. Dabei kann gezeigt werden, dass der Algorithmus für vollständige Graphen einen Approximationsfaktor von 4 besitzt und mit der verwendeten ILP-Formulierung auch nur ein Approximationsfaktor von 2 erreicht werden kann. Für das Problem auf allgemeinen Graphen wird nur ein Algorithmus mit einem $O(\log n)$ Approximationsfaktor angegeben. Allerdings kann gezeigt werden, das mittels der ILP-Formulierung kein besserer Algorithmus möglich ist.

2.3 Algorithmen für MaxAgree

Für das MaxAgree-Problem erreicht ein naiver Algorithmus bereits einen Approximationsfaktor von 1/2. Der Algorithmus gruppiert dazu lediglich alle Elemente in einem Cluster, falls das Gesamtgewicht der positven Kanten größer ist, als das Gesamtgewicht der negativen Kanten. Falls dies nicht der Fall ist, werden alle Elemente in ihr eigenes Cluster gepackt. Zusätzlich haben Basnal et al. bereits in [4] ein PTAS für das Problem auf vollständigen Graphen angegeben. Somit stellt sich hier die Frage, was noch verbessert werden kann. Die Autoren stellen dazu einen 0.7664-Approximationsalgorithmus vor, der auf Semi-definiten Programmen beruht.

2.4 Komplexitätsaussagen

Die Autoren führen verschiedene Reduktionen von bekannten Probelmen auf die Cluster-Probleme durch um zu zeigen, dass es NP-hart ist, die Probleme mit einem bestimmten Faktor zu approximieren. Somit kann es dann, falls $P \neq NP$, keinen Approximationsaglorithmus geben, der das Problem mit diesem gezeigten Faktor approximiert. Für das MinDisAgree-Problem auf allgemeinen Graphen kann hier zwar keine zufriedenstellende Schranke gezeigt werden, jedoch wird in diesem Fall gezeigt, dass das Problem genauso schwer zu lösen ist, wie das Minimum Multicut Problem.
Für das MinDisAgree-Problem auf vollständigen Graphen kann so jedoch gezeigt werden, dass es einen Faktor $a > 1$ gibt, so dass es NP-hart ist, das Problem mit diesem Faktor zu approximieren. Außderdem wird für das MaxAgree-Problem auf allgemeinen Graphen gezeigt, dass es einen kon-

Tabelle 2.1: Zusammenfassung der Ergebnisse

	vollständige Graphen	allgemeine Graphen
MinDisAgree	Faktor 4 Approximation Ganzzahligkeitslücke 2 ex. a $>$ 1 -> NP-hart zu approx.	$O(\log n)$ Approximation Algo Ganzzahligkeitslücke $O(\log n)$ Red. von Minimum Multicut
MaxDisAgree		0.7664 Approximation fast erfüllende Beispiele Red. von Max3Sat

stanten Faktor gibt, so dass es NP-hart ist das Problem mit diesem Faktor zu approximieren. Für das MaxAgree-Problem auf vollständigen Graphen kann so eine Schranke nicht exitieren, da Basnal et al. in [4] bereits ein PTAS angegeben haben.

2.5 Aufbau der Ausarbeitung

Die Ergbnisse sind in der Tabelle 2.1 noch einmal kurz zusammengefasst. Es ist auffällig, dass das MinDisAgree-Problem auf allgemeinen Graphen deutlich schwerer zu approximieren ist, als die übrigen Probleme. Dies ist insofern verwunderlich, weil der einzige Unterschied zum MinDisAgree-Problem auf vollständigen Graphen darin besteht, dass für einige Elemente keine Vergleiche angegeben sind. Somit ist es für die Lösung irrelevant, ob diese Elemente in unterschiedliche Cluster kommen oder ob sie in dasselbe Cluster gruppiert werden. Hinzu kommt, dass der Algorithmus für vollständige Graphen auch einen allgemeinen Graphen als Eingabe bearbeiten kann und eine Clusterung der Knoten liefert. Allerdings ist das Ergebniss eine nicht so gute Approximation wie für vollständige Graphen. Deshalb werden wir im nächsten Kapitel diesen Algorithmus genauer betrachten und untersuchen, wieso er nur auf vollständigen Graphen eine so gute Approximation liefert.
Um weitere Informationen über die Komplexität des MinDisAgree-Problems auf allgemeinen Graphen zu erhalten, werden wir uns das MaxAgree-Problem auf allgemeinen Graphen anschauen. Mittels des Zusammenhangs zwischen den beiden Problemen lassen sich aus Aussagen über das MaxAgree-Problem auch Aussagen über das MinDisAgree-Probelm gewinnen. Dazu werden wir uns den Algorithmus für MaxAgree in speziellen Fällen anschauen und zum anderen werden wir uns die Komplexitätsaussagen über das Problem anschauen.

Kapitel 3

Algorithmen für MinDisAgree

Die in [3] vorgestellten Algorithmen zur Approximation der Lösung des MinDisAgree-Problems arbeiten mit Hilfe eines ganzzahligen Linaren Programms. Die Formulierung des Programms wird in Abschnitt 3.1 beschrieben. Anschließend wir in Abschnitt 3.2 der Rundungsalgorithmus für vollständige Graphen und in Abschnitt 3.5 der Algorithmus für allgemeine Graphen beschrieben. Für den Algorithmus auf vollständigen Graphen kann dann gezeigt werden, dass er eine 4 Approximation des Problems liefert, d.h. die Lösung des Algorithmus geteilt durch die optimale Lösung ist kleiner gleich 4. Der Beweis dafür wird in Abschnitt 3.3 beschrieben.

3.1 Das Lineare Programm

Das beschriebene Lineare Programm ist ähnlich zu dem des GVY Minmum Multicut Algorithmuses[5]. Dazu wird für alle Knotenpaare i, j eine binäre Variable x_{ij} eingefürt. Dabei beschreibt die Varibale x_{ij}, ob Knoten i und j in dem gleichen oder in verschieden Clustern sind: $x_{ij} = 0$ bedeutet, dass die Knoten im gleichen Cluster sind und der Wert 1, dass die Knoten in verschiedenen Clustern sind. In der relaxierten Version des Problems kann man den reellen Wert der Variablen als Entfernung der beiden Punkte betrachten. Damit die Variablenbelegung auch einer gültigen Einteilung entspricht, muss beachtet werden, dass es sich bei den Clustern um Äquivalenzklassen handelt. Deshalb folt aus $x_{ij} = 0$ und $x_{jk} = 0$, dass auch $x_{ik} = 0$ ist. Dies kann mittels einer Dreiecksungleichung modelliert werden.
Neben der oben beschriebenen Nebenbedinungen muss noch die zu optimierende Funktion bestimmt werden. Diese muss sowohl die postitiven Kanten zwischen zwei Clustern, als auch die negativen Kanten innerhalb eines Clusters betrachten.
Somit wird das gesamte Lineare Programm wie folgt beschrieben:

Abbildung 3.1: Algorithmus für MinDisAgree auf vollständigen Graphen

1.	Wähle Knoten u aus S zufällig aus
2.	Sei T Menge aller Knoten die Abstand kleiner $\frac{1}{2}$ von u haben (ohne u selbst)
3.	x sei der mittlere Abstand der Knoten in T von u
	a) $(x \geq \frac{1}{4} \rightarrow u$ in Singelton Cluster $C = \{u\}$
	b) $(x < \frac{1}{4}) \rightarrow$ alle Knoten in ein Cluster $C = T \cup \{u\}$
4.	$S = S - C$

$$\begin{aligned} \text{minimiere} \quad & \left(\sum_{+(ij)} w_{ij} x_{ij} + \sum_{-(ij)} w_{ij} \left(1 - x_{ij}\right) \right) \\ \text{unter Nebenbedingung} \quad & x_{ik} \leq x_{ij} + x_{jk} \end{aligned}$$

oder im ungewichteten Fall:

$$\begin{aligned} \text{minimiere} \quad & \left(\sum_{+(ij)} x_{ij} + \sum_{-(ij)} \left(1 - x_{ij}\right) \right) \\ \text{unter Nebenbedingung} \quad & x_{ik} \leq x_{ij} + x_{jk} \end{aligned}$$

In obigen Formel bezeichnet $+(ij)$ die Menge aller positiven Kanten und $-(ij)$ die Menge aller negativen Kanten.

3.2 Der Algorithmus für vollständige Graphen

Der Algorithmus braucht als Eingabe die optimale Lösung des relaxierten LP. Dazu wird zunächst das ILP relaxiert und mit einem Standardverfahren in polynomialer Zeit gelöst. Der Algorithmus arbeitet dann auf einer Menge S, in der zu Beginn alle Knoten sind. Der Algorithmus führt daraufhin die in Abbildung 3.1 dargestellten Schritte durch bis die Menge S leer ist.

3.3 Analyse des Approximationsfaktors

Die Analyse des Algorithmuses wird zeigen, dass der Algorithmus eine 4-Approximation liefert. Da der Algorithmus auch auf allgemeinen Graphen funktioniert, wollen wir besonders beobachtetn, wieso diese Garantie nur für vollständige Graphen gilt und nicht auf allgemeinen Graphen verallgemeinert werden kann.

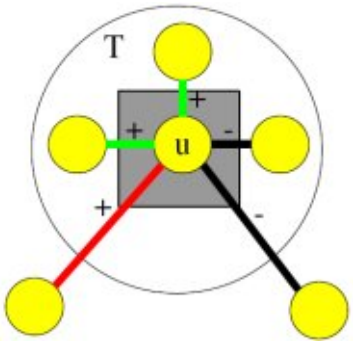

Abbildung 3.2: Singelton Cluster

Um die Behauptung zu zeigen, müssen wir nachweisen, dass das Verhältnis von der Lösung des Algorithmuses zur optimalen Lösung immer kleiner gleich vier ist. Das Problem dabei ist, dass wir die optimale Lösung nicht kennen. Die optimale Lösung des relaxierten Problems ist allerdings bekannt. Außerdem ist jede Lösung des ursprünglichen Problems natürlich auch eine Lösung des relaxierten Problems. Somit muss der Wert der optimalen Lösung auf jeden Fall größer gleich dem Wert der Lösung des ILP sein. Damit reicht es die Ungleichung 3.1 zu zeigen. Damit ist auch der Approximationsfaktor kleiner gleich vier und wir haben obige Behauptung gezeigt.

$$\frac{ALGO}{OPT_{LP}} \leq 4 \tag{3.1}$$

Um die Ungleichung 3.1 zu beweisen, werden wir die LP Kosten einer oder mehrerer Kanten mit den Kosten dieser Kante bzw. Kanten in der Lösung des Algorithmuses vergleichen. Dazu werden wir 2 Fälle unterscheiden. Zunächst wird der Fall eines Singelton Clusters betrachtet. Hierbei werden alle Kanten betrachtet, die inzident zu dem Knoten des Singelton Clusters sind. Danach wird der Fall untersucht, in dem ein größeres Cluster gebildet wurde. Hier werden sowohl die Kanten innerhalb des Clusters, als auch die Kanten, die aus dem Cluster herrausführen betrachet.

Sei nun ein Singelton Cluster mit Knoten u gegeben. Dieser Fall ist in Abbildung 3.2 dargestellt. Damit verursachen alle zu u inzidenten Kanten, die ein negatives Label haben, keine Kosten. Diese sind in Abbildung 3.2 schwarz dargestellt. Für alle positiven Kanten unterscheiden wir wiederrum zwei Fälle. Sei T wie im Algorithmus, die Menge aller Knoten, deren Abstand von u kleiner als $1/2$ ist, ohne u selbst, so betrachten wir zunächst alle positiven Kanten von u zu Knoten in T. Anschließend werden die positiven Kanten zu Knoten betrachtet, die außerhalb von T liegen. Erstere sind in Abbildung 3.2 grün dargestellt, die anderen rot.
Da der Graph einfach ist, können maximal $|T|$ Kanten von u zu Knoten

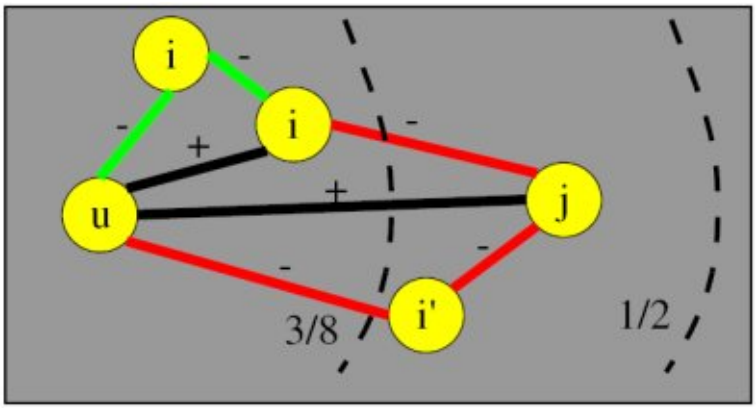

Abbildung 3.3: größeres Cluster

in T führen und somit kann der Beitrag dieser Kanten zu Lösung des LP auch maximal $|T|$ sein. Die LP Kosten aller Kanten zu Knoten aus T lassen sich nach Formel 3.2 mit $|T|/4$ abschätzen, da alle $x_{ui} \leq 1/2$ sind. Dies gilt insbesondere auch, falls der Graph nicht vollständig ist.

$$\sum_{i \in T+(ui)} x_{ui} + \sum_{i \in T-(ui)} (1 - x_{ui}) \geq \sum_{i \in T} x_{ui} \geq \frac{|T|}{4} \tag{3.2}$$

Betrachten wir nun die Kanten von u zu Knoten i außerhalb von T. So kann dies in der Lösung des Algorithmuses wiederrum maximal zu einem Fehler von 1 pro Kante führen. Außerdem gilt, dass $x_{ui} \geq 1/2$ ist, somit ist der Quotient aus beiden kleiner gleich 2 und somit auch kleiner gleich 4. Diese Analyse ist auch auf allgemeinen Graphen weiterhin gültig.

Sei nun ein größeres Cluster gegeben. In diesem Fall können zwei Fehler auftreten. Zum einen können negative Kanten innerhalb des Clusters verlaufen, zum anderen Können positive Kanten zu Knoten außerhalb des Clusters führen.

Bevor wir die beiden Fälle genauer untersuchen, halten wir folgende Beobachtung fest. Wegen der Dreiecksungleichung gilt die linke Ungleichung in 3.3. Somit ergibt sich für die Kosten x_{ij} einer positiven Kante (i, j) die rechte Seite in Folgerung 3.3. Analog ergibt sich für die Kosten $1 - x_{ij}$ einer negative Kante die rechte Ungleichung in 3.4, da die linke Ungleichung in 3.4 gilt.

$$x_{uj} \leq x_{ui} + x_{ij} \quad \rightarrow \quad x_{ij} \geq x_{uj} - x_{ui} \tag{3.3}$$

$$x_{ij} \leq x_{ui} + x_{uj} \quad \rightarrow \quad 1 - x_{ij} \geq max(0, 1 - x_{ui} - x_{uj}) \tag{3.4}$$

Betrachten wir zunächst die negativen Kanten innerhalb eines Clusters wie in Abbildung 3.3. Seien u und T wie im Beweis definiert, so betrachten wir

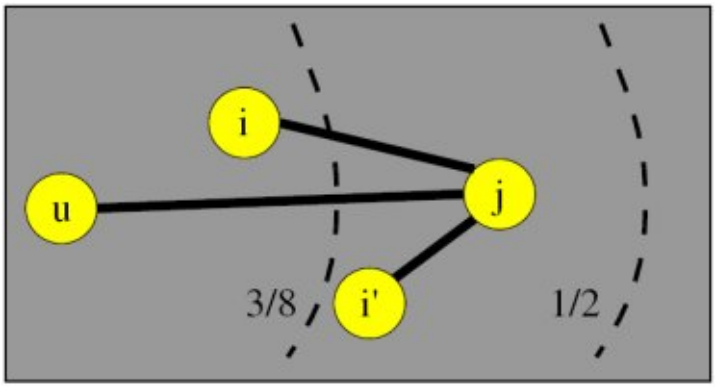

Abbildung 3.4: Situation bei negativen Kanten innerhalb von Cluster für festes j

zunächst negative Kanten (i, j), für die sowohl i als auch j innerhalb einer Entfernung von $3/8$ von u liegen.Diese sind in Abbildung 3.3 grün dargestellt. Diese Kante führt zu einem Fehler von 1 und laut der Berechnung in Formel 3.5, die die obige Beobachtung benutzen, sind die LP-Kosten von $1 - x_{ij}$ mindestens $1/4$. Somit gilt hier weiterhin, dass der Algorithmus eine 4-Approximation berechnet, insbesondere auch, falls der Graph nicht vollständig ist.

$$1 - x_{ij} \geq 1 - x_{ui} - x_{uj} \geq 1 - \frac{3}{8} - \frac{3}{8} = \frac{1}{4} \tag{3.5}$$

Nun müssen noch die Kanten, für die obige Bedingung nicht gilt, betrachtet werden. Diese sind in Abbildung 3.3 rot eingefärbt. Dabei analysieren wir für ein festes j alle Kanten zu Knoten mit geringerer Entfernung. Dieser Fall ist in Abbildung 3.3 veranschaulicht. Für die weiteren Überlegungen müssen wir zwei weitere Variablen definieren. Die Variable p_j ist definieniert, als die Anzahl an posititven Kanten (i, j) für die i näher am Punkt u ist, als das feste j und n_j sei die Anzahl der negativen Kanten, für die die Bedingung gilt.

Mittels dieser Beobachtung lassen sich die gesamten LP Kosten der Kanten von j zu Knoten mit kleineren Entfernung wie in Formel 3.6 umschreiben.

$$\begin{aligned} & \sum_{i:i<j+(ij)} x_{ij} + \sum_{i:i<j-(ij)} (1 - x_{ij}) \\ \overset{(Beobachtung)}{=} & \sum_{i:i<j+(ij)} (x_{uj} - x_{ui}) + \sum_{i:i<j-(ij)} (1 - x_{ui} - x_{uj}) \\ = & \quad p_j x_{uj} - n_j(1 - x_{uj}) - \sum_{i:i<j} x_{ui} \end{aligned} \tag{3.6}$$

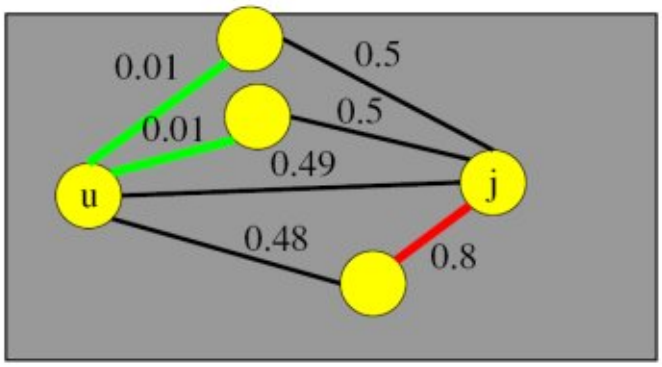

Abbildung 3.5: allgmeiner Graph

Der letzte Term dieser Berechnung ist dabei eine Summe über eine Teilmenge von T. Falls der Graph vollständig ist, ist j mit allen Knoten, die eine kleinere Entfernung haben verbunden und somit fehlen nur Elemente, die eine größere Entfernung als j haben. Knoten j muss eine größere Entfernung als $3/8$ haben, da die Kanten (i, j) sonst nach dem ersten Fall behandlt werden müssten. Daher fehlen im Falle eines vollständigen Graphen in der Summe nur Werte größer als $3/8$ und deshalb ist der Mittelwert der Summe weiterhin kleiner als $1/4$. So kann man nur in vollständigen Graphen die Kosten mit Formel 3.7 abschätzen.

$$p_j x_{uj} - n_j(1 - x_{uj}) - \frac{p_j + n_j}{4} \tag{3.7}$$

Damit sind die LP-Kosten, die wir mit j assoziieren, nach unten durch eine lineare Funktion begrenzt. Diese liegt zwischen $p_j/8 + 3n_j/8$ für $x_{uj} = 3/8$ und $p_j/4 + n_j/4$ für $x_{uj} = 1/2$. Die mit j assozierten Kosten des Algorithmuses belaufen sich auf n_j. Somit kann man im Falle des vollständigen Graphen auch hier eine Faktor 4 Approximation garantieren.

In Abbildung 3.5 ist das Bild eines allgemeinen Graphen für den oben beschrieben Fall dargestellt. Dabei sind die negativen Kante rot und die positiven Kanten grün dargestellt. Die schwarzen Kanten geben nur die Werte der Variablen an, sind aber keine Kanten des ursprünglichen Graphen. In diesem Beispiel sieht man, dass die mit j assozierten LP-Kosten 0.2 betragen. Die mit j assozierten Kosten der Lösung betragen alldrings 1, da eine Kante falsch klassifizeirt wurde. Somit liefert der Algorithmus in diesem Fall keine 4-Approximation mehr.

Für die positiven Kanten zu Knoten außerhalb des Clusters kann man eine ähnliche Betrachtung machen und auch in diesem Fall kann gezeigt werden, dass es sich bei dem Algorithmus um eine Faktor 4 Approximation handelt. Somit ist nun bewiesen, dass der Algorithmus für den Fall, dass es sich bei

Abbildung 3.6: Algorithmus für MinDisAgree auf allgemeinen Graphen

1. $C \leftarrow \varnothing$ /*Menge Cluster*/
2. Solange i, j existiert mit $x_{ij} > 2/3$
   (a) $S = B_x(i, r)$ für $r > 1/3$
   (b) $C = C \cup S$
   (c) Entferne S und alle Kante zu S aus dem Graphen
3. Return C

dem Graphen um einen vollständigen handelt, eine 4-Approximation der optimalen Lösung liefert.

3.4 Approximationsgrenzen der ILP-Formulierung

Für vollständige Graphen kann man sich nun noch überlegen, ob man durch andere Rundungsstrategien mit der gleichen ILP-Formulierung noch bessere Approximationen erhalten kann. Dabei kann gezeigt werden, dass die Ganzzahligkeitslücke der LP-Formulierung 2 beträgt und man somit keinen Rundungsalgorithmus formulieren kann, der mehr als eine 2-Approximation liefert.
Außerdem wird in der Veröffentlichung gezeigt, das mittels der hier angewendeten Region Growing Technik maximal eine 3-Approximation erreicht werden kann. Wenn man die Rundungsalgorithmen weiter einschränkt, so kann gezeigt werden, dass ein Rundungsalgorithmus mit k Schwellwerten maximal eine $3 + 1/k$ Approximation erreicht werden kann. Mit dem hier gezeigten Ansatz, der nur einen Schwellwert benutzt, kann somit auch keine bessere Approximation erreicht werden.
Für den Fall von allgemeinen Graphen könnnen wir gar keinen Approximationsalgorithmus angeben, der eine konstante Approximation liefert, da in diesem Fall die Ganzzahligkeitslücke bereits in $O(\log n)$ liegt. Somit ist der im nächsten Abschnitt angegebene Algorithmus optimal unter der Voraussetzung, dass man diese LP-Formulierung benutzt.

3.5 Der Algorithmus für allgemeine Graphen

Der beschriebene Algorithmus benutzt die optimale Lösung der relaxierten Version des obene beschrieben LPs, um daraus eine gute Lösung zu konstruieren. Dabei funktionert der Algorithmus wie die GVY region growing Technik. Die Werte der Lösung des LP werden dabei, wie vorher bereits erwähnt, als Distanz zwischen den Knoten interpretiert.
Für den Algorithmus müssen noch zwei Mengen definiert werden. Zuerst $B_x(i, r) = \{j : x_{ij} \leq r\}$, die Menge aller Punkte j deren Abstand zum

Punkt i kleiner gleich r ist. Außerdem wird mit $\delta(S)$ die Menge aller Kanten von der Knotenmenge S zur Komplementmenge $\overline{S} = V \backslash S$ bezeichnet.
Der Algorithmus ist in Pseudocode in Abbildung 3.5 dargestellt. Für den darin benutzten Radius r wird in der Analyse im Paper gezeigt, dass es einen Radius r gibt, so dass ein Approximationsfaktor von $O(\log n)$ ereicht wird. Bei der Analyse des Algorithmuses werden dabei wieder die LP Kosten der Kanten mit den Kosten dieser Kanten in der Lösung des Algorithmuses verglichen.

Kapitel 4

Komplexität der Approximation

In diesem Kaptiel betrachten wir die Komplexität der Probleme. Dabei werden wir uns genauer den Beweis zur Grenze der Approximierbarkeit des MaxAgree Problems betrachten. Danach werden wir noch einmal den Zusammenhang zwischen den Problemen betrachten. Zum Schluss werden noch kurz die anderen Ergebnisse im Bereich der Komplexität der Approximatierbarkeit des Papers betrachtet.

4.1 MaxAgree auf allgemeinen Graphen

In diesem Abschnitt werden wir mittels einer Reduktion des Max3Sat Problems zeigen, dass es für alle $\epsilon > 0$ NP-hart ist, die gewichtete Version des MaxAgree-Problems auf allgemeinen Graphen mit einem Faktor von $79/80 + \epsilon$ zu approximieren.
Mittels einer sehr ähnlichen Reduktion des gleichen Problems lässt sich dann zeigen, dass es für aller $\epsilon > 0$ NP-hart ist, die ungewichtete Verion des MaxAgree-Problems mit einem Faktor von $115/116 + \epsilon$ zu approximieren.

4.1.1 Max3Sat

Bei dem Max3Sat-Problem sind eine Menge von Klauseln mit jeweils genau 3 boolschen Variablen gegeben. Das Probelm besteht nun darin, eine Variablenbelegung zu finden, die eine maximale Anzahl von Klauseln erfüllt. Es ist bekannt, dass es für alle $\epsilon > 0$ NP-hart ist, dieses Problem mit eine Faktor von $7/8 + \epsilon$ zu approximieren [2].
In unserem Fall wollen wir das Problem noch etwas weiter einschänken, in dem wir verlangen, dass die Literale x_i und $\overline{x_i}$ in gleich vielen Klauseln vorkommen. Dabei sei B_i die Anzahl der Formeln. Diese Einschränkung ändert

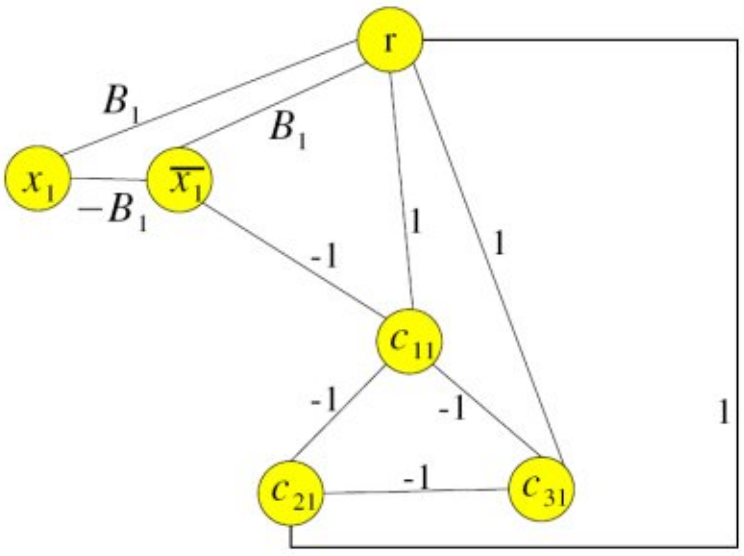

Abbildung 4.1: Beispielkonstruktion

jedoch nicht die Komlexität der Approximation.

4.1.2 Transformation

Für die Reduktion muss nun eine beliebige Instanz des Max3Sat-Problems zu einer Instanz des MaxAgreeProblems transformiert werden. Die Hauptaufage besteht dabei darin, einen Graphen für das MaxAgree-Problem zu konstruieren. Dafür wird ein Graph mit 3 Arten von Knoten erstellt. Zuerst gibt es einen Wurzelknoten r. Außerdem gibt es für jede Variable zwei Knoten x_i und $\overline{x_i}$ sowie drei Knoten c_{1j}, c_{2j}, c_{3j} für jede Klausel C_j.
Der Wurzelknoten ist zu jedem Klauselknoten durch die Kante (r, c_{ij}) mit Gewicht 1 sowie zu jedem Varibalenknoten durch eine Kante mit Gewicht B_i verbunden. Außerdem sind die Knoten x_i und $\overline{x_i}$ durch eine Kante mit Gewicht $-B_i$ verbunden. Die Knoten einer Klausel bilden ein Dreieck, wobei die Kanten jeweils ein Gewicht von minus eins haben. Außerdem sind die Knoten x_i und c_{jk} genau dann verbunden, wenn das Literal x_i an j-ter Stelle in Klausel C_k vorkommt. Das Gewicht dieser Kante beträgt -1. Ein Beispiel für eine solche Konstruktion ist in Abbildung 4.1 dargestellt.

4.1.3 Rücktransformation

Für die Reduktion des Max3Sat-Problems wird außerdem noch eine Rücktransformation des Ergebnisses benötigt. Dabei muss die Lösung des MaxAgree-Algorithmuses auf dem oben konstruierten Graphen in eine Lösung

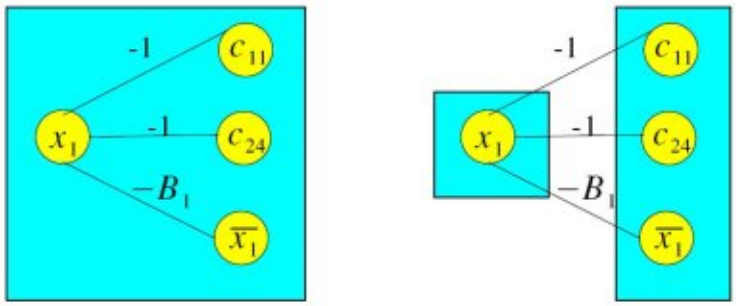

Abbildung 4.2: Beide Variablenknoten in einem Cluster

für das ursprüngliche Problem umgewandelt werden. Dazu wird eine Variable auf wahr gesetzt, genau dann wenn sie in einem Singelton Cluster liegt. Dies ist genau dann der Fall, wenn sie nicht mit r in einem Cluster ist.
Zunächst muss gezeigt werden, dass es für das Cluster eines Variablenknoten nur genau diese beiden Möglichenkeiten gibt. Dazu nehmen wir an, dass der Variablenknoten x_i in einem Cluster sei, in dem nicht der Knoten r ist etwa wie in Abbildung 4.2 gegeben. Da die Kante (x_i, r) die einzige zu x_i inzidente Kante mit positiven Label ist, ist x_i nur über negative Kanten innerhalb des Clusters verbunden. Daher verringert sich der Wert der Lösung nicht, wenn wir x_i, wie in Abbildung 4.2 gezeigt, in ein Singelton-Cluster gruppieren. Somit kann jede Clusterung des Graphen einfach in eine andere transformiert werden, für die die Bedingung gilt und dessen Wert mindestens genauso groß ist.
Damit obige Definition der Variablen gültig ist, muss außerdem gelten, dass genau einer der Variablenknoten x_i und $\overline{x_i}$ im selben Cluster wie r ist.
Dazu betrachten wir zunächst den Fall, dass beide Knoten im selben Cluster wie r sind. In diesem Fall können wir den Knoten x_i in ein Singelton-Cluster gruppieren ohne den Wert der Lösung zu verringern. Nun ist zwar die Kante (x_i, r) falsch klassifiziert, dafür aber die Kante $(x_i, \overline{x_i})$ richtig. Da beide das Gewicht B_i haben, ändert sich der Wert der Lösung durch diese Kanten nicht. Alle anderen zu x_i inzidenten Kanten haben negatives Gewicht, so dass sich durch die Änderung der Clusterung der Wert der Lösung höchstens vergrößern kann (s. Abb. 4.3).
Außerdem muss der Fall verhindert werden, dass kein Knoten im selben Cluster wie r ist. Nehmen wir dazu an, dass beide Knoten x_i und $\overline{x_i}$ in einem Singelton-Cluster sind. In diesem Fall können wir x_i in das Cluster von r gruppieren ohne den Wert der Lösung zu verringern. Darduch ist die Kante (x_i, r) nun richtig klassifiziert. Dafür sind maximal alle B_i Kanten von x_i zu Klauselknoten nun falsch klassifiziert. Da die Kante (x_i, r) jedoch ein Gewicht von B_i hat kann sich der Wert der Lösung durch die Änderung nur verbessern.

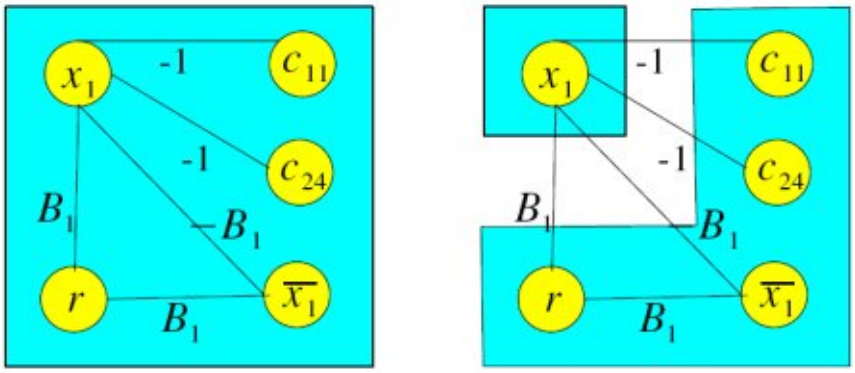

Abbildung 4.3: Knoten im selben Cluster

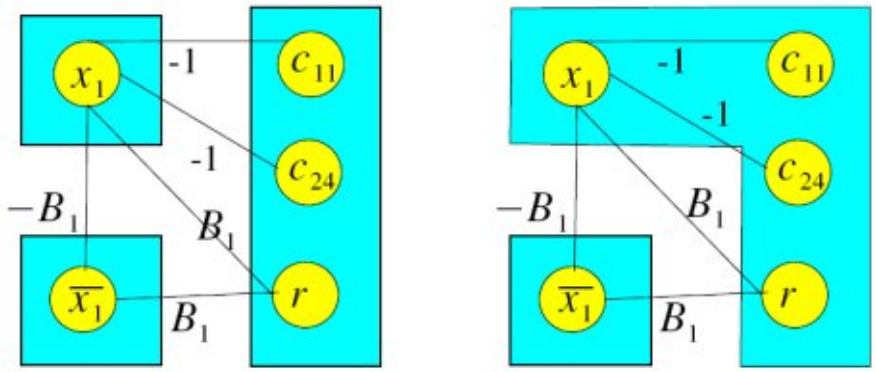

Abbildung 4.4: Beide Knoten in Singelton Cluster

Somit lässt sich eine beliebige Clusterung des Graphen leicht in eine transformieren, deren Wert mindestens genau so groß ist und aus dem sich einfach die Variablenbelegung für das Max3Sat-Problem ablesen lässt.
Auch die Clusterung der Klauselknoten muss unter Umständen geändert werden, um die Analyse des Wertes der Lösung zu vereinfachen. Diese Veränderungen sind wieder einfach durchzuführen und verringern den Wert der Lösung nicht. Betrachtet man zunächst die Klauselknoten zu nicht erfüllten Klauseln. Da sich nach obiger Überlegung alle inzidenten Variablenknoten im Cluster mit r befinden, können alle Klauselknoten in eine Singeltoncluster gruppiert werden ohne den Wert der Lösung zu verringern.
Falls eine Klausel erfüllt ist, so ist mindestens eine der inzidenten Literale wahr und somit in einem Singelton-Cluster, zum Beispiel die erste. Dann kann der Knoten c_{1j} in das Cluster mit r gruppiert werden und die anderen beiden in ein Signelton-Cluster, ohne den Wert der Lösung zu verringern.
Durch die Transformation auf den Variablen- und Klauselknoten haben wir jetzt einen Graphen erhalten, auf dem wir die Analyse machen können.

4.1.4 Analyse

Betrachtet man zunächst alle Kanten, die inizident zu Vaiablenknoten sind, so ergibt sich für diese folgendes Gewicht der Lösung. Da beide Variablenknoten in verschiedenen Clustern sind, sind die Kanten zwischen den Variablenknoten richtig klassifizeirt. Diese haben ein Gesamtgewicht von $\sum_{i=1}^{n} B_i$. Außerdem ist für jede Variable genau ein Knoten im selben Cluster wie r. Die Kanten dazwischen haben auch ein Gewicht von $\sum_{i=1}^{n} B_i$. Die zu den Variabeln inzidenten Klauselknoten sind immer in einem anderen Cluster. Somit sind alle Kanten zwischen Variableknoten und Klauselknoten richtig klassifiziert. Das ergibt eine Gesamtgewicht von $3m$. Außerdem sind alle Knoten einer Klausel in verschiedenen Clustern. Das Gewicht aller Kanten zwischen Klauselknoten ist $3m$. Falls m' Klausel erfüllt, so sind genau m' Klauselknoten im selben Cluster wie r. Dies führt zu weiteren richtigen klassifizeirten Kanten mit einem Gewicht von m'. Somit ergibt sich folgende Beziehung zwischen der Lösung des MaxAgree-Problems und der Lösung von Max3Sat:

$$L_{MA} = \sum_{i=1}^{n} 2B_i + 6m + m' = 9m + m' = 9m + L_{M3Sat} \tag{4.1}$$

Es ist bekannt, dass es NP-hart ist zu unterscheiden, ob für die Lösung von Max3Sat $OPT_{M3Sat} = m$ oder $OPT_{M3Sat} \leq (7/8 + \epsilon)m$ gibt. Somit ist es nach obiger Überlegung auch NP-hart zu unterscheiden, ob für die Lösung von MaxAgree $OPT_{MA} = 10m$ oder $OPT_{MA} \leq (9 + 7/8 + \epsilon)m$ gibt. Falls $NP \neq P$ kann es somit keinen polyionomialen Approximationsalgorithmus mit einem Approximationsfaktor von $\frac{(9+7/8+\epsilon)m}{10m} = 79/80 + \epsilon'$ geben.
Für die ungewichtete Version des Problems kann man eine analoge Transformation vornehemen. Man muss hierbei nur Kanten mit einem Gewicht von w durch w ungewichtete Wege ersetzten. Da bei der Konstruktion nur ganzzahlige Gewichte benutzt werden ist dies auch möglich. Damit ergibt sich eine Grenze für die polynomiale Approximierbarkeit von $\frac{115}{116} + \epsilon$.

4.1.5 Folgerungen für MinDisAgree

Mittels des Zusammenhangs zwischen MinDisAgree und MaxAgree führt das Ergebnis des letzten Abschnittes auch zu Aussagen über die Komplexität von MinDisAgree. Der Zusammanhang zwischen den beiden Problemen besteht darin, dass eine Lösung des MaxAgree Problems mit Wert L_{MA} auch eine Lösung des MinDisAgree Problems liefert, deren Wert $L_{MD} = W - L_{MA}$ ist. Dabei ist W die Summe aller Kantengewichte.
Bei der oben beschrieben Konstuktion ergibt sich ein Gesamtgewicht des Graphen von $\frac{27}{2}m$. Daraus resultiert, dass es NP-hart zu unterscheiden ist, ob für die optimiale Lösung $OPT_{MD} = \frac{27}{2}m - 10m$ oder $OPT_{MD} = \frac{27}{2}m - (9 + 7/8 + \epsilon)m$ gibt. Hierraus ergibt sich dann, dass es keinen polynomialen

Approximationsalgorithmus geben kann, falls $NP \neq P$, der MinDisAgree auf allgemeinen Graphen mit einem Faktor von $\frac{29}{28} - \epsilon'$ approximiert.

4.2 Andere Probleme

In dem Paper [3] wurde außerdem eine Reduktion von Minimum Multicut auf MinDisAgree auf allgemeinen Graphen durchgeführt. Darduch konnte gezeigt werden, das diese beiden Probleme gleich schwer sind. Es ist ein offenes Problem, ob Minimum Multicut mit einem konstanten Faktor approximiert werden kann.
Außerdem wurde eine Reduktion von Max 2-colorable subgraph auf MinDisAgree auf vollständigen Graphen durchgeführt. Daraus folgt die Existenz eines $a > 1$, für dass es NP-hart ist, MinDisAgree auf vollständigen Graphen mit dem Faktor a zu approximieren.

Kapitel 5

Algorithmen MaxAgree

Wie schon im ersten Kapiel beschrieben, versucht man beim MaxAgree-Problem nicht die Anzahl der falsch klassifizirten Kanten zu minimieren, sondern die Zahl der richtig klassifizierten Kanten zu maximieren. Dies ist zwar ein sehr ähnliches Problem, allerdings kann man bei diesem Problem keine gute Approximation durch eine Herrangehensweise wie im letzt Kapitel erreichen. Deshalb haben die Autoren des Papers [3] das Problem als ein Semi-definites Programm (SDP) beschrieben. Die genaue Formulierung dieses Problems wird im nächsten Abschnitt beschrieben.
Anschließend wird der von den Autoren auf der Lösung dieses SDP basierende Algorithmus in Abschnitt 5.2 vorgestellt und analysiert. Im letzten Abschnitt des Kapitels werden noch fast-erfüllte Beispiele behandelt und es wird insbesondere auf den Zusammenhang zwischen MaxAgree und MinDisAgree eigegangen.

5.1 Semi-definite Programme

Semi-definite Prgramme sind eine weitere Möglichkeit um Optimierungsprobleme zu lösen. Sie können ähnlich wie ILPs zur Approximation komplexer Probleme benutzt werden. Dazu wird zunächst eine Relaxierung des Problems als SDP formuliert. Dieses kann dann in polynomialer Zeit gelöst werden. Die Lösung wiederrum kann zu einer approximativen Lösung des ursprünglichen Problems gerundet werden.

5.1.1 Motivation

Zur Motivation dieses Verfahrens betrachten wir eine Lösung des Problems und ordnen jedem Cluster einen Einheitsvektor zu. Jedem Konten wird dann auch der Einheitsvektor seines Clusters zugeordnet. Damit lässt sich nun einfach beschreiben, ob zwei Konten im selben Cluster sind, da gilt:
$v_i * v_j = 1 \Leftrightarrow i$ und j im selben Cluster

5.1.2 Formulierung

Betrachtet man die zu dem ILP aus Abschnitt 3.1 analoge Formulierung für das MaxAgree Problem so ergibt sich folgende Maximierungsfunktion:

$$max(\sum_{+(ij)} w_{ij}(v_i * v_j) + \sum_{-(ij)} w_{ij}(1 - v_i * v_j)) \tag{5.1}$$

Bezeichne P die Matrix, deren Reihen die Vektoren der Punkte sind, so lässt sich diese Maximierungsfunktion zu folgender Funktion umschreiben:

$$maxC * (PP^T) = C * X$$

Die Relaxierung des Problems besteht nun darin, dass wir nicht verlangen, dass X die Einheitsmatrix sein muss, sonders dass es eine beliebige semi-definite Matrix sein kann. Da sie semi-definit ist, gibt es eine Matrix P, so dass $PP^T = X$ ist. Diese Matrix P kann dann benutzt werden, um eine Lösung für das ursprüngliche Probelm zu erhalten.
Vergleicht man dieses Vorgehen mit dem ILP beim MinDisAgree-Problem, so sieht man, dass man wieder ein Funktion hat, die optimiert wird. Allerdings entsprechen beim SDP die Variablen den Knoten und nicht wie beim ILP den Kanten.

5.2 Algortihmus

Das Rundungsschema H_t wählt zufällig t Hyperebenen aus. Diese teilen den Raum der Vektoren in 2^t Teilräume, die jeweils einem Cluster entsprechen. Die Lösung des Semi-definiten Programms liefert für jeden Knoten einen Vektor. Der Knoten wir jetzt in das Cluster eingeteilt, das dem Teilraum entspricht, in dem der zugehörige Vektor liegt.
Der Algorithmus besteht nun darin, sowohl H_2 als auch H_3 zu berechnen und die bessere von beiden Lösungen zu wählen. Wir werden ihn im weiteren mit $Best(H_2, H_3)$ bezeichnen.
Im nächsten Kapitel werden wir zeigen, dass der Erwartungswert für den Approximationsfaktor bei diesem Algorithmus bei 0.766 liegt.

5.3 Analyse des Approximationsfaktors

Der Algorithmus $Comb(H_2, H_3)$ wählt mit einer Wahrscheinlichkeit von $1-\alpha$ die Lösung von H_2 und mit einer Warscheinlichkeit von α die Lösung von H_3. Somit gilt auf jeden Fall, dass die Lösung von $Best(H_2, H_3)$ besser ist, als die Lösung dieses Algorithmuses. Im weiteren werden wir zeigen, dass der

Algorithmus $Comb(H_2, H_3)$ die Lösung mit einem Faktor von 0.7664 approximiert, was damit auch für unseren eigentlichen Algorithmus gilt.
Bei der Analyse des Algorithmus wird dabei für jede Kante der erwartete Beitrag zur Lösung relativ zum Beitrag zur Lösung des SDP betrachtete. Falls dieser Faktor für alle Kanten kleiner als 0.7664 ist, gilt das somit auch für das ganze Problem. Da die Lösung des Semi-definiten Programms größer ist als die Lösung des ursprünglichen Problems, ist somit auch der Approximationsfaktor kleiner als 0.7664.
Betrachtet man nun die Knoten i und j, zwischen deren Vektoren ein Winkel von θ liegt, so befinden sich die beiden Knoten mit einer Wahrscheinlichkeit von $(1 - \frac{\theta}{\pi})^t$ im selben Cluster.
Zuerst werden wir den Fall behandeln, dass die Kante (ij) eine positive Markierung hat. Laut der Formel 5.1 ist der Beitrag der Kante zur Lösung des SDP dann $v_i * v_j = cos(\theta)$. Außerdem werden die beiden Endknoten i und j nach den vorherigen Überlegungen vom Algorithmus H_2 mit einer Wahrscheinlichkeit von $(1 - \frac{\theta}{\pi})^2$ und vom Algorithmus H_3 mit einer Wahrscheinlichkeit von $(1 - \frac{\theta}{\pi})^3$ in das gleiche Cluster gruppiert. Der Algorithmus $Comb(H_2, H_3)$ gruppiert die beiden Knoten somit mit folgender Wahrscheinlichkeit in dasselbe Cluster:

$$(1 - \alpha) * (1 - \frac{\theta}{\pi})^2 + \alpha * (1 - \frac{\theta}{\pi})^3$$

Falls beide Knoten in dasselbe Cluster eingeteilt werden, wurde die Kante ij richtig klassifiziert und trägt somit mit einem Wert von 1 zur Lösung bei. Im anderen Fall wird die Kante falsch klassifiziert und hat somit keinen Anteil an der Lösung. Somit ist der erwartete Beitrag der Kante ij gerade gleich der obigen Wahrscheinlichkeit.
Eine analoge Analyse kann man auch für den Fall durchführen, dass die Kante negatives Gewicht hat und erhält dann einen Beitrag A_{SDP} der Kante zur Lösung des SDP, sowie einen Beitrag $A_L(\theta)$ zur Lösung des Algorithmus.
Somit ist der Approximationsfaktor durch den Ausdruck in Formel 5.2 beschränkt.

$$min_{\theta \in (0,\pi/2)} \left(\frac{(1 - \alpha) * \left(1 - \frac{\theta}{\pi}\right)^2 + \alpha * \left(1 - \frac{\theta}{\pi}\right)^3}{cos(\theta)}, \frac{A_L(\theta)}{A_{SDP}} \right) \tag{5.2}$$

Es kann gezeigt werden, dass dieses Minimum maxiamal für den Wert $\alpha = 0.1316$ ist und dann ein Approximationsfaktor von 0.7664 erreicht wird.

5.4 Fast erfüllte Beispiele

Fast erfüllte Beispiele sind Graphen bei denen die Lösung des SDP fast der maximal möglichen Lösung W, der Summe aller Gewichte, entspricht. Formal

bedeutet dies, dass gilt:

$$OPT_{SDP} = (1 - \epsilon)W$$

In der Veröffentlichung von [3] wird gezeigt, dass der Algorithmus $H_{log(1/\epsilon)}$ dann einen Approximationsfator von $(1 - O(\sqrt{\epsilon}log(1/\epsilon)))W$ garantiert.
Wie bereits im Abschnitt über die Komplexität des MinDisAgree-Problems erwähnt, gibt es folgenden Zusammenhang zwischen den Problemen, wobei W das Gesamtgewicht aller Kanten des Graphen ist.

$$L_{MD} = W - L_{MA} \tag{5.3}$$

Falls wir somit ein fast-erfüllte Beispiel als MinDisAgree-Problem betrachten ergibt sich mittels des Zusammenhangs, dass das Semi-definite Programm eine Lösung $OPT_{SDP} = \epsilon W$ liefert. Falls wir den obigen Algorithmus auf das Probelm anwenden, so erhalten wir eine Lösung $ALGO_{MD} = W - ALGO_{MA} = O(\sqrt{\epsilon}log(1/\epsilon))W$.

$$\frac{ALGO_{MD}}{OPT_{MD}} = \frac{O\left(\sqrt{\epsilon}log\left(\frac{1}{\epsilon}\right)\right)W}{\epsilon W} = O\left(\frac{1}{\sqrt{\epsilon}}log\left(\frac{1}{\epsilon}\right)\right)W \tag{5.4}$$

In Formel 5.4 ist der Approximationsfaktor des Algorithmuses $H_{log(1/\epsilon)}$ für das MinDisAgree Problem zu sehen. Daraus ergibt sich, dass der Algorithmus für jedes feste Epsilon einen konstanten Approximationsfaktor hat. Dies lässt die Vermutung zu, dass es schwierig sein wird eine super-konstante Inapproximierbarkeit des Problems zu zeigen.

Kapitel 6

Zusammenfassung

In der Ausarbeitung wurde gezeigt, dass MinDisAgree auf allgemeinen Graphen unter allen hier betrachteten Problemen das am schwersten zu approximierende ist. Außerdem gibt es hier noch die große Differenz zwischen der bewiesenen unteren Schanke für die Approximierbarkeit und der Preformance der bekannten Algorithmen.
Es wurde in der Ausarbeitung dargelegt, wieso der Algorithmus für vollständige Graphen nicht einfach auf allgemeine angewendet werden kann. Außerdem wurde gezeigt, dass man über das betrachtete Lineare Programm keine Verbesserung erziehlen kann. Desweiteren wurde gezeigt, dass man das Problem höchstens mit einem konstanten Faktor approximieren kann. Ob dies wirklich möglich ist, konnte zwar nicht gezeigt werden, allerdings wurde bewiesen, dass es zumindest genauso schwierig ist wie Minimum Multicut. Außerdem wurde mittels der fast-erfüllte Beispiele auch gezeigt, dass es schwierig ist, eine super-konstante Inapproximiermarkeit zu zeigen.

Literaturverzeichnis

[1] E.Demiane and N.Immorilca. Correlation clustering with partial information. In *Proc. of 6th APPROX*, 2003.

[2] J.Hastad. Some optimal inapproximability results. In *JACM*, 2001.

[3] V.Guruswami M.Charikar and A.Wirth. Clustering with qualitative information. In *Proc. of 44th FOCS*, 2003.

[4] A.Blum N.Bansal and S.Chawla. Correlation clustering. In *Proc. of 43rd FOCS*, 2002.

[5] V.Vazirani N.Garg and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. In *SIAM J Comp*, 1996.