Studienarbeit: Clustern mit beschränkter Clustergröße

von
Jochen Speck

Betreuer: Prof. Dr. Dorothea Wagner, Dr. Alexander Wolff, Marco Gaertler

ITI Wagner, Fakultät für Informatik, Universität Karlsruhe

Dezember 2006

Zusammenfassung

„Correlation Clustering“ ist das Problem, die Knoten eines Graphen zu partitionieren, dessen gewichtete Kanten mit + oder − beschriftet sind, wobei das Gewicht der durch die Partitionierung unzerschnittenen + Kanten und der zerschnittenen − Kanten maximiert werden soll. Ein solches Problem tritt z. B. auf, wenn Mitarbeiter in Gruppen eingeteilt werden sollen und deren Sympathien und Abneigungen untereinander bekannt sind. Die vorliegende Arbeit untersucht dieses Problem für den Fall, dass zusätzlich die maximale Anzahl von Knoten in einer Gruppe der Partition durch eine Konstante beschränkt ist. Dabei wird gezeigt, dass das „Clustern mit beschränkter Clustergröße“ NP-hart ist. Weiter werden zwei Approximationsalgorithmen angegeben und ihre Approximationsgüte gezeigt. Danach wird mit einer statistischen Untersuchung gezeigt, dass die Approximation in vielen Fällen besser ist als die bewiesenen Resultate.

Hiermit versichere ich, die Arbeit selbstständig angefertigt, alle Hilfsmittel und alles, was aus Arbeiten anderer entnommen wurde, angegeben oder kenntlich gemacht zu haben.

Unterschrift: .....

Inhaltsverzeichnis

1 Einführung **3**
1.1 Bekannte Ergebnisse aus verwandten Bereichen 3
1.2 Eigene Ergebnisse 4
1.3 Aufbau der Arbeit 4

2 Problembeschreibung **4**
2.1 Definitionen und Notationen 4
2.2 Formulierung als ILP 5

3 Theoretische Untersuchung **6**
3.1 Clustern mit beschränkter Clustergröße ist NP-hart 6
3.2 Faktor k Approximation von MaxAgree 8
3.3 Faktor $k^2/2$ Approximation von MinDisagree 10

4 Statistische Untersuchung **13**
4.1 Beschreibung des Versuchsaufbaus 13
4.2 Messungen 14

5 Zusammenfassung und Ausblick **16**

1 Einführung

In dieser Arbeit wird „Correlation Clustering“, wie in [3] beschrieben, mit weiteren Einschränkungen untersucht. Ein solches Problem tritt zum Beispiel auf, wenn man eine größere Gruppe von Mitarbeitern in kleinere Gruppen aufteilen möchte, weiter seien die Sympathien oder wer mit wem wie gut zusammenarbeitet bekannt, oder die Mitarbeiter geben in einem Fragebogen an, mit wem sie zusammenarbeiten möchten und mit wem nicht (gewichtet). Das Problem kann auch auftreten, wenn man eine Menge von Texten nach gemeinsamen Themen in Gruppen einteilen möchte, und dabei die Themenübereinstimmungen oder Unterschiede zwischen den einzelnen Texten bekannt sind.
Im ursprünglichen Problem besteht die Problemeingabe aus einer Menge von n Objekten und einer Menge von gewichteten, positiven oder negativen Beziehungen zwischen Paaren dieser Objekte. Die Eingabe lässt sich äquivalent als Graph mit gewichteten Kanten und n Knoten, dessen Kanten zusätzlich mit $+$ oder $-$ beschriftet sind, beschreiben. Beim „Correlation Clustering“ soll die Menge der Objekte nun so partitioniert werden, dass zwischen Objekten, die zu verschiedenen Gruppen der Partition gehören, möglichst wenig Gewicht der positiven Beziehungen liegt, und zwischen Objekten, die in einer gemeinsamen Gruppe liegen, möglichst wenig Gewicht der negativen Beziehungen liegt. Dieses Ziel ist äquivalent zu der Aufgabe, dass zwischen Objekten, die in einer gemeinsamen Gruppe liegen, möglichst viel Gewicht der positiven Beziehungen liegt, und zwischen Objekten, die zu verschiedenen Gruppen der Partition gehören, möglichst viel Gewicht der negativen Beziehungen liegt. Dabei werden die positiven Beziehungen innerhalb einer Gruppe der Partition und die negativen Beziehungen zwischen Objekten zweier verschiedener Gruppen als Übereinstimmungen bezeichnet. Die negativen Beziehungen innerhalb einer Gruppe und die positiven Beziehungen zwischen Objekten zweier verschiedener Gruppen werden als Fehler bezeichnet.

Bei den obigen Beispielen ist in vielen Fällen durch eine Ressourcenbeschränkung (Raumgröße, Fahrzeuggröße, Listenlänge einer Themenübersicht, usw.) oder andere Dinge (maximal erwünschter Kommunikationsaufwand, usw.) die maximale Gruppengröße vorgegeben. Das Finden einer Partitionierung, die die Übereinstimmungen maximiert oder die Fehler minimiert, und die gleichzeitig die Ressourcenbeschränkung einhält, führt zum folgenden Problem.

„Correlation Clustering“ wird zusätzlich um eine Größenschranke k erweitert, die die maximale Anzahl von Objekten innerhalb einer Gruppe der Partition festlegt. Diese Größenschranke ist wesentlich kleiner als die Gesamtzahl der gegebenen Objekte. Dieses Problem wird im Folgenden als „Clustern mit beschränkter Clustergröße“ bezeichnet.

1.1 Bekannte Ergebnisse aus verwandten Bereichen

Das gleiche Problem ohne die Beschränkung der Clustergröße wurde in [3] untersucht. Dabei wurde gezeigt, dass das Problem der Minimierung des Gesamtgewichts der Fehler APX-hart ist, und ein $O(log\ n)$ Approximationsalgorithmus für das Problem vorgestellt.

In [1] wird das Problem für ungewichtete vollständige Graphen ohne Beschränk-

ung der Clustergröße untersucht. Dabei wurde für die Maximierung der Anzahl der Übereinstimmungen ein PTAS angegeben, und ein Approximationsalgorithmus mit konstantem Approximationsfaktor für das Problem der Minimierung der Anzahl der Fehler vorgestellt.

Die Ergebnisse aus [1] werden in [2] unter anderem um Ergebnisse zum „Correlation Clustering“ auf allgemeinen ungewichteten Graphen ohne Beschränkung der Clustergröße erweitert. Dabei wird gezeigt, dass sowohl das Problem der Approximation der maximalen Anzahl der Übereinstimmungen als auch der minimalen Anzahl der Fehler auf allgemeinen Graphen APX-hart ist. Weiter werden ein $O(log\ n)$ Approximationsalgorithmus für das Minimieren der Anzahl der Fehler und ein Approximationsalgorithmus mit konstantem Approximationsfaktor für das Problem der Maximierung der Anzahl der Übereinstimmungen vorgestellt.

1.2 Eigene Ergebnisse

In dieser Arbeit wird gezeigt, dass das Problem des Clusterns mit beschränkter Clustergröße NP-hart ist. Zusätzlich werden zwei Approximationsalgorithmen vorgestellt. Der erste approximiert das maximale Gewicht der Übereinstimmungen mit einem Approximationsfaktor von k, der zweite approximiert das minimale Gewicht der Fehler mit einem Approximationsfaktor von $\frac{k^2}{2}$. Weiter werden die Algorithmen mit zufälligen Eingaben getestet, um zu zeigen, dass beiden Algorithmen die Approximation des Optimums wesentlich besser gelingt, als die gezeigten Approximationsfaktoren dies vermuten lassen.

1.3 Aufbau der Arbeit

Der weitere Aufbau der Arbeit sieht wie folgt aus: In Abschnitt 2 wird das Problem formal beschrieben und eine Formulierung als ILP angegeben. In Abschnitt 3 wird als Erstes gezeigt, dass das Problem NP-hart ist, und dann werden zwei Approximationsalgorithmen mit polynomialer Laufzeit angegeben, so dass gezeigt wird, dass beide Bewertungen eine Approximation mit konstantem Faktor zulassen. In Abschnitt 4 wird das Approximationsverhalten der Algorithmen für zufällige Eingaben untersucht. Dabei ist zu erkennen, dass die Approximation durch die beiden Algorithmen im Schnitt wesentlich besser ist als die Ergebnisse aus Abschnitt 3. Abschnitt 5 enthält eine Zusammenfassung der Ergebnisse und eine Liste offener Probleme.

2 Problembeschreibung

In diesem Abschnitt wird das Problem formal beschrieben, und eine Formulierung des Problems als ILP vorgestellt

2.1 Definitionen und Notationen

Sei $G = (V, E)$ ein einfacher schleifenfreier Graph mit n Knoten. Jede Kante $e \in E$ habe ein Label ($+$ oder $-$) und ein Gewicht $\omega(e) \in \mathbb{R}^+$. Die Kanten mit

Label + werden im Weiteren als Pluskanten (E^+), die Kanten mit Label − als Minuskanten (E^-) bezeichnet.

Eine Clusterung $\$$ eines solchen Graphen ist eine Partitionierung $(S_1, \ldots, S_t)$ der Knoten. Eine Pluskante ist dabei richtig geclustert, falls ihre beiden Endknoten in derselben Gruppe der Partition liegen, sonst ist sie falsch geclustert. Eine Minuskante ist richtig geclustert, falls ihre beiden Endknoten in verschiedenen Gruppen liegen, und sonst falsch geclustert. Dabei werden hier zwei verschiedene Ziele betrachtet:

1. Maximiere das Gesamtgewicht der richtig geclusterten Kanten: MaxAgree
Die Bewertung für MaxAgree ist damit:

$$\omega_{ma}(\$) = \sum_{e \in E^+ \cup E^-} 1_{(\text{e richtig geclustert})} \cdot \omega(e)$$

2. Minimiere das Gesamtgewicht der falsch geclusterten Kanten: MinDisagree
Die Bewertung für MinDisagree ist damit:

$$\omega_{md}(\$) = \sum_{e \in E^+ \cup E^-} 1_{(\text{e falsch geclustert})} \cdot \omega(e)$$

Die Optimallösungen für diese beiden Ziele sind zwar identisch, aber eine Lösung approximiert die beiden Ziele im Allgemeinen verschieden gut. Deshalb wird hier für jedes Ziel eine andere Technik verwendet.

Beim Suchen dieser Optima ist als Einschränkung zu beachten, dass hier ein $k \in \mathbb{N},\ k \geq 3$ existiert, mit $|S_i| \leq k\ \forall i \in \{1, \ldots, t\}$ d. h. jeder Cluster enthält maximal k Knoten. Dabei sei k gedacht als kleine Konstante, d. h. $k \ll n$.

2.2 Formulierung als ILP

Dieses Problem läßt sich als ILP formulieren:
Wähle für jedes Knotenpaar u, v eine Variable $x_{uv} \in \{0, 1\}$ mit $x_{uv} = 1$, falls u und v im selben Cluster liegen, und $x_{uv} = 0$ sonst. Dabei gelten für beide Ziele dieselben Nebenbedingungen:
Die Relation $R = \{(u, v) \in V \times V | x_{uv} = 1\}$ ist eine Äquivalenzrelation, daher muss diese Eigenschaft durch Nebenbedingungen festgelegt werden.
1. Reflexivität: $x_{vv} = 1\ \forall v \in V$
2. Symmetrie: $x_{uv} = x_{vu}\ \forall u, v \in V$
3. Transitivität:
$x_{uv} = 1, x_{vw} = 1 \rightarrow x_{uw} = 1 \Leftrightarrow x_{uv} + x_{vw} \leq x_{uw} + 1\ \forall u, v, w \in V$
Weiter muss die Zusatzeinschränkung, dass maximal k Knoten in einem Cluster liegen dürfen, beachtet werden:
$\sum_{u \in V} x_{vu} \leq k\ \forall v \in V$
Die zu maximierende Zielfunktion für MaxAgree ist ω_{ma}, gegeben in der Formel (1), analog ist die zu minimierende Zielfunktion für MinDisagree ω_{md}, gegeben in Formel (2).

$$\omega_{ma}(\$) = \sum_{\{u,v\}\in E^+} x_{uv}\cdot\omega(\{u,v\}) + \sum_{\{u,v\}\in E^-}(1-x_{uv})\cdot\omega(\{u,v\}) \quad (1)$$

$$\omega_{md}(\$) = \sum_{\{u,v\}\in E^-} x_{uv}\cdot\omega(\{u,v\}) + \sum_{\{u,v\}\in E^+}(1-x_{uv})\cdot\omega(\{u,v\}) \quad (2)$$

Die Anzahl der Variablen in diesem ILP ist in $O(n^2)$, die Anzahl der Einschränkungen ist in $O(n^3)$.

3 Theoretische Untersuchung

In diesem Abschnitt wird als Erstes gezeigt, dass das untersuchte Problem NP-hart ist. Danach werden jeweils ein Algorithmus zur Approximation von MaxAgree und MinDisagree vorgestellt und Aussagen über deren Approximationsgüte gezeigt.

3.1 Clustern mit beschränkter Clustergröße ist NP-hart

Beweis durch Reduktion von „Partition into Triangles“ (siehe [4] Seiten 68,69) auf das zu „Clustern mit beschränkter Clustergröße“ gehörige Entscheidungsproblem. Damit wird gezeigt, dass das Entscheidungsproblem und damit auch „Clustern mit beschränkter Clustergröße“ NP-hart ist. Das Entscheidungsproblem hat außer allen Eingaben, die „Clustern mit beschränkter Clustergröße“ hat, noch zusätzlich eine Eingabe $\hat{\omega}_{ma} \in \mathbb{R}$ oder $\hat{\omega}_{md} \in \mathbb{R}$. Als Ausgabe wird lediglich gegeben, ob eine Clusterung $\$$ existiert mit $\omega_{ma}(\$) \geq \hat{\omega}_{ma}$ beziehungsweise mit $\omega_{md}(\$) \leq \hat{\omega}_{md}$.

Gegeben: Eine Instanz von „Partition into Triangles“, d. h. ein Graph $G = (V, E)$, für den entschieden werden soll, ob eine Partitionierung der Knotenmenge möglich ist, für die gilt: jede Gruppe der Partition enthält genau 3 Knoten u, v, w und jede mögliche Kante innerhalb einer Gruppe dieser Partition liegt in E, d. h. $\{u, v\}, \{u, w\}, \{v, w\} \in E$.

Diese Instanz P von „Partition into Triangles“ wird im Folgenden in eine Instanz C von „Clustern mit beschränkter Clustergröße“ transformiert, so dass sich aus einer optimalen Lösung von C die Lösung von P in polynomialer Zeit berechnen lässt. Damit ist gezeigt, dass, falls sich C in polynomialer Zeit lösen ließe, sich auch die Entscheidung für P in polynomialer Zeit berechnen lässt.

Fall 1: $k = 3$ Man nehme G und beschrifte alle Kanten von G mit + und gebe ihnen das Gewicht 1. Für die optimale Lösung von „Clustern mit beschränkter Clustergröße“ auf diesem Graphen gilt, dass die Anzahl von Kanten innerhalb von Clustern maximal ist. Falls auf G eine Partitionierung in Dreiecke existiert, ist diese auch eine optimale Clusterung, d. h. $MaxAgree[3](G) = n \Leftrightarrow MinDisagree[3](G) = |E| - n$. Falls auf G keine Partitionierung in Dreiecke existiert, so ist mindestens ein durch die Partition induzierter Subgraph nicht vollständig $\Rightarrow MaxAgree[3](G) < n \Leftrightarrow MinDisagree[3](G) > |E| - n$.

Fall 2: $k = 4$ Erzeuge aus dem Graphen G aus P einen Graphen G' für den C gelöst wird. Analog zum Fall $k = 3$ nimmt man G und beschriftet alle Kanten

mit + und gibt ihnen Gewicht 1. Um zu große Cluster im Originalgraph zu vermeiden führt man Zusatzknoten ZV ein, und zwar für jede mögliche Kombination von drei Knoten im Originalgraph genau einen (dies sind $\binom{|V|}{3}$ also polynomial viele). Von jedem dieser Zusatzknoten werden zu jedem Knoten der zugehörigen 3-er Menge Pluskanten mit Gewicht 4 eingefügt, und jedes Paar von Zusatzknoten wird durch eine Minuskante vom Gewicht 13 verbunden. Dadurch werden für eine optimale Clusterung zwei Dinge erreicht:
1. Es befinden sich nie zwei Zusatzknoten in einem gemeinsamen Cluster. Ein Zusatzknoten ist zu genau drei Pluskanten inzident, und damit kann ein Zusatzknoten durch maximal drei Pluskanten mit Gewicht 4 mit den restlichen Knoten des Clusters verbunden sein, und diese wiegen zusammen weniger als 13. Deshalb könnte man die Clusterung verbessern, falls man von einem Cluster mit zwei Zusatzknoten einen Zusatzknoten abtrennen würde.
2. In jedem Cluster, in dem sich ein Originalknoten v befindet, ist auch ein Zusatzknoten, der mit diesem Knoten verbunden ist. Das Gesamtgewicht der Pluskanten, die von v zu anderen Originalknoten in seinem Cluster gehen könnten, ist maximal 3 (alles Kanten im Originalgraphen, $|S| \leq 4$), aber die Verbindung zu einem Zusatzknoten hat Gewicht 4. Da es für $|V| \geq 6$ mehr Zusatzknoten mit Pluskante zu v als Originalknoten gibt, und wegen 1. kein Originalknoten mit mehr als einem Zusatzknoten in einem Cluster ist, gibt es immer einen Zusatzknoten mit Pluskante zu v der alleine in seinem Cluster ist. Falls nun v mit keinem Zusatzknoten in seinem Cluster verbunden wäre, könnte man die Clusterung verbessern, indem man v aus seinem Cluster entfernt und mit einem durch eine Pluskante verbundenen Zusatzknoten, der in seinem Cluster alleine ist, zu einem neuen Cluster verbindet.
Daher ist jeder Originalknoten in einer optimalen Clusterung mit genau einem Zusatzknoten verbunden, d. h. mit ihm in einem gemeinsamen Cluster und durch eine Pluskante verbunden. Dadurch enthält jeder Cluster maximal drei Knoten aus dem Originalgraphen (jeder Zusatzknoten ist Endknoten von genau 3 Pluskanten).
Gesamtbewertung: Betrachte das Ergebnis für $MaxAgree[4](G')$ bzw. $\overline{MinDisagree[4](G')}$. In einer Optimalclusterung gilt für jeden Knoten $v \in V$: es existiert genau ein $z \in ZV$ mit $\{v, z\} \in E(G')$ und v und z liegen im gleichen Cluster. Deshalb enthält eine Optimalclusterung immer genau $|V|$ Pluskanten mit Gewicht 4 innerhalb der Cluster. Außerdem gibt es für die Optimalclusterung keine Minuskanten innerhalb der Cluster. Daraus folgt: $MaxAgree[4](G') = |V(G)| \cdot 4 + |\{\text{richtig geclusterte Kanten in G}\}| + 13 \cdot |E^-(G')|$ und
$MinDisagree[4](G') = (3 \cdot |ZV| - |V(G)|) \cdot 4 + |\{\text{falsch geclusterte Kanten in G}\}|$
Deshalb ergibt sich aus einer optimalen Clusterung auf dem Gesamtgraph eine optimale 3-er Clusterung auf dem Originalgraph,und diese löst „Partition into Triangles“ (s. o.).

Fall 3: $k > 4$ Hier muss man nur die Methode, zu große Cluster zu verhindern, aus Fall 2 erweitern. Dazu verändert man die Kantengewichte an den Zusatzknoten. Also sei G' wie in Fall 2. Die Pluskanten von einem Zusatzknoten zu einem Knoten aus seiner 3-er Menge haben nun Gewicht k (statt 4). Die Minuskanten zwischen je zwei Zusatzknoten haben nun Gewicht $3k + 1$ (statt 13). Damit gelten wieder analog 1. und 2. aus Fall 2 und die sich dadurch ergebenden Einschränkungen der Clusterung auf G. Damit ergibt wieder eine optimale Clusterung des Gesamtgraphen eine optimale 3-er Clusterung des Originalgraphen.

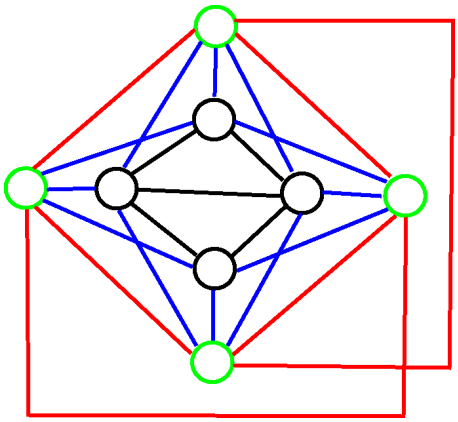

Abbildung 1: Beispiel für Fall 2/3: Der Originalgraph ist schwarz, die hinzugefügten Knoten grün, die hinzugefügten Pluskanten blau und die hinzugefügten Minuskanten rot.

Die Anzahl der hinzugefügten Objekte ist immer polynomial in den Eingangsgrößen.

Daher kann man mit einer Optimallösung von „Clustern mit beschränkter Clustergröße“ entscheiden, ob ein Graph eine Partitionierung in Dreiecke zulässt. Damit ist gezeigt, dass das Problem NP-hart ist, deshalb werden im Folgenden zwei Approximationsalgorithmen vorgestellt.

3.2 Faktor k Approximation von MaxAgree

Im folgenden wird einen Approximationsalgorithmus für MaxAgree mit Güte k vorgestellt. Die Grundidee ist ein Greedy-Algorithmus, der einen Cluster mit maximaler Bewertung sucht, diesen ausgibt, und mit dem Restgraphen fortfährt. Dabei sei die Bewertung eines Clusters das Gesamtgewicht der innerhalb des Clusters liegenden Pluskanten abzüglich des Gesamtgewichts der innerhalb des Clusters liegenden Minuskanten. Der Algorithmus erzeugt zunächst eine leere Clusterung. Danach wird ein Cluster mit maximaler Bewertung gesucht, der zwischen 2 und k Knoten enthält. Existiert kein Cluster mit einer Bewertung > 0 wird ein einzelner Knoten als Cluster verwendet. Der so gefundene Cluster wird zur Clusterung hinzugefügt und seine Knoten aus dem Graphen gelöscht. Das Suchen von maximalen Clustern wird so lange wiederholt, bis der Graph keine Knoten mehr enthält.

Der Pseudo-Code des Algorithmus ist in Algorithmus 1 gegeben; er ist polynomial bezüglich aller Größen im Graph, hat aber exponentielle Laufzeit in k. Da k hier als Konstante betrachtet wird, ist die Laufzeit somit polynomial.

Lemma 3.1 *Algorithmus 1 approximiert das Optimum von MaxAgree mit einem Approximationsfaktor von k.*

Beweis:
Sei $\$_o$ eine Clusterung die das Optimum von MaxAgree erreicht, und $\$_g$ die

Algorithm 1 Greedy-Clustering

Gegeben: G gewichteter Graph, k maximale Anzahl der Knoten in einem Cluster

```
1: $ ← ∅ {Clustering}
2: while |V| > 0 do
3:    S_b ← {v}   v ∈ V beliebig {bester Cluster}
4:    g_b ← 0 {bestes Clustergewicht}
5:    for i = 2 to min{|V|, k} do
6:       for all S ⊆ V mit |S| = i do
7:          {es gibt (|V| choose i) solcher Teilmengen}
8:          g+(S) ← Gesamtgewicht der Pluskanten innerhalb von S
9:          g−(S) ← Gesamtgewicht der Minuskanten innerhalb von S
10:         g(S) ← g+(S) − g−(S)
11:         if g(S) > g_b then
12:            S_b ← S , g_b ← g(S)
13:         end if
14:      end for
15:   end for
16:   V ← V − S_b
17:   $ ← $ ∪ {S_b}
18: end while
```

von Algorithmus 1 erzeugte. Mit $g^+(S)$, $g^-(S)$ und $g(S)$ aus dem Algorithmus gilt $\omega_{ma}(\$) = \sum_{S\in\$} g(S) + \sum_{e\in E^-} \omega(e)$. Betrachte hier zunächst $\tilde{\omega}_{ma}(\$) = \sum_{S\in\$} g(S)$. Weder in $\$_o$ noch in $\$_g$ existiert ein Cluster S mit $g(S) < 0$, da sich ein solcher Cluster immer zu kleineren Clustern mit Bewertung 0 zerlegen läßt.

Man betrachte nun den ersten von Algorithmus 1 zu $\$_g$ hinzugefügten Cluster S_1, für diesen gilt $g(S_1) \geq g(S)\ \forall S \in \$_o$, außerdem enthält S_1 maximal k Knoten. Wenn man nun alle Cluster aus $\$_o$, die mit S_1 mindestens einen Knoten gemeinsam haben, aus $G = (V, E)$ entfernt, so erhält man $G_1 = (V_1, E_1)$ mit $V_1 = V - \bigcup_{S\in\$_o,\ S\cap S_1\neq\emptyset} S$. Die von $\$_o$ auf G_1 induzierte Clusterung $\$_o(G_1)$ hat eine um maximal $k \cdot g(S_1)$ kleinere Bewertung als $\$_o$ (bezüglich $\tilde{\omega}_{ma}$).

Analog kann man für den zweiten Cluster S_2, der von Algorithmus 1 erzeugt wurde, alle Cluster aus G_1 entfernen, die mit S_2 mindestens einen Knoten gemeinsam haben. Da S_2 der Cluster mit der besten Bewertung auf $\hat{G}_1 = (\hat{V}_1, \hat{E}_1)$ mit $\hat{V}_1 = V - S_1 \supseteq V_1$ ist, ist $g(S_2) \geq g(S)\ \forall S \in \$_o(G_1)$. Wenn man nun alle Cluster aus $\$_o(G_1)$, die mit S_2 mindestens einen Knoten gemeinsam haben, aus G_1 entfernt, so erhält man $G_2 = (V_2, E_2)$ mit $V_2 = V_1 - \bigcup_{S\in\$_o(G_1),\ S\cap S_2\neq\emptyset} S$. Deshalb ist $\tilde{\omega}_{ma}(\$_o(G_2)) \geq \tilde{\omega}_{ma}(\$_o(G_1)) - k \cdot g(S_2)$.

Wenn man nun analog für alle Cluster, die von Algorithmus 1 erzeugt wurden, so fortfährt, kommt man zu einem $r \in \mathbb{N}$ für das gilt $G_r = \emptyset$ und damit auch $\tilde{\omega}_{ma}(\$_o(G_r)) = 0$. Da man bis dahin maximal $k \cdot \tilde{\omega}_{ma}(\$_g)$ von $\tilde{\omega}_{ma}(\$_o)$ abgezogen hat, gilt: $\tilde{\omega}_{ma}(\$_o) \leq k \cdot \tilde{\omega}_{ma}(\$_g)$ und damit $\omega_{ma}(\$_o) \leq k \cdot \omega_{ma}(\$_g)$

□

Die Abschätzung der Approximationsgüte ist scharf: Für jedes k und ein beliebiges $\epsilon > 0$ existiert ein Graph mit $\omega_{ma}(\$_o) = \frac{k}{1+\epsilon} \cdot \omega_{ma}(\$_g)$. Betrachte etwa

die folgende Konstruktion: Man beginnt mit einem Stern der $k-1$ Blätter hat; alle Kanten haben Gewicht $1+\epsilon$ und sind Pluskanten. Anschließend hängt man an alle Knoten (auch an das Zentrum) $k-1$ weitere Blätter, die mit Pluskanten des Gewichts 1 verbunden sind. Der Greedy-Clustering Algorithmus bildet einen Cluster aus dem initialen Stern, und packt alle anderen Knoten einzeln in ihren eigenen Cluster. Dies ergibt einen Zielfunktionswert $\omega_{ma}(\$_g) = (k-1)\cdot(1+\epsilon)$. Die optimale Clusterung wäre erreicht, wenn jeder Knoten des originalen Sterns in einem Cluster mit seinen neu angehängten Knoten liegen würde. Dies ergäbe einen Zielfunktionswert $\omega_{ma}(\$_0) = k\cdot(k-1)$. Abbildung 2 zeigt eine solche Instanz für $k=4$

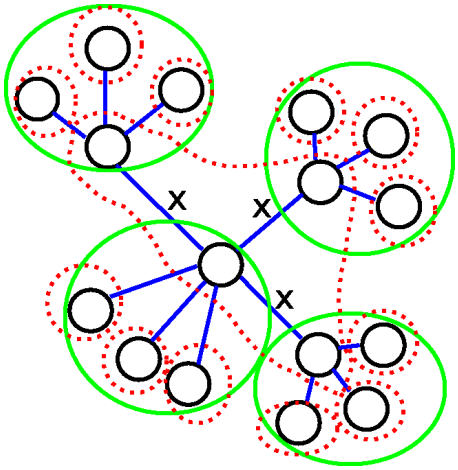

Abbildung 2: Ein Beispiel, dass die Approximationsgüte scharf abgeschätzt wurde. Die Kanten des intialen Sterns sind mit x gekennzeichnet.

Die grünen Linien deuten die optimale Clusterung an, die rot gestrichelten Linien die Clusterung, die von Algorithmus 1 berechnet wird. Analoge Beispiele lassen sich für jedes andere k konstruieren.

3.3 Faktor $k^2/2$ Approximation von MinDisagree

Hier wird das relaxierte Lineare Programm (ILP siehe oben) gelöst, und dessen Lösung problemspezifisch gerundet. Der Algorithmus 2 bildet aus der Lösungsmatrix $X = ((x_{uv}))\ u,v \in V$ eine Adjazenzmatrix, die dadurch definierten Zusammenhangskomponenten bilden die Cluster. Der Algorithmus erzeugt die Adjazenzmatrix $A = ((a_{uv}))$ aus X nach folgender Vorschrift:

$$a_{uv} = \begin{cases} 1 & \text{falls} \quad x_{uv} \geq 1 - 2/k^2 \\ 0 & \text{sonst} \end{cases}$$

Die durch A definierten Zusammenhangskomponenten bilden die Cluster; sie werden mit Hilfe des Algorithmus von Warshall ausgegeben.

Lemma 3.2 *Die durch Algorithmus 2 gebildeten Cluster enthalten maximal k Knoten.*

Algorithm 2 Simple-Clustering

Gegeben: G gewichteter Graph, k maximale Anzahl der Knoten in einem Cluster, relaxierte Lösung x_{uv} $\forall (u,v) \in V \times V$ des ILPs

{bilde die Adjazenzmatrix A}
for all $u \in V$ **do**
  for all $v \in V$ **do**
    if $x_{uv} \geq 1 - 2/k^2$ **then**
      $A[u,v] \leftarrow 1$
    else
      $A[u,v] \leftarrow 0$
    end if
  end for
end for
Bilde transitive Hülle D von A (Algorithmus von Warshall)
{bilde die Clusterung}
$\$ \leftarrow \emptyset$ {Clustering}
while $|V| > 0$ **do**
  $S \leftarrow \{v\}$ $v \in V$ beliebig {neuer Cluster in dem v liegt}
  for all $w \in V$ **do**
    if $\exists u \in S \ : \ D[u,w] = 1$ **then**
      {falls $D[u,w] = 1$ so ist auch $D[v,w] = 1 \ \forall v \in S$}
      $S \leftarrow S \cup \{w\}$
    end if
  end for
  $V \leftarrow V - S$
  $\$ \leftarrow \$ \cup \{S\}$
end while

Beweis:
Beweis durch Widerspruch: Annahme: Es existiert ein Cluster mit mindestens $k+1$ Knoten. Es wird nun gezeigt, dass dann in diesem Cluster ein Knoten v existiert, der die Bedingung $\sum_{u \in V} x_{vu} \leq k$ verletzt.

Betrachte eine Zusammenhangskomponente H bezüglich A, mit o. B. d. A. $k+1$ Knoten, die diesen Cluster induziert. Sei H' ein aufspannender Baum für H. Sei v Wurzel in H', mit der Eigenschaft, dass kein Teilbaum, der an v hängt mehr als $\lfloor \frac{k+1}{2} \rfloor$ Knoten hat (ein solcher Knoten existiert immer!). Der Abstand von u und v sei die minimale Anzahl von Kanten, die ein Weg von v nach u in H benötigt. Wenn man nun die Summe der Abstände aller Knoten in H von v betrachtet, so wird diese maximal, falls an v genau zwei Teilbäume hängen, die vollständig zu einem Pfad entartet sind, und H keine weiteren Kanten enthält. Diese Teilbäume unterscheiden sich in ihrer Größe um maximal einen Knoten.

Eine Kante $\{u, w\}$ bezüglich A existiert genau dann, wenn für x_{uw} aus der relaxierten Lösung gilt $x_{uw} \geq 1 - 2/k^2$. Deshalb gilt für einen Knoten $w \in H$ mit Abstand i von v $x_{vw} \geq 1 - i \cdot 2/k^2$ (Transitivität !).

Fall 1: k gerade, mit obigen Übelegungen gilt:

$$\begin{aligned}
\sum_{w \in H} x_{vw} &\geq 1 + 2 \cdot \sum_{i=1}^{k/2} (1 - i \cdot \frac{2}{k^2}) \\
&= 1 + k - \frac{4}{k^2} \cdot \sum_{i=1}^{k/2} i \\
&= 1 + k - \frac{4}{k^2} \cdot (\frac{k}{2} \cdot (\frac{k}{2} + 1) \cdot \frac{1}{2}) \\
&= 1 + k - \frac{1}{k} \cdot (\frac{k}{2} + 1) \\
&= 1 + k - \frac{k+2}{2k}
\end{aligned}$$

Da aber $\frac{k+2}{2k} < 1$ für $k \geq 3$, ist $\sum_{w \in H} x_{vw} > k \Rightarrow$ Widerspruch zu den Nebenbedingungen des ILPs.

Fall 2: k ungerade, mit obigen Übelegungen gilt:

$$\begin{aligned}
\sum_{w \in H} x_{vw} &\geq 1 + 2 \cdot \sum_{i=1}^{(k-1)/2} (1 - i \cdot \frac{2}{k^2}) + (1 - \frac{k+1}{2} \cdot \frac{2}{k^2}) \\
&= 1 + (k-1) - \frac{4}{k^2} \cdot \sum_{i=1}^{(k-1)/2} i + 1 - \frac{k+1}{k^2} \\
&= 1 + k - \frac{k+1}{k^2} - \frac{4}{k^2} \cdot (\frac{k-1}{2} \cdot (\frac{k-1}{2} + 1) \cdot \frac{1}{2}) \\
&= 1 + k - \frac{k+1}{k^2} - \frac{1}{k^2} \cdot (k-1) \cdot \frac{k+1}{2} \\
&= 1 + k - \frac{(k+1)^2}{2k^2}
\end{aligned}$$

Da aber $\frac{(k+1)^2}{2k^2} < 1$ für $k \geq 3$, ist $\sum_{w \in H} x_{vw} > k \Rightarrow$ Widerspruch zu den Nebenbedingungen des ILPs.

Damit können die $k+1$ Knoten nicht in einem gemeinsamen Cluster liegen, und somit kann kein Cluster mit mehr als k Knoten entstehen.

□

Lemma 3.3 *Algorithmus 2 approximiert das Optimum von MinDisagree mit einem Approximationsfaktor von $k^2/2$.*

Beweis:
Betrachte zunächst die Approximation für Plus- und Minuskanten getrennt.

Pluskanten: Die relaxierte Lösung des ILP hat eine mindestens ebensogute Bewertung wie die Optimallösung. Eine von Algorithmus 2 falsch geclusterte Pluskante war in der relaxierten Lösung zu mindestens $2/k^2$ falsch geclustert, sonst wäre sie in einen Cluster gekommen. Deshalb erzeugt Algorithmus 2 eine Clusterung, die einen um maximal Faktor $k^2/2$ größeren Fehler auf den Pluskanten hat als die optimale Clusterung.

Minuskanten: Wenn eine Minuskante in einem Cluster liegt, so haben ihre Endknoten u, v maximal Abstand $k-1$ bezüglich A. Damit gilt $x_{uv} \geq 1 - \frac{2 \cdot (k-1)}{k^2} \geq \frac{1}{2}$ für $k \geq 3$, d. h. die Kante lag auch in der relaxierten Lösung zu mindestens der Hälfte in einem Cluster. Deshalb erzeugt Algorithmus 2 eine Clusterung, die einen um maximal Faktor 2 größeren Fehler auf den Minuskanten hat als die optimale Clusterung.

Damit ist gezeigt, dass Algorithmus 2 eine Clusterung erzeugt, die einen um maximal Faktor $k^2/2$ größeren Fehler hat als die optimale Clusterung.

□

4 Statistische Untersuchung

In diesem Abschnitt soll gezeigt werden, dass die Algorithmen 1 und 2 für viele Parameterkombinationen im Schnitt wesentlich besser approximieren als dies durch die hier bewiesenen Schranken erscheint.

4.1 Beschreibung des Versuchsaufbaus

Das Versuchsprogramm ist in Java geschrieben und läuft folgendermaßen ab: Als erstes ruft das Hauptprogramm den Generator auf, der einen gewichteten Graphen erzeugt. Danach wird der erzeugte gewichtete Graph an die verschiedenen Clusterungsalgorithmen übergeben, die jeweils eine Clusterung bestimmen. Als letztes ermittelt das Hauptprogramm das Gesamtgewicht der Agreements und Disagreements für jede Clusterung und bestimmt die Approximationsverhältnisse.

Zum Generator: Die Graphgenerierung basiert auf den Parametern k, n, *pcc*, *pin*, *pout* und w_{max}, die dem Generator beim Programmstart übergeben werden. Die Graphgenerierung ist dabei ein zufälliger Vorgang, der durch die Parameter gesteuert wird. Dabei ist n die Anzahl der Knoten im Graph, und k die maximale Clustergröße. Zunächst enthalte der Graph n ungeclusterte Knoten. Basierend auf k werden Cluster (im Folgenden Originalcluster) gebildet, indem eine zufällige Anzahl von $c \in \{1 \dots k\}$ Knoten aus der Menge der ungeclusterten Knoten entnommen wird; diese bilden einen neuen Originalcluster. Dies wird so lange fortgesetzt bis alle Knoten geclustert sind. Danach werden die gelabelten und gewichteten Kanten erzeugt. Dabei erzeugt man zwischen je zwei Knoten im selben Originalcluster mit Wahrscheinlichkeit *pin* eine Kante. Falls die Knoten in verschiedenen Originalclustern liegen, ist die Wahrscheinlichkeit für eine Kante *pout*. Eine Kante innerhalb eines Clusters ist korrekt, falls sie eine Pluskante ist. Eine Kante zwischen zwei Clustern ist korrekt, falls sie eine Minuskante ist. Der Parameter *pcc* ist die Fehlerwahrscheinlichkeit des Kantenlabels, d. h. *pcc* ist die Wahrscheinlichkeit einer Kante innerhalb eines Clusters, eine Minuskante zu sein, und die Wahrscheinlichkeit einer Kante zwischen zwei Clustern, eine Pluskante zu sein. Dabei beschreibt w_{max} das Maximalgewicht einer Kante, d. h. das Gewicht wird zufällig aus $\{1 \dots w_{max}\}$ gewählt.

Zu den Clusterungsalgorithmen: Die Clusterungsalgorithmen umfassen je eine Implementierung von Algorithmus 1 und Algorithmus 2 und eine Implementierung, die die optimale Clusterung findet. Die in Algorithmus 2 benötigte Lösung des relaxierten ILP wird dabei mit ILOG CPLEX berechnet, das als Bibliothek eingebunden ist. Die optimale Lösung ergibt sich, indem man das Problem als ILP formuliert und dieses durch CPLEX lösen lässt. Da das Lösen von ILPs ein NP-hartes Problem ist, kann das Finden der optimalen Lösung unter Umständen sehr lange dauern.

Da zur Berechnung des Approximationsverhältnisses die Bewertung der optimalen Lösung bekannt sein muss und die Berechnungen nicht ewig dauern sollten, mussten die Parametersätze auf solche beschränkt werden, die eine einigermaßen schnelle Berechnung der Optimallösung zulassen.

4.2 Messungen

Für die Überblicksmessungen werden als Standardparameter gesetzt: $k = 4$, $pcc = 0.1$, $n = 50$, $pin = 0.7$, $pout = 0.2$ und $w_{max} = 5$, wobei immer nur der gerade betrachtete Parameter von diesen Werten abweicht. Für jeden einzelnen Boxplot wurden die Ergebnisse von 30 Durchläufen mit zufällig generierten Graphen verwendet.

Die Haupteinschränkung beim Testen war der hohe Zeitbedarf zum Berechnen der optimalen Lösung. Beispielsweise benötigte der Durchlauf von 30 Tests mit der Standardbelegung insgesamt 6 Minuten, zum Vergleich benötigten lediglich 5 Durchläufe für $n = 72$ schon 18 Minuten, 5 Durchläufe für $n = 76$ sogar 44 Minuten. Da für größere n im Vergleich zur Rechenzeit keine größeren Erkenntnisse mehr zu erwarten waren, wurde nur bis $n = 70$ mit 30 Durchläufen getestet. Die Standardparameter sind absichtlich relativ klein (bzgl. des untersuchten Bereiches) gewählt, um die Gesamtrechenzeit nicht unnötig zu verlängern. Das maximale Kantengewicht wurde absichtlich niedrig angesetzt um „Ausreißer“ durch einzelne sehr schwere Kanten zu verhindern. Außerdem sind bei den be-

schriebenen Anwendungen sowieso nur wenige diskrete Werte für die Kanten sinnvoll, so dass sich auch hier kein Problem ergibt.

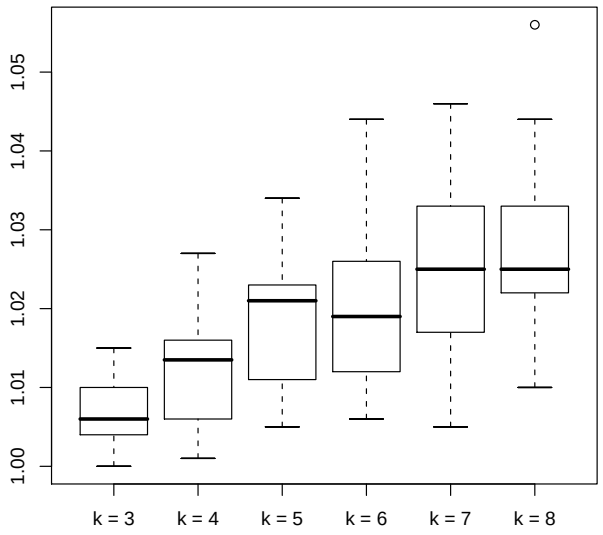

Abbildung 3: Approximationsverhältnis für MaxAgree von Algorithmus 1 für verschiedene k

Da k im Vergleich zu n „klein“ sein sollte, waren auch nur Tests für kleinere k sinnvoll.

Die benötigte Zeit für die Tests mit verschiedenen Werten von *pcc* war sehr stark vom Wert von *pcc* abhängig. Die Zeit für 30 Durchläufe wuchs von wenigen Minuten für die kleinen Werte bis auf 14 Stunden für $pcc = 0.35$ und 35 Stunden für $pcc = 0.4$. Damit war ein Testen für noch größere Werte von *pcc* praktisch ausgeschlossen.

Auch bei *pout* war es aufgrund stark wachsender Laufzeiten nicht möglich, den gesamten natürlichen Parameterbereich zu testen. Die Laufzeit für 30 Durchläufe wuchs von wenigen Minuten für kleine Werte von *pout* bis auf über 3 Stunden für $pout = 0.6$ und für $pout = 0.7$ wurden für die ersten 15 Durchläufe schon über 4 Stunden benötigt.

Außer den stark schwankenden Laufzeiten ergaben sich bei den Tests keine besonderen Auffälligkeiten.

Ergebnisse: Für Algorithmus 1 ergaben sich bei allen Tests Mediane für das Approximationsverhältnis von MaxAgree von ≤ 1.03, was wesentlich besser ist als die in Abschnitt 3 gezeigte theoretische Schranke von k (hier meist $k = 4$). Da hier für kleinere n relativ große Bereiche aller Eingabeparameter getestet wurden, kann man sagen, dass Algorithmus 1 für die meisten Eingaben das Optimum von MaxAgree wesentlich besser als mit Faktor k approximiert, wenn

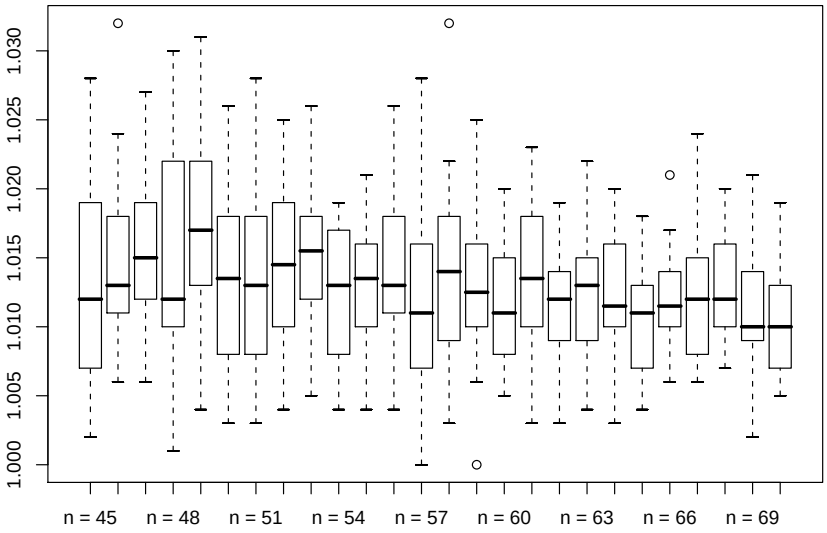

Abbildung 4: Approximationsverhältnis für MaxAgree von Algorithmus 1 für verschiedene n

die Graphstruktur eine signifikante Clusterung aufweist. Abbildung 4 zeigt, dass sich die mittlere Approximationsgüte bei wachsendem n kaum verändert, daher ist die Approximation auch für größere n wahrscheinlich wesentlich besser als k.

Für Algorithmus 2 ergaben sich bei allen Tests Mediane für das Approximationsverhältnis von MinDisagree von ≤ 1.8, was wesentlich besser ist als die in Abschnitt 3 gezeigte theoretische Schranke von $\frac{k^2}{2}$ (hier meist $k = 4$). Da hier für kleinere n relativ große Bereiche aller Eingabeparameter getestet wurden, kann man sagen, dass Algorithmus 2 für die meisten Eingaben das Optimum von MinDisagree wesentlich besser als mit Faktor $\frac{k^2}{2}$ approximiert, wenn die Graphstruktur eine signifikante Clusterung aufweist. Abbildung 9 zeigt, dass sich die mittlere Approximationsgüte bei wachsendem n nur leicht verschlechtert, daher ist die Approximation auch für größere n wahrscheinlich wesentlich besser als $\frac{k^2}{2}$.

5 Zusammenfassung und Ausblick

In dieser Arbeit wurde das Problem des „Clusterns mit beschränkter Clustergröße“ vorgestellt und gezeigt, dass das Finden der optimalen Lösung NP-hart ist. Weiter wurden zwei Algorithmen angegeben und gezeigt, dass diese die beiden Ziele des „Clusterns mit beschränkter Clustergröße“ mit Faktor k bzw. $\frac{k^2}{2}$ approximieren. Außerdem wurde empirisch belegt, dass die beiden angegebenen Algorithmen auf den meisten Probleminstanzen ein wesentlich besseres Appro-

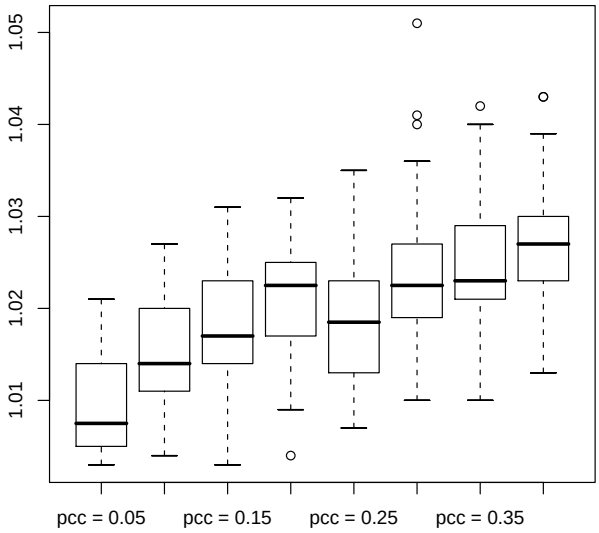

Abbildung 5: Approximationsverhältnis für MaxAgree von Algorithmus 1 für verschiedene *pcc*

ximationsverhältnis erzielen.

Aus diesen Ergebnissen ergibt sich direkt die Frage, ob dieses Problem APX-hart ist oder ob ein PTAS für eines der beiden Ziele existiert.

Eine weitere interessante Frage ergibt sich, wenn man beide Ziele gleichzeitig approximieren will, da es für beide vorgestellten Algorithmen Probleminstanzen gibt, für die sie das jeweils andere Ziel beliebig schlecht approximieren. Vielleicht kann eine Kombination aus beiden Algorithmen beide Ziele einigermaßen vernünftig approximieren. Andererseits gibt es für beide Approximationsziele möglicherweise Algorithmen, die diese Ziele wesentlich besser approximieren als die hier vorgestellten Algorithmen.

Ein weiteres offenes Problem ist die Darstellung einer berechneten Clusterung. Die Herausforderung besteht zum einen darin, die Clusterung so zu zeichnen, dass ein Betrachter die Cluster schnell erkennen kann, aber auch die Struktur des geclusterten Graphen noch erkennbar ist. Zum anderen ist es wahrscheinlich noch schwerer, einen Graphen mit einer gegebenen Zeichnung und einer Clusterung so zu zeichnen, dass die Clusterung gut zu erkennen ist, aber auch die Entsprechungen von Knoten in der Clusterung und Knoten in der ursprünglichen Zeichnung noch zu erkennen sind.

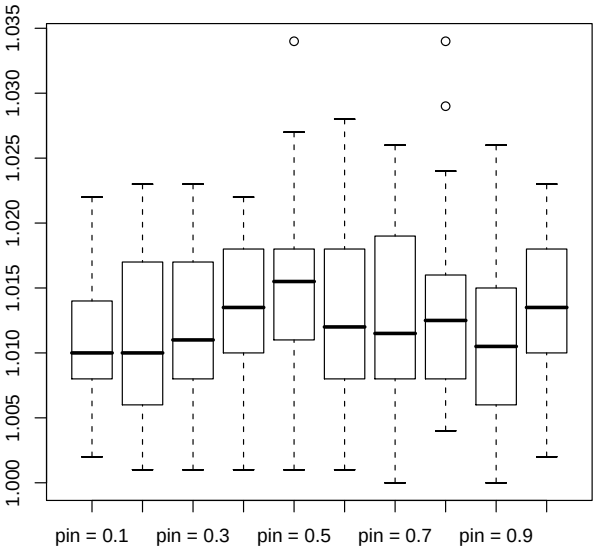

Abbildung 6: Approximationsverhältnis für MaxAgree von Algorithmus 1 für verschiedene *pin*

Literatur

[1] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. 2004.

[2] Moses Charikar, Venkatsean Guruswami, and Anthony Wirth. Clustering with qualitive information. 2003.

[3] Erik D. Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. 2005.

[4] Michael R. GareyDavid S. Johnson. *Computers and Intractability. A Guide to the Theory of $\mathcal{NP}$-Completeness*. W. H. Freeman and Company, 1979.

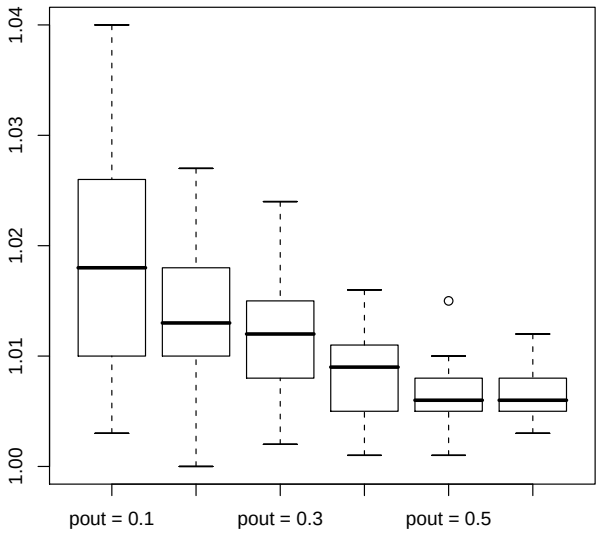

Abbildung 7: Approximationsverhältnis für MaxAgree von Algorithmus 1 für verschiedene *pout*

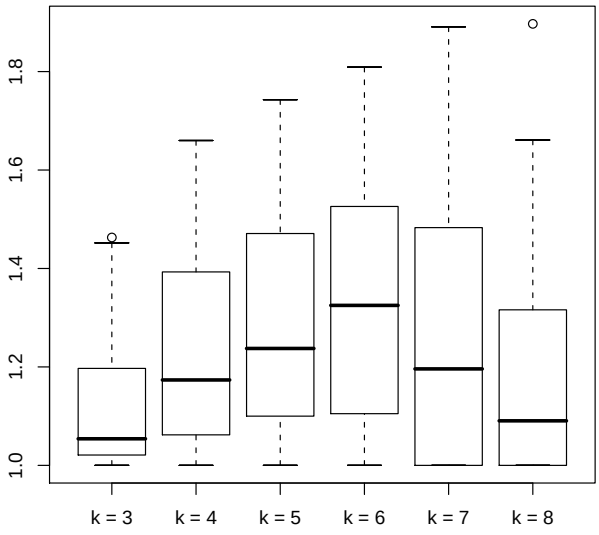

Abbildung 8: Approximationsverhältnis für MinDisagree von Algorithmus 2 für verschiedene k

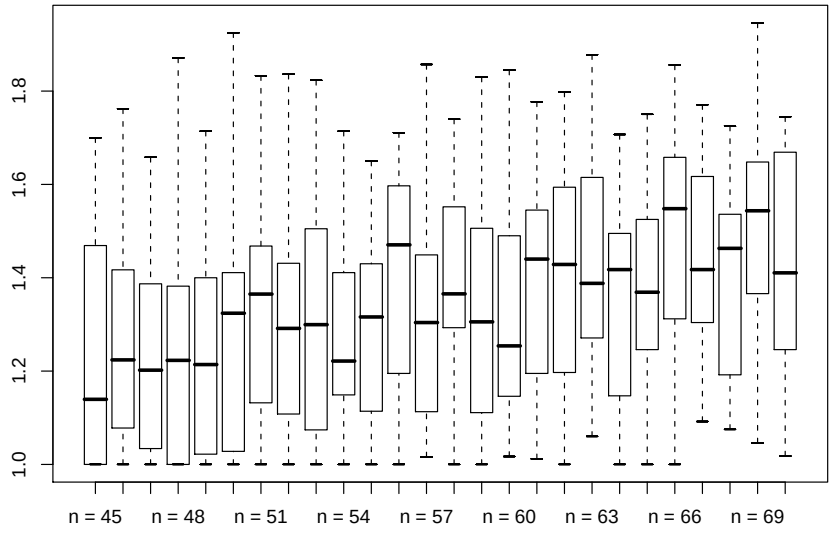

Abbildung 9: Approximationsverhältnis für MinDisagree von Algorithmus 2 für verschiedene n

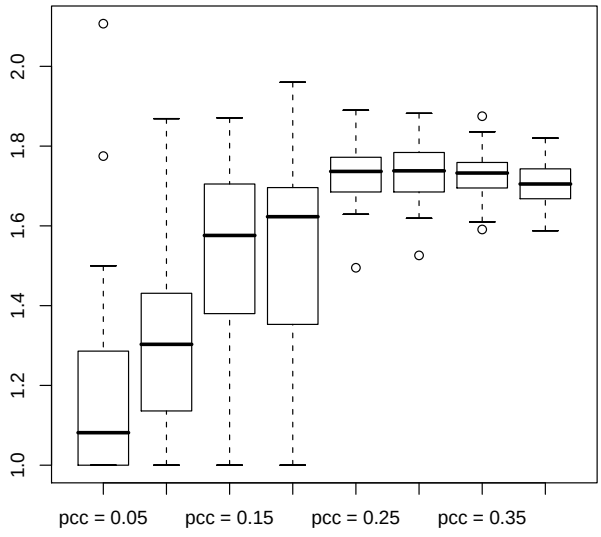

Abbildung 10: Approximationsverhältnis für MinDisagree von Algorithmus 2 für verschiedene *pcc*

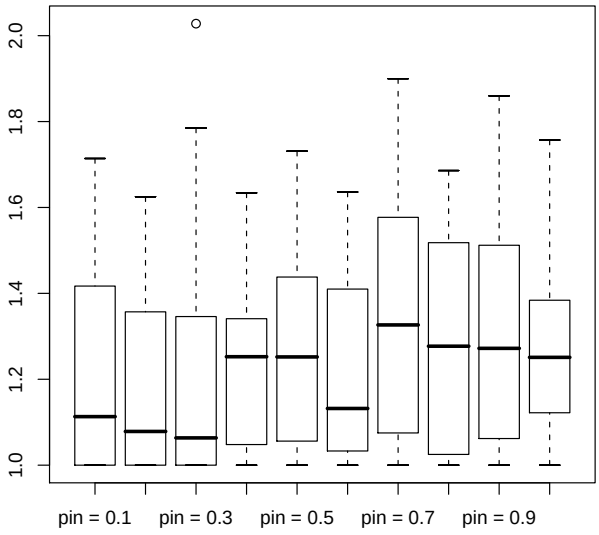

Abbildung 11: Approximationsverhältnis für MinDisagree von Algorithmus 2 für verschiedene *pin*

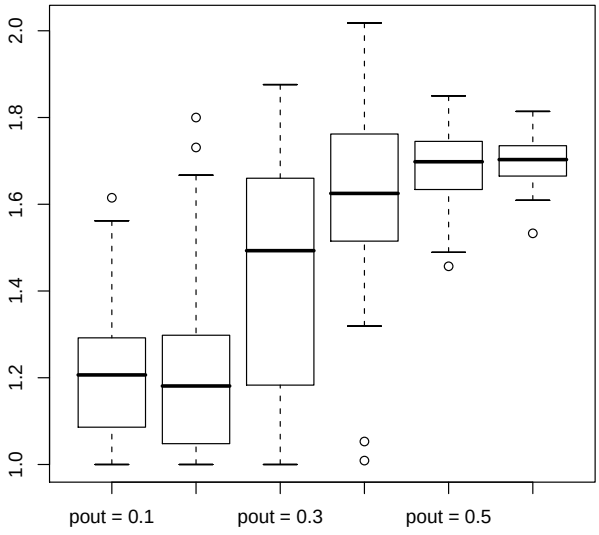

Abbildung 12: Approximationsverhältnis für MinDisagree von Algorithmus 2 für verschiedene *pout*