LP-basierte Heuristiken für Graphenclusterung mit Modularity als Qualitätsmaß

Studienarbeit

Manuel Krings
1. Februar 2008

Betreuer: Dr. Marco Gaertler, Prof. Dr. Dorothea Wagner
Institut für Theoretische Informatik

Zusammenfassung

Die Suche einer optimalen Graphenclusterung bezüglich des Modularity-Index' ist $\mathcal{NP}$-vollständig. Um auch größere Graphen effizient clustern zu können sind gute Heuristiken zur Lösung dieses Problems erforderlich. Das Problem lässt sich als ganzzahliges lineares Programm (ILP) beschreiben, was die Verwendung der relaxierten Lösung nahe legt. Diese Arbeit beschäftigt sich im Wesentlichen mit solchen Heuristiken, die auf einer Relaxierung des ILPs basieren.

Ich versichere hiermit wahrheitsgemäß, die Arbeit bis auf die dem Aufgabensteller bereits bekannte Hilfe selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderung entnommen wurde.

Unterschrift: . . . . . .

Inhaltsverzeichnis

1 Einleitung

Diese Studienarbeit beschäftigt sich mit der Clusterung von Graphen, also dem Aufteilen von Graphen durch die Zuordnung seiner Knoten zu kleineren Subgraphen. Graphen können unter anderem verschiedenste Gegebenheiten aus der realen Welt repräsentieren, wie zum Beispiel (soziale) Netzwerke. Hier interessiert man sich häufig für Gruppen innerhalb dieser Netzwerke, die besonders stark miteinander verbunden sind, also viel miteinander „zu tun haben“. Solche Gruppen werden durch die Clusterung von Graphen gefunden.

Als populäres Maß für die Güte einer Clusterung wurde der Modularity-Index eingeführt und in dieser Studienarbeit als Grundlage zur Beurteilung einer Clusterung verwendet. Da das zugehörige Optimierungsproblem, also das Berechnen einer Clusterung mit maximalem Mondularity-Index, $\mathcal{NP}$-schwer ist, sind gute Heuristiken zur Optimierung erforderlich, um auch größere Instanzen von Graphen clustern zu können. Ein Ansatzpunkt zur Approximation einer optimalen Lösung ist die Formulierung des Clusterungs-Problems als ganzzahliges lineares Programm (ILP) und dessen Relaxierung. Mit solchen Heuristiken beschäftigt sich auch diese Studienarbeit.

Zunächst werden dazu im zweiten Kapitel die Grundlagen vorgestellt. Heuristiken zur Berechnung einer möglichst guten Clusterung werden in Kapitel drei betrachtet. Anschließend befasst sich Kapitel vier mit den für die Experimente verwendeten Testgraphen. Der Aufbau und die Ergebnisse der Testläufe werden mit ihrer Auswertung im fünften Kapitel präsentiert. Schließlich werden im sechsten Kapitel die Ergebnisse zusammengefasst.

2 Grundlagen

In diesem Kapitel wird zunächst der *Modularity-Index*, das in dieser Arbeit verwendete Maß zur Bewertung der Güte einer Graphen-Clusterung, vorgestellt und die Notwendigkeit von Heuristiken erklärt, um eine Clusterung bezüglich dieses Maßes zu optimieren. Anschließend wird die Formulierung des Problems als ganzzahliges lineares Programm beschrieben, sowie ein Greedy-Algorithmus vorgestellt, der den Modularity-Index als Bewertungsfunktion nutzt.

2.1 Modularity-Index

Der *Modularity-Index* ist ein von Newman und Girvin eingeführtes Maß für die Güte einer Clusterung [1]. Der Modularity-Index basiert auf der Differenz der Anzahl der Kanten innerhalb eines jeden Clusters und der Anzahl der erwarteten Kanten in diesen Clustern. Er wird für eine gegebene Clusterung $\mathcal{C}$ zu einem Graphen $G = (V, E)$ berechnet durch die Formel

$$q(\mathcal{C}) = \frac{1}{2m} \sum_{(u,v) \in V^2} \left(E_{uv} - \frac{\deg(u) \cdot \deg(v)}{2m} \right) X_{uv}$$

$$\text{mit } E_{uv} = \begin{cases} 0 & (u,v) \notin E \\ 1 & (u,v) \in E \end{cases}$$

$$\text{und } X_{uv} = \begin{cases} 0 & \text{u und v in unterschiedlichen Clustern} \\ 1 & \text{u und v im gleichen Cluster} \end{cases}$$

Der Modularity-Index liegt immer zwischen $-1/2$ und 1. Ein Wert von 0 kann allerdings immer durch eine triviale Clusterung (nur Singletons oder alle Knoten in einem Cluster) erreicht werden.

Das Problem, $q(\mathcal{C})$ zu maximieren, ist $\mathcal{NP}$-Vollständig [2]. Daher ist es sinnvoll möglichst gute Approximationsalgorithmen zu finden.

2.2 Algorithmen

Im Folgenden werden die bereits bekannten Algorithmen vorgestellt. Diese basieren zum einen auf linearer Programmierung und zum anderen auf einem iterativen Greedy-Verfahren.

2.2.1 Greedy-Algorithmus

Der erste Algorithmus zur Optimierung des Modularity-Index' wurde von Clauset, Newman und Moore vorgestellt [3]. Dabei handelt es sich um ein Greedy-Verfahren, das iterativ Cluster verschmilzt.

Genauer gesagt startet das Verfahren mit der initialen Clusterung bei der alle Knoten als Singletons vorliegen, das heißt jeder Knoten liegt in seinem eigenen Cluster. In jedem Schritt werden dann zwei Cluster miteinander verschmolzen. Dabei wird immer das Cluster-Paar vereinigt, dessen Verschmelzung den Modularity-Index am stärksten erhöht.

Die effiziente Umsetzung dieses Verfahrens (Algorithmus 1) arbeitet mit einer Matrix, in der für jedes Paar von Clustern steht, um welchen Wert sich der Modularity-Index ändert, wenn man diese beiden Cluster miteinander verschmelzen würde. Das Verschmelzen von Clustern erfordert nur eine Addition zweier Spalten bzw. Zeilen in der Matrix, um sie zu aktualisieren. Wenn die Matrix keine positiven Werte mehr enthält, bricht der Algorithmus ab, da der Modularity-Index dann nicht weiter verbessert werden kann.

Algorithmus 1 : Greedy-Algorithmus

Input : Graph G
Clustering $C \leftarrow$ Singletons(G)
$M \leftarrow$ baueMatrix(C)
while $max(M) > 0$ **do**
  $(i, j) \leftarrow (i, j) : M(i, j) = max(M)$
  C.verbinde(i,j)
Output : C

2.2.2 Formulierung als ILP

Das Optimierungsproblem lässt sich als ganzzahliges lineares Programm (ILP) ([2]) formulieren und so mit Standard-Programmen lösen. Hierdurch erhält man eine optimale Lösung des Problems. Die Formulierung als ILP basiert auf der Idee, Variablen für alle möglichen Knotenpaare einzuführen, die eine Unterteilung des Graphen in Cluster beschreiben. Dabei werden quadratisch viele Variablen benötigt und kubisch viele Bedingungen formuliert.

Für jedes mögliche Knotenpaar gibt es eine Variable, die 0 ist, wenn beide Knoten im selben Cluster liegen oder 1, wenn sie in verschiedenen Clustern sind; man kann sie also als Distanz auffassen.

Die Bedingungen sorgen dafür, dass zum einen die Distanzen zwischen 0 und 1 liegen und man zum anderen eine konsistente Lösung erhält, die eine gültige Clusterung repräsentiert. Im letzteren Fall muss also eine Art Transitivitätsbedingung gelten, das

heißt, für je 3 verschiedene Knoten muss für die 3 Distanzen untereinander gelten: Sind zwei Distanzen gleich 1 (bzw. 0), so muss auch die dritte Distanz 1 (bzw. 0) sein.

3 Heuristiken

In diesem Kapitel werden verschiedene erarbeitete Heuristiken zur Optimierung des Modularity-Index' vorgestellt. Sie basieren alle auf der Relaxierung der im letzten Kapitel beschriebenen ILP-Formulierung.

Lässt man bei der Lösung des als ILP formulierten Problems die Ganzzahligkeitsbedingung weg, so erhält man die relaxierte Lösung. Diese relaxierte Lösung stellt eine obere Schranke für eine optimale Lösung des Problems dar. Das Lösen dieses linearen Programms (LP) hat dann nur noch polynomiellen Aufwand, dafür können nun beliebige rationale Werte zwischen 0 und 1 auftreten.

Die Lösung dieses LPs kann verwendet werden, um einen Graphen in Cluster aufzuteilen. Dazu muss man Rundungs-Algorithmen entwickeln um die Lösung ganzzahlig zu machen. Im Folgenden präsentieren wir kurz einige Beobachtungen, die als Basis für unsere Rundungsstrategien gedient haben.

Die Struktur der relaxierten Lösung

Bei der Untersuchung verschiedener Graphen stellten wir fest, dass die relaxierte Lösung oft nur die Werte 0, 1/2 und 1 enthielt. In zwei Fällen, nämlich bei den beiden Zufallsgraphen, kommen auch andere Brüche, wie zum Beispiel 1/3 vor. Dabei ist zu beachten, dass das Ergebnis keine exakten Brüche liefert, sondern, durch die ungenaue Rechenarithmetik, meist Näherungswerte (wie zum Beispiel 0,49999999999 statt 0,5) auftreten. Man konnte beobachten, dass die Laufzeit zur Berechnung des ILPs zunahm, je größer der Anteil von Paaren war, die ungleich 0 oder 1 waren.

Vergleich von LP- und ILP-Lösung

Bei einigen Graphen, wie zum Beispiel den *Cliquen mit Antennen* (siehe Abbildung 4.2(d)), stimmen LP- und ILP-Lösung überein. Am Beispiel des Graphen *Baum* sind in Tabelle 3.1 die Unterschiede zwischen der ILP- und der LP-Lösung aufgeführt. Die Anzahl der Fälle bei denen die Distanz der LP-Lösung sich um 1 von der ILP-Lösung unterscheidet liegt hier bei nur 2%. Die meisten Distanzen (89 %) wurden korrekt zugeordnet. Die übrigen Distanzen sind rationale Zahlen zwischen 0 und 1.

ILP	LP	Anzahl	Anteil
0	0	64	81 %
1	1	631	8 %
0	0,5	39	5 %
0	1	16	2 %
1	0,5	29	4 %
1	0	1	0 %

Tabelle 3.1: Vergleich von LP- und ILP-Lösung am Beispiel des Graphen *Baum*.

3.1 Schwellwert-Runden

Eine naheliegende Idee ist das Ergebnis des relaxierten LPs zu runden, um eine ganzzahlige Lösung zu erhalten. Der Algorithmus „Schwellwert-Runden“ benutzt dazu einen vorgegebenen Schwellwert S und verbindet ausgehend von Singletons alle Cluster mit Knoten, deren Distanzen diesen Schwellwert nicht übersteigen, jeweils miteinander (siehe Algorithmus 2).

Algorithmus 2 : Schwellwert-Runden

Input : Graph $G = (V, E)$, LP-Lösung(Distanzen) D, Schwellwert S
Clusterung C $\leftarrow$ Singletons(G)
forall $(i, j) \in V \times V$ **do**
  if $D(i, j) < S$ **then**
    C.verbinde(i,j)
Output : C

Dieser einfache Algorithmus hat den Nachteil, dass die Güte der Lösung vom vorgegebenen Schwellwert abhängig ist. Da man für einen Graphen im Allgemeinen nicht weiß, welcher Schwellwert das beste Ergebnis liefert, ist es hier erforderlich den jeweils besten Schwellwert durch probieren zu ermitteln.

3.2 Runden mit Sortieren

Der Algorithmus „Schwellwert-Runden“ aus dem letzten Abschnitt lässt sich verbessern, indem er den optimalen Schwellwert selbst findet.

Dieser Algorithmus (siehe Algorithmus 3) wurde implementiert, indem die Distanzen aus der Lösung des LPs zunächst sortiert werden und anschließend schrittweise in aufsteigender Folge die dazugehörigen Knoten miteinander verschmolzen werden bis der Graph nur noch aus einem einzigen großen Cluster besteht. Dabei merkt sich der Algorithmus die Clusterung mit dem besten Modularity-Index und gibt diesen am Ende aus.

Algorithmus 3 : Verbesserter Algorithmus LP-Runden

Input : Graph $G = (V, E)$, LP-Lösung(Distanzen), D=(Knoten i, Knoten j, Distanz d)
Clusterung $C \leftarrow$ Singletons(G)
Clustering bestC $\leftarrow C$
sortiere(D)
forall *int i=0; i < D.length; i++* **do**
  C.verbinde(D[i])
  if $modularity(C) > modularity(bestC)$ **then**
    $bestC \leftarrow C$
Output : bestC

Das Ergebnis muss aber nicht besser sein, als das des einfachen Schwellwert-Rundens, da alle Distanzen mit dem gleichen Abstand in einer zufälligen Reihenfolge abgearbeitet werden. So kann es passieren, dass beispielsweise für den Schwellwert $0,5$ - er umfasst dann Distanzen wie $0,499999$, nicht aber $0,5000000001$ - eine bessere Lösung gefunden wird.

Es ist also sinnvoll innerhalb einer Klasse von Distanzen eine geeignete Reihenfolge zu finden, in der man die Cluster miteinander verschmilzt.

3.3 LP-Runden mit Greedy

Es hat sich gezeigt, dass die Güte der Clusterung hauptsächlich von der geschickten Rundung der 0,5er-Distanzen abhängt. Daher bietet sich ein Algorithmus an, der das Ergebnis der relaxierten LP-Lösung mit einer anderen Optimierungsstrategie, wie zum Beispiel einem Greedy-Verfahren, kombiniert.

Ausgehend von der relaxierten Lösung werden zunächst die Knoten mit „sicheren“ Distanzen, in der Regel also die Knoten mit einer Distanz von 0, zu einem Cluster verschmolzen. Hierbei wird ein Schwellwert benötigt, bis zu dem diese Verschmelzung vorgenommen wird. Hier bietet sich für die meisten Testgraphen ein beliebiger Wert, der größer 0 und kleiner 0,5 ist, an. Für die Graphen, deren LP-Lösung auch Distanzen wie etwa $1/3$ einschließt, ist es auch entscheidend, ob der Schwellwert größer oder kleiner als diese Distanz ist. Auf die resultierende Clusterung wird der Greedy-Algorithmus angewendet.

Der Algorithmus (siehe Algorithmus 4) besteht also aus der Hintereinanderausführung von Schwellwertrunden und Greedy, wobei auf die Initialisierung mit Singletons zu Beginn des Greedy-Algorithmus verzichtet wird.

Bei den hier verwendeten Testgraphen war es nur im Fall des kleinen Zufallsgraphen von Vorteil einen höheren Schwellwert als 0 anzugeben.

Algorithmus 4 : Kombination von LP-Runden und Greedy

Input : Graph $G = (V, E)$, LP-Lösung(Distanzen) D, Schwellwert S
Clusterung C $\leftarrow$ Singletons(G)
forall $(i, j) \in V \times V$ **do**
  if $D(i, j) < S$ **then**
    C.verbinde(i,j)
$M \leftarrow$ baueMatrix(C)
while $max(M) > 0$ **do**
  $(i, j) \leftarrow (i, j) : M(i, j) = max(M)$
  C.verbinde(i,j)
Output : C

3.4 Weitere Rundungsalgorithmen

Verschiedene weitere Rundungsstrategien wurden entwickelt. Da die einzelnen Cluster der optimalem Lösung selten kleinste Cluster enthielten, gehen die im Folgenden vorgestellten Algorithmen davon aus, dass jeweils der kleinste Cluster oder einer der kleinsten Cluster mit einem anderen verschmolzen wird.

3.4.1 Algorithmen basierend auf dem Erwartungswert der Interclusterkanten

Da der Modularity-Index auf der Differenz aus tatsächlichen Kanten und dem Erwartungswert der angenommen Kanten innerhalb eines Clusters basiert, ist die Idee entstanden einen Algorithmus basierend auf dieser Differenz zu entwickeln.

In diesem Fall werden in jedem Schritt zunächst die zehn kleinsten Cluster ermittelt. Dann wird für jeden dieser Cluster unter allen anderen Clustern derjenige gesucht, der die größte Differenz zwischen der tatsächlichen Anzahl der Kanten und dem Erwartungswert hat. Schließlich werden die beiden Cluster mit der höchsten Differenz miteinander verbunden. Als Abbruchbedingung wurde anfangs angenommen, dass eine optimale Clusterung erreicht ist, wenn der Graph aus $\sqrt{n}$ Clustern besteht, wobei n für die Anzahl der Knoten steht.

Da das Ergebnis der Implementierung dieses Algorithmus nicht zufriedenstellend war, wurden weitere Varianten entwickelt:

Variation 1: Um auch beim Runden von der Lösung des relaxierten Problems zu profitieren, wird die Differenz durch das jeweilige Kantengewicht der LP-Lösung gewichtet.

Variation 2: Als Abbruchbedingung wurde nun die Unterschreitung der Differenz unter eines bestimmten Schwellwerts angenommen.

Variation 3: Anstelle der zehn kleinsten Cluster wurden die zehn Cluster mit den wenigsten Kanten ausgewählt.

Der Algorithmus mit allen seinen Variationen erwies sich allerdings als nicht Konkurrenzfähig und zeigte keine Verbesserung gegenüber dem einfachen Schwellwertrunden und wurde nicht weiter verfolgt.

3.4.2 Algorithmen basierend auf dem Kantengewicht

Der folgende Algorithmus geht immer davon aus, dass der jeweils kleinste Cluster mit einem anderen verbunden werden muss.

In jedem Schritt wird hier der jeweils kleinste Cluster mit dem Cluster verbunden, zu dem eine Kante mit der niedrigsten Distanz aus der LP-Lösung führt.

Dieser Algorithmus erwies sich als keine gute Idee, da die LP-Lösung oft nur 3 Werte annimmt und zu viele Möglichkeiten lässt den Cluster zu verbinden; die Auswahl des Clusters geschah hier zufällig unter den Kandidaten mit niedrigster Distanz. Deshalb wurde er verbessert, in dem nicht mehr nur eine Kante betrachtet wurde, sondern die durchschnittliche Distanz von allen Kanten zwischen zwei Clustern.

Auch dieser Lösungansatz zeigte keine Verbesserung gegenüber dem einfachen Schwellwertrunden.

4 Verwendete Testgraphen

In diesem Kapitel werden die verwendeteten Testgraphen beschrieben. Die Kanten- und Knotenzahl, sowie den optimalen Modularity-Index kann man der Tabelle 4.1 entnehmen. In den Abbildungen 4.1 und 4.2 sind einige dieser Graphen zu sehen, dabei wird die optimale Clusterung, sofern sie bekannt ist, durch Einfärbung der Knoten dargestellt.

4.1 Reale Graphen

Als reale Testgraphen wurden die Graphen *Dolphins*, *Football*, *Karate* und *Polbooks* aus früheren Arbeiten [4, 5, 6] zum Modularity-Index gewählt. Die Gestalt der Graphen ist in Abbildung 4.1 zu sehen. Der Graph „Dolphins“ bildet das Netzwerk sozialer Kontakte zwischen Delfinen ab, wie es auch schon in [4] benutzt wurde. Im Graph „Polbooks“ repräsentiert jeder Knoten ein politisches Buch, eine Kante verbindet zwei Bücher, die oft vom selben Leser gelesen wurden.

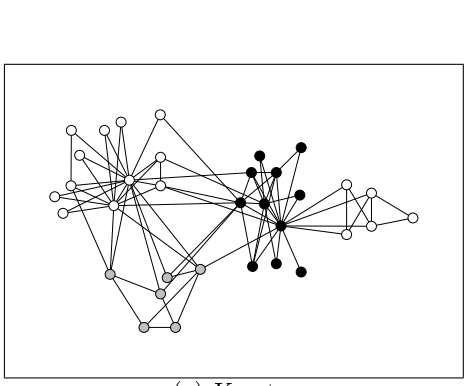
4.2 Künstliche Graphen

Die realen Testgraphen wurden durch eine Reihe künstlicher Graphen verschiedener Art ergänzt. Dabei handelt es sich zum einen um Graphen mit einer Struktur, die intuitiv „einfach“ zu clustern sind (Bäume, auf Cliquen basierende Graphen, zufällige Clustergraphen). Zum anderen sind es Graphen die entweder ganz zufällig erzeugt wurden oder eine Struktur haben, die keine Clusterung nahelegt (zum Beispiel der Graph *Gitter*). Die künstlichen Graphen werden im Folgenden beschrieben:

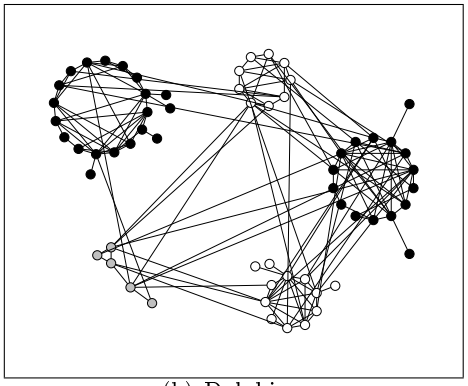
Der Graph *Baum* ist ein zufälliger Baum mit 40 Knoten, 39 Kanten und bis zu drei Kindknoten. Er ist in Abbildung 4.2(a) dargestellt.

Die beiden Graphen *Cliquen-Baum-Klein* und *Cliquen-Baum-Groß* bestehen jeweils aus einigen Cliquen, die durch eine minimale Anzahl von Kanten miteinander verbunden sind, so dass ein zusammenhängender Graph entsteht. Durch ihre klare Struktur sind sie sehr einfach zu clustern und haben einen hohen Modularity-Index. Der Graph *Cliquen-Baum-Klein* besteht aus 37 Knoten, *Cliquen-Baum-Groß* aus 77 Knoten. Die Struktur dieser Graphen ist in Abbildung 4.2(b) zu sehen.

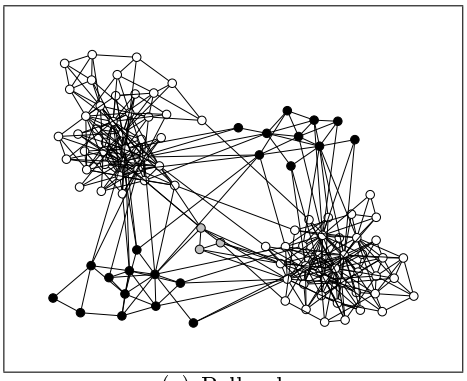
Der Graph *Gitter* 5×7 (Abbildung 4.2(c)) ist ein zweidimensionales Gitter aus 5×7 Knoten. Jeder Knoten in diesem Gitter ist mit seinem linken, rechten, oberen und



(a) Karate.



(b) Dolphins.



(c) Polbooks.

Abbildung 4.1: Reale Testgraphen. Die optimale Clusterung wird durch die Knotenfärbung dargestellt.

Tabelle 4.1: Übersicht der verwendeten Graphen

Graph	Knoten	Kanten	Modularity
Dolphin	64	159	0,53
Football	128	723	0,51
Karate	38	77	0,43
Polbooks	105	441	0,53
Baum	40	39	0,70
Cliquen-Baum-Klein	37	90	0,76
Cliquen-Baum-Groß	77	208	0,86
Gitter-5×7	35	58	0,53
Kreis-30	30	30	0,63
Clique-Antenne-15-2	45	135	0,19
Clique-Antenne-25-2	75	350	0,13
Clique-Antenne-25-3	125	375	0,24
Zufall-Groß	70	300	$< 0{,}41$
Zufall-Klein	28	40	0,49
Zufalls-Cluster-Groß	208	468	0,83
Zufalls-Cluster-Klein	112	249	0,81

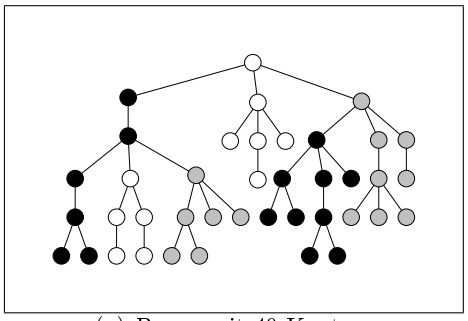
unteren Nachbarn verbunden, alle „inneren“ Knoten haben also Grad 4.

Der Graph *Kreis-30* ist ein Kreis mit 30 Knoten. Alle Knoten haben hier den Grad 2.

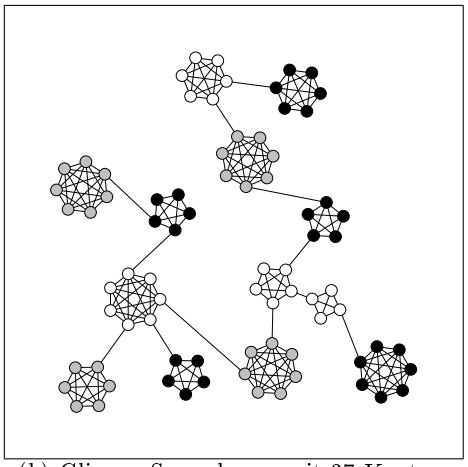
Die Graphen *Clique-Antenne-15-2*, *Clique-Antenne-25-2* und *Clique-Antenne-25-3* bestehen jeweils aus einer Clique mit 15 bzw. 25 Knoten, wobei an jedem Cliquenknoten eine „Antenne“ bestehend aus zwei oder drei weiteren Knoten angebracht ist. Die Struktur dieser Graphen ist in Abbildung 4.2(d) zu sehen.

Die Graphen *Zufalls-Cluster-Groß* und *Zufalls-Cluster-Klein* sind zufällige Clustergraphen, die mit dem „Significant Gaussian Random Generator“ erstellt wurden. Hier kann der Anteil von Inter- und Intraclusterkanten vorgegeben und damit Clustergraphen erstellt werden.

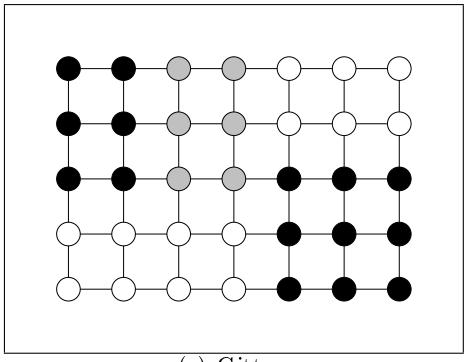
Außerdem wurden zwei zufällig generierte Zufallsgraphen *Zufall-Groß* und *Zufall-Klein* erstellt, die 28 bzw. 70 Knoten haben, bei denen keine Clusterung vorgegeben wurde.



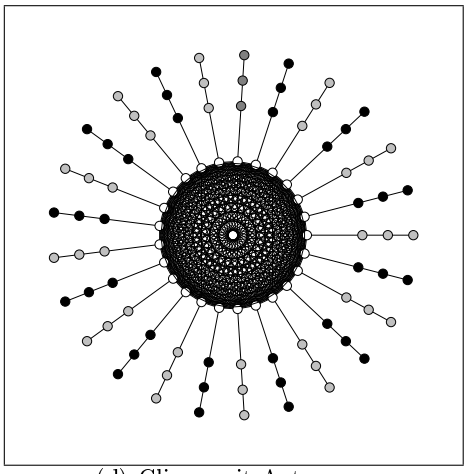
(a) Baum mit 40 Knoten.



(b) Cliquen-Spannbaum mit 37 Knoten.



(c) Gitter.



(d) Clique mit Antennen.

Abbildung 4.2: Beispiele für verwendete Testgraphen

5 Experimentelle Auswertung der Algorithmen

Die wichtigsten der in Kapitel 3 beschriebenen Algorithmen wurden mit den in Kapitel 4 verwendeten Testgraphen getestet. Dieses Kapitel befasst sich mit den Aufbau und den Ergebnissen dieser Tests sowie deren Auswertung.

5.1 Aufbau der Experimente

Alle Algorithmen wurden in Java implementiert. Dafür wurde die Java-Graphen-Bibliothek yFiles [7] mit Erweiterungen des Instituts für Theoretische Informatik benutzt. Zum Lösen der ILPs und LPs wurde auf das C-Programm lp_solve [8] zurückgegriffen. Die Berechnung fand auf dem PC compute5 des ITI Wagner statt, einem AMD Opteron-Zweiprozessorsystem mit 2,6-GHz-Prozessoren und 16 GB RAM.

Die Algorithmen, die einen Schwellwert benötigen wurden mit den Schwellwerten zwischen 0,1 und 0,6 in 0,1er-Schritten getestet. Gewertet wurde jeweils das beste Ergebnis.

5.1.1 Auswahl der Testgraphen

Neben den vier natürlichen Testgraphen, die auch schon bei früheren Arbeiten zur Graphen-Clusterung verwendet wurden, wurde eine Reihe künstlicher Graphen erstellt. Dabei handelt es sich um möglichst verschiedenartige Graphen, die ein breites Spektrum an Graphen repräsentieren sollen. Die meisten Testgraphen davon haben eine regelmäßige Struktur oder repräsentieren bestimmte Graphen-Typen wie zum Beispiel einen Baum. Um das Testfeld abzurunden wurden zusätzlich zwei Graphen mit zufälliger Struktur erstellt. Sie repräsentieren Graphen, bei denen intuitiv eine Clusterung nicht erkennbar ist; eine gute Clusterung ist somit für den Computer vermutlich auch schwer zu finden.

5.2 Ausführliche Auswertung

Bei einigen Graphen erhält man schon mit der Lösung des relaxierten Problems die optimale Lösung - unabhängig davon, ob diese optimale Lösung einen hohen oder niedrigen Modularity-Index liefert. Wir betrachten diese zwei Fälle getrennt voneinander.

Graph	Runden	opt. Runden	Greedy	Runden+Greedy
Dolphin	0,526	0,528	0,492	**0,529**
Football	0,495	0,497	0,465	**0,508**
Karate	**0,431**	**0,431**	0,397	**0,431**
Polbooks	0,517	0,523	0,502	**0,527**
Baum	0,645	0,672	**0,692**	0,689
Cliquen-Baum-Klein	**0,759**	**0,759**	**0,759**	**0,759**
Cliquen-Baum-Groß	**0,858**	**0,858**	**0,858**	**0,858**
Gitter-5 $\times$ 7	0,408	0,482	0,517	**0,518**
Kreis-30	**0,633**	**0,633**	0,620	**0,633**
Clique-Antenne-15-2	**0,192**	**0,192**	0,161	**0,192**
Clique-Antenne-25-2	**0,131**	**0,131**	0,103	**0,131**
Clique-Antenne-25-3	**0,238**	**0,238**	0,160	**0,238**
Zufall-Groß	0,117	0,100	**0,274**	0,241
Zufall-Klein	0,467	0,458	**0,488**	**0,488**
Zufalls-Cluster-Groß	**0,831**	**0,831**	0,823	**0,831**
Zufalls-Cluster-Klein	**0,809**	**0,809**	0,806	**0,809**

Tabelle 5.1: Übersicht der verwendeten Graphen. Beste Ergebnisse sind hervorgehoben.

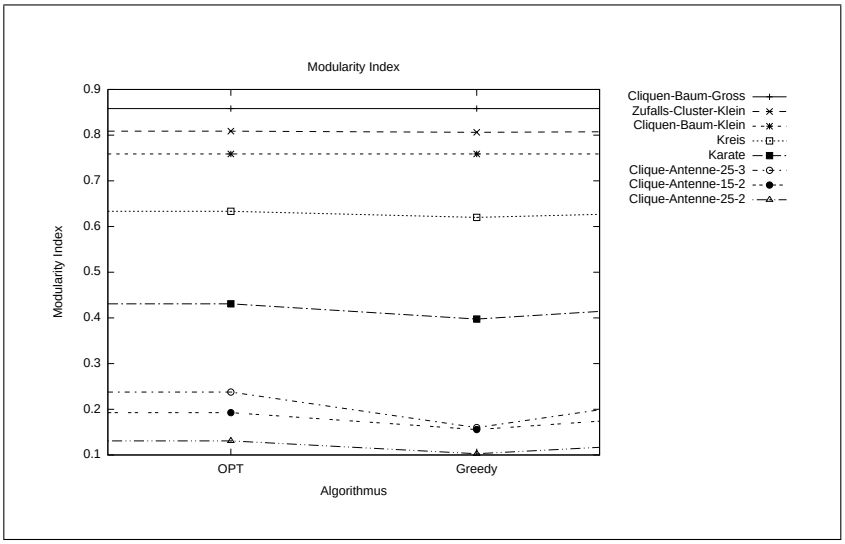
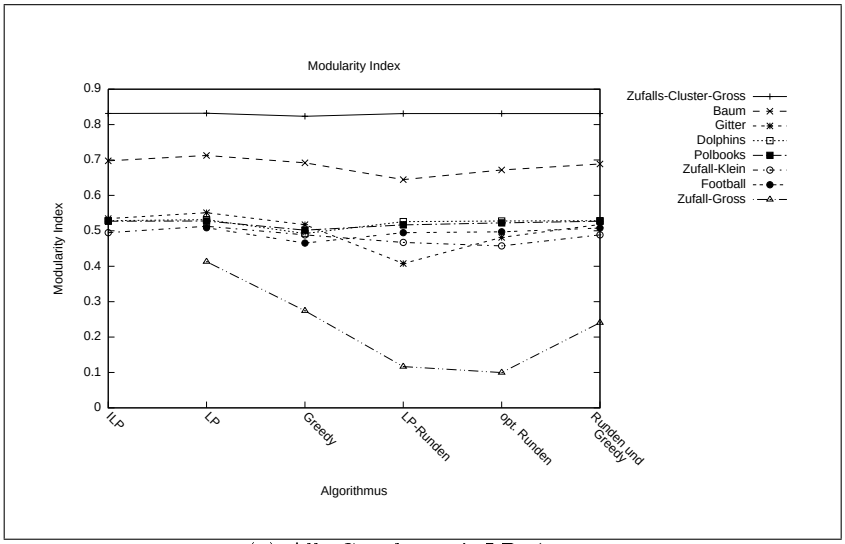
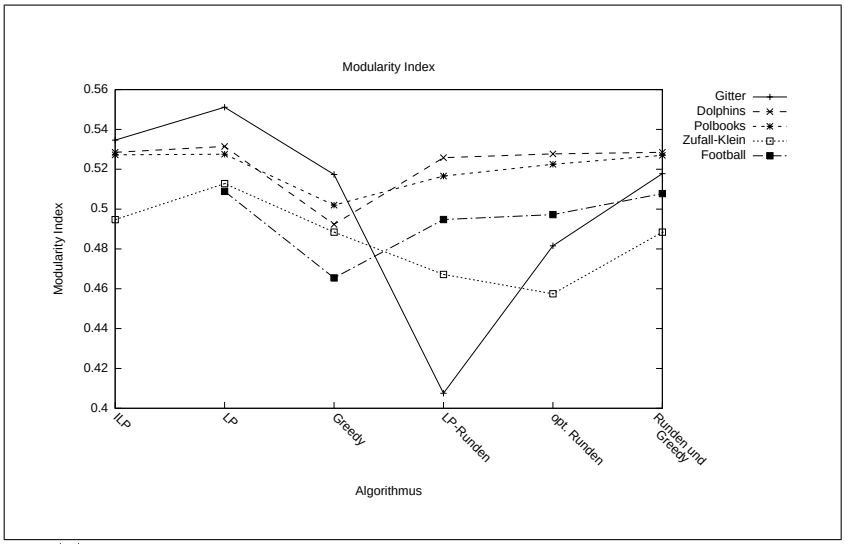


Abbildung 5.1: Ergebnisse des Greedy-Algorithmus im Vergleich zum Optimum für Graphen mit LP = ILP.



(a) Alle Graphen mit LP $\neq$ ILP.



(b) Vergrößerung mit den Graphen mit ähnlichen Modularity-Index.

Abbildung 5.2: Modularity-Index der Graphen mit LP $\neq$ ILP.

5.2.1 Graphen mit LP = ILP

Die Graphen, deren Lösungen von LP und ILP gleich sind, sind in Abbildung 5.1 zu sehen. Unter ihnen befindet sich lediglich ein natürlicher Graph, der Graph *Karate*. Der Greedy-Algorithmus liefert in dieses Fällen auch gute Ergebnisse, kann aber nur in einem Fall, nämlich beim Graphen *Cliquen-Baum-Groß*, eine optimale Lösung finden.

5.2.2 Graphen mit LP ≠ ILP

Die Ergebnisse der anderen Testgraphen, in denen noch gerundet werden muss, kann man in Abbildung 5.2 sehen.

Unter den künstlichen Graphen liegen mit Ausnahme des Graphen *Zufall-Groß* die Werte hier sehr dicht beieinander. Nur bei dem Graphen *Zufall-Groß* ist der Greedy-Algorithmus den Rundungsstrategien deutlich überlegen. Aber auch bei den anderen künstlichen Graphen ist der Greedy-Algorithmus erfolgreich und liefert ähnlich gute Ergebnisse wie die LP-Rundungsstrategien, in zwei Fällen ist er sogar besser.

Auffällig ist, dass der einfache Schwellwertalgorithmus beim Graphen *Gitter* ein deutlich schlechteres Ergebnis liefert. Die Ursache hierfür ist vermutlich, dass die meisten Knoten bei der LP-Lösung eine Distanz von 0,5 haben und somit die weitere Clusterung sehr zufällig verläuft. Werden zwei Knoten miteinander verbunden, die bei einer optimalen Lösung nicht in einen Cluster gehören, dann kann das eine gute Clusterung verhindern. Diese Situation ist in Abbildung 5.3 dargestellt.

Bei den natürlichen Graphen schneiden die Rundungs-Strategien immer besser ab als der einfache Greedy-Algorithmus; die Unterschiede sind auch hier allerdings gering. Das einfache LP-Runden liefert in den meisten Fällen schon sehr gute Ergebnisse. Das Ergebnis liegt nur bei zwei Testgraphen, bei *Zufall-Groß* und *Baum*, unter 90 % der optimalen Lösung. Hier wurde jeweils das beste Ergebnis der getesteten Schwellwerte verwendet.

Insgesamt kann man im Vergleich der verschiedenen Rundungsstrategien ein einheitliches Bild beobachten: Mit Ausnahme der Zufallsgraphen hat der verbesserte Rundungsalgorithmus, der den optimalen Schwellwert selbst bestimmt, gegenüber dem einfachen Schwellwertrunden einen leicht höheren Modularity-Index. Das erklärt sich durch die arithmetische Ungenauigkeit der LP-Lösung; hier ist es Zufall, ob beispielsweise bei einem Schwellwert von 0,5 zwei Knoten mit der Distanz 0,5 verbunden werden oder nicht.

Eine weitere Verbesserung des Modularity-Index' zeigt sich bei allen Graphen beim kombinierten Verfahren aus Runden und Greedy, das hier immer das beste Ergebnis unter allen Rundungsstrategien liefert. Bei den natürlichen Graphen *Dolphins* und *Football* wurde sogar eine optimale Lösung gefunden.

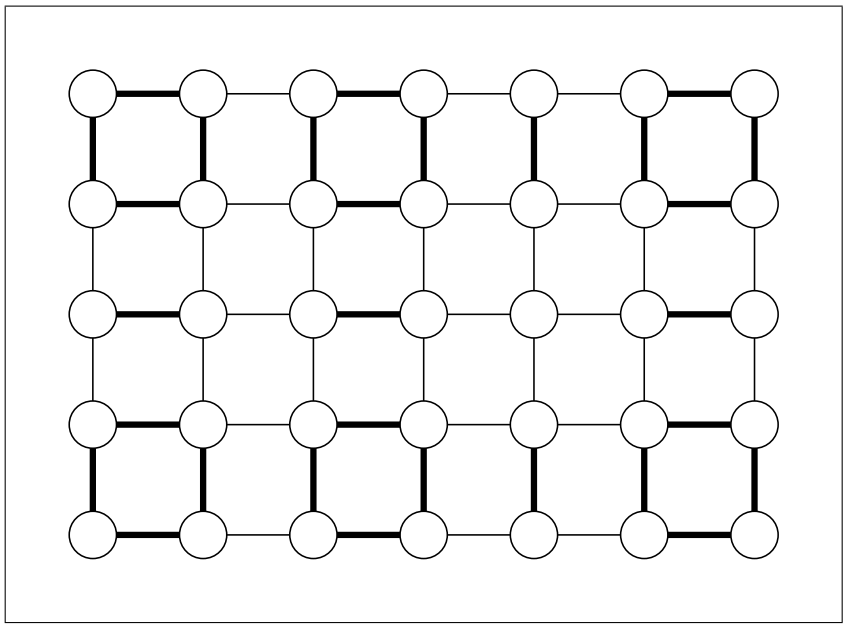


Abbildung 5.3: LP-Lösung des Graphen Gitter. Fette kanten repräsentieren Distanz 0, dünne Kanten die Distanz 0,5.

5.2.3 Laufzeiten der Heuristiken

Der Greedy-Algorithmus hat die mit Abstand beste Laufzeit und lag bei allen Graphen im einstelligen Millisekunden-Bereich.

Interessant ist der Vergleich der Laufzeiten zur Berechnung der LP- und ILP-Lösung. Für Graphen, in denen schon die LP-Lösung ein optimales Ergebnis lieferte, war die Laufzeit für die Lösung der ILP-Instanz in allen Fällen sogar etwas niedriger, der Unterschied war aber sehr gering und lag bei maximal 2,3 %.

Bei den Graphen mit unterschieden zwischen der LP und der ILP-Lösung lag die Laufzeit zur Lösung des ILPs um ein vielfaches über der des LPs, bei den Graphen Zufall-Groß und Football (die Berechnung der LP-Version dauerte 12,6 Stunden) musste die Berechnung nach einer Woche abgebrochen werden, da bis dahin kein Ergebnis vorlag.

5.3 Zusammenfassung der Ergebnisse

Bei einigen Graphen erhält man schon mit der Lösung des relaxierten Problems die optimale Lösung. In diesen Fällen ergab sich allerdings bei der relativ niedrigen Laufzeit kein Vorteil gegenüber der Berechnung der ILP-Lösung; die Laufzeiten lagen hier sogar geringfügung über der von der ILP-Version. Die betreffenden Graphen sind allesamt

künstliche Graphen: Clique-Antenne-X-Y, die Cliquen-Spannbäume, Kreis und Zufall-Klein.

Insgesamt zeigte sich die Kombination aus Runden und Greedy als deutlich bester Algorithmus. Der einfache Greedy-Algorithmus lieferte nur beim Graphen *Baum* einen unwesentlich besseres Ergebnis und eine kleine Verbesserung beim Graphen Zufall-Groß.

6 Zusammenfassung und Ausblick

Da die Berechnung einer Optimalen Clusterung bezüglich des Modularity-Index' $\mathcal{NP}$-Schwer ist, wurden Heuristiken zur Approximation einer guten Clusterung gesucht. Dabei wurden insbesondere Algorithmen, die auf der Relaxierung des als ILP formulieren Problems beruhen, untersucht.

Es hat sich gezeigt, dass bereits einfache Algorithmen, bezogen auf den Modularity-Index, meist relativ gute Ergebnisse liefern. Allerdings ist es schwierig, mit einfachen schnellen Algorithmen noch bessere Ergebnisse zu finden.

Nach den Ergebnissen aus den Tests ist es vermutlich erforderlich die aufwändige Berechnung der LP-Lösung durchzuführen, um sich der optimalen Lösung weiter zu nähern. Die relaxierte LP-Lösung hat sich als sinnvolle Grundlage für Heuristiken zur Optimierung des Modularity-Index' erwiesen, die einfachen Clusterungs-Strategien in der Regel überlegen sind.

Zusammenfassend kann gesagt werden, dass von den auf der relaxierten LP-Lösung aufbauenden Rundungsalgorithmen vor allem die Kombination aus LP-Runden und Greedy-Algorithmus die meistversprechendsten Lösungen geliefert hat. Aber auch LP-Rundungsstrategien, die auf einem Schwellwert basieren, zeigten schon gute Ergebnisse. Insbesondere der Algorithmus, der den optimalen Schwellwert durch Sortieren findet, könnte vermutlich noch verbessert werden, indem die Reihenfolge der Verschmelzungen innerhalb einer Menge von Knotenpaaren mit gleicher Distanz geschickter und gezielter bestimmt wird.

Die Berechnung der Lösung der relaxierten LP-Formulierung des Problems ist trotz der Vereinfachung noch sehr zeitaufwändig. Hat man aber die Lösung des relaxierten Problems einmal berechnet, so ist es sinnvoll auf Grundlage dieses Ergebnisses verschiedene Rundungsstrategien zu testen und das beste Ergebnis unter ihnen auszuwählen, da die hier verwendeten Rundungsalgorithmen selbst eine sehr niedrige Laufzeit haben.

Die Entwicklung von alternativen Strategien zur Approximation eines guten Modularity-Index wäre interessant, um noch besser Einschätzen zu können, wie wichtig die Berechnung der LP-Relaxierung zur Lösung des Problems ist. Außerdem wäre es hilfreich theoretische Schranken für die Approximationsalgorithmen zu berechnen, um die Qualität der unterschiedlichen Strategien objektiv vergleichen zu können.

Literaturverzeichnis

[1] Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(026113), 2004.

[2] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Höfer, Zoran Nikoloski, and Dorothea Wagner. On Modularity - NP-Completeness and Beyond. Technical Report 2006-19, ITI Wagner, Faculty of Informatics, Universität Karlsruhe (TH), 2006.

[3] Aaron Clauset, Mark E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(066111), 2004.

[4] M. E. J. Newman. Detecting community structure in networks. *European Physical Journal B*, 38:321–330, 2004.

[5] Mark E. J. Newman. Fast algorithm for detecting community structure in networks. Phys. Rev. E 69, 066133 (2004), September 2003.

[6] M. E. J. Newman. Modularity and community structure in networks. *PROC.NATL.ACAD.SCI.USA*, 103:8577, 2006.

[7] yWorks GmbH. yfiles. `http://www.yworks.com`, 2007.

[8] Open Source. lp_solve. `http://lpsolve.sourceforge.net`, 2007.