Studienarbeit: Experimentelle Untersuchung von Clusterungsgeneratoren mittels Qualitätsindizes

von
Daniel Delling

Betreuer: Prof. D. Wagner, Marco Gaertler
Dozentin: Prof. D. Wagner

ILKD Prof. D. Wagner, Universität Karlsruhe

Sommersemester 2004

Zusammenfassung

Die Gruppierung von Elementen in Strukturen ist ein Problem, dass ein lange Tradition in der menschlichen Entwicklung hat. Die mathematische Modellierung mit Hilfe von Graphen erscheint unproblematisch. Doch die Computer gestützte Gruppierung oder Clusterung von Knoten eines Graphen erweist sich als schwierig, da das Bewertungsmaß für Clusterungen häufig die Intuition ist. Um dieses wichtige Problem trotzdem lösen zu können, gibt es mehrere Verfahren, von denen manche darauf basieren, Clusterungen sukzessiv zu verbessern. Daher benötigt man Bewertungsfunktionen, so genannte Indizes, für Clusterungen. Allerdings ist hierbei problematisch, dass bisher kein Index existiert, der jede Clusterung jedes Graphen der Intuition entsprechend bewertet. Statt dessen existieren mehrere Indizes mit Vor- und Nachteilen.

Wir untersuchten verschiedene Graphentypen mit vorgegebener intuitiv gut erscheinender Clusterung, um für die Sukzessiv-Verfahren Anhaltspunkte zu geben, auf welchen Typen von Graphen welche Indizes am besten der Intuition entsprechen. Außerdem untersuchten wir Graphen, bei denen die Anzahl der möglichen Clusterungen stark eingegrenzt ist.

Danksagung

Ich möchte Prof. D. Wagner und Marco Gaertler für die sehr nette und umfangreiche Betreuung danken.

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

Kapitel 1

Einleitung

Das Finden von Gruppen, so genannten Clustern, in Strukturen ist ein wichtiges Optimierungsproblem mit vielen Anwendungen. Das Humane Genom zum Beispiel ist aufgeschlüsselt und codiert 45.000 Proteine, wovon ungefähr 6.000 bekannt sind. Die große Aufgabe ist momentan, Gruppen von Proteinen zu finden, die stark untereinander agieren, aber nur wenig mit anderen Proteinen. Bei einzelnen Tieren mit weniger Proteinen ist dies schon teilweise gelungen [1]. Die Erstellung solcher Interaktionsdiagramme scheint ein vielversprechender Ansatz zu sein, um bei speziellen Krankheiten zu bestimmen, inwieweit nur einzelne Gruppen in ihrer Funktion gestört sind. Wenn zum Beispiel nur eine Gruppe Fehlfunktionen aufweißt, alle anderen Proteine aber voll funktionsfähig sind, könnte man versuchen Medikamente zu entwickeln, die nur die Fehlfunktionen dieser Proteingruppe beseitigen. Aber auch bei der Diagnose und Früherkennung von Krankheiten könnten solche Diagramme hilfreich sein.

Man kann dieses Problem als Clusterungsproblem eines Interaktionsgraphen modellieren: Die Proteine sind die Knoten des Graphen, eine Kante besteht zwischen zwei Knoten genau dann, wenn sie miteinander agieren. Der Interaktionsgraph des Humanen Genoms besteht dem entsprechend aus circa 45.000 Knoten. Momentan ist die Anzahl der Kanten noch nicht bekannt, da noch nicht alle Interaktionen zwischen Proteinen erforscht sind. Nichtsdestotrotz ist es bei solch großen Instanzen zwingend notwendig, diese Gruppierung oder auch Clusterung der Proteine mit Hilfe von Computer durchzuführen.

Aber auch in der Datenanalyse [2] bei Physikern oder im Bankenwesen spielt Clusterung eine große Rolle. Generell ist bei der Clusterung schwer zu sagen, welche Clusterung eines Graphen die Beste ist. Das Bewertungsmaß ist in diesem Falle häufig die Intuition, sodass man häufig davon spricht, dass eine

Clusterung intuitiv gut erscheint. Es stellt sich nun das Problem, dass man computergestützt eine Clusterung finden möchte, die man intuitiv als gut bezeichnen würde.

Einige Ansätze um dieses Problem in den Griff zu bekommen, basieren darauf, eine beliebige Initialclusterung zu erzeugen und diese dann sukzessiv zu verbessern. Da diese Verfahren Clusterungen vergleichen, benötigt man eine Bewertungsfunktion, so genannte Indizes. Leider existiert bisher kein Index, der der Intuition entsprechend Clusterungen bewertet. Statt dessen existieren mehrere Indizes mit verschiedenen Vor- und Nachteilen, von denen vier häufig für diese Verfahren verwendet werden.

Wir generierten verschiedene Graphtypen, die eine intuitive Clusterung durch die Art und Weise, wie die Graphen generiert werden, induzieren. Diese Clusterungen bewerteten wir mit vier bekannten Indizes, um zu analysieren, auf welchen Graphtypen welche Indizes am ehesten der Intuition entsprechen. Diese Informationen könnten dann genutzt werden, um bei den Sukzessiv-Verfahren für bestimmte Graphtypen die entsprechenden Indizes zu benutzen. Wir ermittelten experimentell, dass sämtliche untersuchten Graphen mit zugehöriger intuitiver Clusterung mit einer Kombination von zwei der vier Indizes sehr gut bewertet werden. Um dieses Ergebnis zu bestätigen, generierten wir Graphen, bei denen man mehrere Clusterungen intuitiv als gut bezeichnen könnte, wobei ein bis zwei Clusterungen besser als die anderen erscheinen.

In Kapitel 2 werden wir wichtige Begriffe definieren und einige Grundlagen zum Thema Clusterung schaffen. Kapitel 3 wird sich mit den Clusterindizes selbst und ihren Vor- und Nachteilen beschäftigen. Die Graphtypen, die wir untersuchten, werden in Kapitel 4 vorgestellt. In Kapitel 5 werden wir Testreihen für Graphinstanzen der Graphen aus Kapitel 4 mit 500, 1000 und 5000 Knoten diskutieren. Die Ergebnisse werden in Kapitel 6 vorgestellt und in Kapitel 7 werden wir eine Zusammenfassung der Ergebnisse liefern, sowie einen Ausblick auf mögliche Vertiefungen des Stoffes geben.

Kapitel 2

Grundlagen

Alle hier untersuchten Graphen sind ungerichtet und einfach (d.h. keine Schleifen und parallelen Kanten). Es werden Grundlagen zu Graphen[3] und der Komplexitätstheorie[4] als bekannt vorausgesetzt.

Im ungewichteten Fall bezeichnen wir einen Graphen mit $G = (V, E)$ und im gewichteten Fall mit $G = (V, E, w)$, wobei $w : E \to \mathbb{R}$ eine Gewichtsfunktion ist, die jeder Kante e des Graphen ein Gewicht $w(e)$ zuordnet. Die Anzahl der Knoten V des Graphen G wollen wir mit n, die Anzahl der Kanten mit m bezeichnen. Das Gesamtgewicht aller Kanten des gewichteten Graphen bzeichnen wir mit $w(G)$. Des Weiteren bezeichnen wir mit $A(G)$ die Adjazenzmatrix des Graphen G.

Neben der bekannten Adjazenzmatrix benötigen wir noch die normalisierte Adjazenzmatrix $M(G)$.

Definition 1 *Sei $A(G)$ die Adjazenzmatrix des Graphen G und $D(G) \in \mathbb{N}^{n \times n}$ eine Diagonalmatrix, deren Einträge d_{ii} dem Knotengrad des Knoten i des Graphen entsprechen. Die Matrix $M(G) = D(G)^{-1}A(G)$ bezeichnen wir als normalisierte Adjazenzmatrix des Graphen G.*

Um uns intensiver mit Clusterung beschäftigen zu können, benötigen wir eine Definition für einen Clustergraphen.

Definition 2 *Ein Graph G heißt Clustergraph, falls G eine knotendisjunkte Vereinigung von Cliquen ist.*

Ein Clustergraph besteht also aus Cliquen. Somit ist ein Clustergraph durch eine Zuordnung der Knoten in Cliquen eindeutig definiert. Eine solche Einteilung der Knoten nennt man Partition.

Definition 3 *Eine Aufteilung von V in paarweise disjunkte, nichtleere Teilmengen V_i nennt man Partition $P = (V_1, \ldots, V_p)$ von V.*

Wir betrachten in unserem Falle aber nicht Graphen, die ausschließlich aus knotendisjunkten Cliquen bestehen, sondern wollen untersuchen, wie „ähnlich“ ein beliebiger Graph G einem Clustergraphen ist. Hierfür kann als Maß gelten, wieviele Kantenoperationen (entfernen oder hinzufügen von Kanten) man auf dem Graph G ausführen muss, um einen Clustergraphen zu erhalten. Die Minimierung der Anzahl der Kantenoperationen ist NP-vollständig und wird in der Literatur als *Cluster Editing Problem* bezeichnet [5].

Problem 1 (Cluster Editing Problem) *Sei $G = (V, E)$ ein ungerichteter Graph. Gesucht ist eine Kantenmenge F mit $|F|$ minimal, sodass $G' = (V, E\Delta F)$ mit $E\Delta F := (E \setminus F) \cup (F \setminus E)$ ein Clustergraph ist.*

Analog zu der Definition der Partition ist der Begriff Clusterung eines beliebigen Graphen definiert. Eine Clusterung eines Graphen ist eine Einteilung der Knoten des Graphen in verschiedene Mengen, die in dem transformierten Clustergraphen die Cliquen darstellen sollen.

Definition 4 *Eine Einteilung von V eines Graphen G in $(C_1, \ldots, C_p)$ mit C_i, C_j paarweise disjunkt nennt man Clusterung $\mathcal{C}$ des Graphen G. Die C_i einer Clusterung $\mathcal{C}$ werden als Cluster bezeichnet.*

Mit dieser Definition einer Clusterungen können wir einen Schnitt als eine Clusterung des Graphen mit zwei Clustern auffassen.

Definition 5 *Der Schnitt eines Graphen $G = (V, E)$ ist eine Clusterung des Graphen G mit $\mathcal{C} = (C, V \setminus C)$.*

Cluster eines Graphen G induzieren ähnlich zu Knotenmengen des Graphen einen Subgraphen von G.

Definition 6 *Gegeben seien ein Graph $G = (V, E)$ und eine Clusterung $\mathcal{C} = (C_1, \ldots, C_p)$ des Graphen G. Den Graph $G[C_i] = (C_i, E_G[C_i])$ bezeichnen wir als clusterinduzierten Subgraphen.*

Man kann die Kanten eines Graphen und einer zugehörigen Clusterung $\mathcal{C} = (C_1, \ldots, C_p)$ in zwei Typen unterscheiden. Solche Kanten, die innerhalb eines Clusters zwei Knoten verbinden, oder solche Kanten, die zwei Knoten verschiedener Cluster verbinden.

Definition 7 *Gegeben seien ein Graph $G = (V, E)$ und eine Clusterung $\mathcal{C} = (C_1, \ldots, C_p)$ des Graphen G. Man bezeichnet*

$$E_{inter} = \big\{\{u, v\} \in E \mid u \in C_i, v \in C_j, i \neq j\big\}$$

als Intercluster Kanten und analog

$$E_{intra} = \big\{\{u, v\} \in E \mid u, v \in C_i\big\}$$

als Intracluster Kanten.

Im Folgenden wollen wir mit $m(\mathcal{C})$ die Anzahl der Intracluster Kanten und mit $\overline{m}(\mathcal{C})$ die Anzahl der Intercluster Kanten einer Clusterung bezeichnen. Für gewichtete Graphen bezeichnet zusätzlich $w(\mathcal{C}) = \sum_{e \in E_{intra}} w(e)$ das Gesamtgewicht der Intracluster Kanten und $\overline{w}(\mathcal{C}) = \sum_{e \in E_{inter}} w(e)$ das Gesamtgewicht der Intercluster Kanten.

Kapitel 3

Indizes

Ein Ansatz, um die intuitiv beste Clusterung eines Graphen zu finden, berechnet eine Initialclusterung und versucht diese sukzessiv zu verbessern. Um Clusterungen eines Graphen zu vergleichen, benötigt man einen Index, der eine Clusterung bewertet. Dieser Index soll Clusterungen mit einem Wert zwischen 0 und 1 bewerten, wobei ein Wert nahe 1 eine - bezüglich dieses Index - gute Clusterung und 0 eine Schlechte auszeichnen soll.

Allerdings ist in der Literatur kein Index zu finden, der im allgemeinen Clusterungen so bewertet, wie dies intuitiv als richtig erscheint. Vielmehr existieren mehrere Indizes, die jeweils Vor- und Nachteile besitzen. Wir wollen im Folgendem vier dieser Indizes analysieren und Vor- bzw. Nachteile der jeweiligen Indizes diskutieren.

Allen hier vorgestellten Indizes ist gemein, dass die Maximierung des Indexwertes im Allgemeinen NP-schwer ist. Da wir aber Graphen mit gegebener intuitver Clusterung untersuchten, wollen wir auf diese Problematik nicht weiter eingehen.

3.1 Coverage

Der ungewichtete *Coverage* Index ist das Verhältnis von Kanten innerhalb der Cluster zu der Gesamtanzahl der Kanten des Graphen.

$$coverage_u(\mathcal{C}) := \frac{m(\mathcal{C})}{m} = \frac{m(\mathcal{C})}{m(\mathcal{C}) + \overline{m}(\mathcal{C})}$$

Der Coverage Index wird im Allgemeinen als schlecht bezeichnet, da er über

die Dichte der einzelnen Cluster wenig aussagt. Des Weiteren wird die triviale Clusterung $\mathcal{C} = \{V\}$ immer mit 1 bewertet.

Ferner hat ein minimaler Schnitt des Graphen immer einen sehr hohen Coverage Wert, doch der minimale Schnitt eines Graphen ist im Allgemeinen keine gute Clusterung des Graphen. Bei einem Graphen, der aus drei knotendisjunkten Cliquen besteht, Clique 1 hat zwei Kanten zu Clique 2 und eine Kante zu Clique 3, ist der minimale Schnitt des Graphen die Aufteilung in Clique 1 vereinigt Clique 2 und Clique 3. Die Aufteilung in 3 Cluster - den Cliquen entsprechend - scheint aber die bessere Clusterung zu sein.

Es müssten also weitere Forderungen eingeführt werden, wie zum Beispiel, dass die Clusterung aus mindestens p Cluster bestehen muss. Aber auch hier können Beispielgraphen und -clusterungen erzeugt werden, die einen hohen Coverage Wert besitzen, aber schlechte Clusterungen sind. Die Clusterung des Graphen in Abbildung 3.1 bewertet der Coverage Index mit 0,9, obwohl man solch eine Clusterung intuitiv nicht als gut bezeichnen würde.

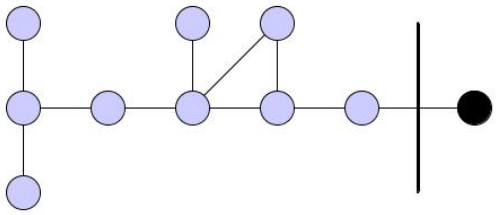

Abbildung 3.1: Eine schlechte Clusterung mit hohem Coverage Index

Der Graph an sich ist nicht gut clusterbar. Für solche Graphen möchte man, dass ein Index allen denkbaren Clusterungen schlechte Werte zuordnet. Dies ist bei dem Coverage Index offensichtlich nicht gegeben.

Gewichteter Coverage Index Man kann für gewichtete Graphen auch eine gewichtete Version des Coverage Index definieren. Hierbei wird das Gesamtgewicht der Intracluster Kanten ins Verhältnis zum Gesamtgewicht aller Kanten des Graphen G gesetzt.

$$coverage_g(\mathcal{C}) := \frac{w(\mathcal{C})}{w(G)} = \frac{w(\mathcal{C})}{w(\mathcal{C}) + \overline{w}(\mathcal{C})}$$

Die gewichtete Version des Coverage Index beseitigt nicht die Nachteile der ungewichteten Version.

3.2 Performance

Der *Performance* Index von einer Clusterung zählt die Anzahl der „korrekt interpretierten Knotenpaaren“. Man addiert die Anzahl der Intracluster Kanten zu der Anzahl der Knotenpaare, die richtigerweise nicht verbunden sind, da die Knoten des jeweiligen Paares verschiedenen Clustern zugeordnet sind. Dieser Wert wird in das Verhältnis mit der Anzahl der Kanten in einem vollständigen Graphen mit n Knoten gesetzt.

$$performance_u(\mathcal{C}) := \frac{m(\mathcal{C}) + \sum_{\{v,w\}\notin E, v\in C_i, w\in C_j, i\neq j} 1}{\frac{1}{2}n(n-1)}$$

Die Berechnung des Performance Index nach dieser Formel besitzt quadratische Laufzeit in der Anzahl der Knoten des Graphen. Aus diesem Grund ist folgende Umformung unter Laufzeitaspekten günstig:

$$\begin{aligned}
performance_u(\mathcal{C}) &= \frac{m(\mathcal{C}) + \sum_{\{v,w\}\notin E, v\in C_i, w\in C_j, i\neq j} 1}{\frac{1}{2}n(n-1)} \\
&= \frac{2m(\mathcal{C}) + 2\big(\frac{1}{2}n(n-1) - 2\overline{m}(\mathcal{C}) - \sum_{i=1}^{k} |C_i|(|C_i|-1)\big)}{n(n-1)} \\
&= \frac{n(n-1) - 2\overline{m}(\mathcal{C}) + 2m(\mathcal{C}) - \sum_{i=1}^{k} |C_i|(|C_i|-1)}{n(n-1)} \\
&= 1 - \frac{2\overline{m}(\mathcal{C}) - 2m(\mathcal{C}) + \sum_{i=1}^{k} |C_i|(|C_i|-1)}{n(n-1)}
\end{aligned}$$

Eine Clusterung eines Graphen hat also einen hohen Performance Index, wenn sehr viele Intracluster und wenige Intercluster Kanten existieren. Zum Beispiel wird die Clusterung eines Clustergraphen - bestehend aus knotendisjunkten Cliquen - von dem Performance Index mit 1 bewertet. Aber auch bei diesem Index gibt es bereits kleine Graphen, bei denen der Performance Index nicht ganz optimale Ergebnisse liefert. Die linke Clusterung des Graphen in Abbildung 3.2 hat den Wert von ungefähr 0,89. Die intuitiv besser ercheinende rechte Clusterung des Graphen hat hingegen nur den Wert von circa 0,71.

Nichtsdestotrotz ist der Performance Index ein recht guter Index, der selten schlechten Clusterungen gute Wert zuordnet.

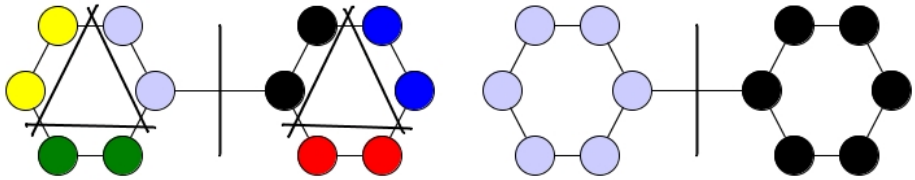

Abbildung 3.2: links die schlechtere und rechts die bessere Clusterung

Gewichtete Performance Man kann auch eine gewichtete Version des Performance Index definieren. Hierfür benötigt man noch das maximale Kantengewicht $W = \max_{e \in E}(w(e))$ des gewichteten Graphen $G = (V, E, w)$.

$$performance_g(\mathcal{C}) := \frac{w(\mathcal{C}) + W \sum_{\{v,w\} \notin E, v \in C_i, w \in C_j, i \neq j} 1}{W \frac{1}{2} n(n-1)}$$

Aus gleichen Gründen, wie bei dem ungewichteten Perfomance Index, empfiehlt es sich, anstatt den Index mit dieser Formel zu berechnen, die Formel umzuformen. Dies führt zu folgendem Ergebnis:

$$performance_g(\mathcal{C}) = 1 - \frac{2\overline{w}(\mathcal{C}) - 2w(\mathcal{C}) + W \sum_{i=1}^{k} |C_i|(|C_i| - 1)}{Wn(n-1)}$$

Bei der gewichteten Version des Performance Index wird das maximale Kantengewicht benutzt. Dies kann zu ungewöhnlichen und unerwünschten Ergebnissen führen, wenn zum Beispiel der Graph sehr viele Kanten mit geringem und wenige Kanten mit hohem Gewicht besitzt. Dadurch wird der Graph mit einem vollständigen Graphen - bestehend aus n Knoten und $\binom{n}{2}$ Kanten, die alle das maximale Gewicht W besitzen - verglichen.

3.3 Intercluster Conductance

Die *Leitfähigkeit* (Conductance) eines Schnittes $\mathcal{C} = (C, V \setminus C)$ vergleicht die Anzahl der Kanten im Schnitt mit der Anzahl der Kanten in einem der beiden durch den Schnitt induzierten Subgraphen.

$$\phi(C) := \begin{cases} 1 & \text{für } C \in \{\emptyset, V\} \\ 0 & \text{für } C \notin \{\emptyset, V\} \text{ und } \overline{m}(\mathcal{C}) = 0 \\ \dfrac{\overline{m}(\mathcal{C})}{\min\left(\sum_{v \in C} \deg v, \sum_{v \in V \setminus C} \deg v\right)} & \text{sonst} \end{cases}$$

Ein Schnitt besitzt also eine geringe Leitfähigkeit, wenn die Größe des Schnittes im Verhältnis zur Dichte der beiden durch den Schnitt induzierten Subgraphen klein ist.

Mit Hilfe der Leitfähigkeit läßt sich nun der Intercluster Conductance Index $\delta(\mathcal{C})$ definieren.

$$\delta(\mathcal{C}) := 1 - \max_{i \in \{1,\ldots,k\}} \phi(C_i)$$

Zusätzlich gilt, dass der Intercluster Conductance Wert für den gesamten Graphen immer 1 ist.

Eine Clusterung besitzt also einen hohen Intercluster Conductance Index, wenn jeder Cluster wenig Kanten zu anderen Clustern besitzt. Den Nachteil dieses Index soll die Abbildungen 3.3 verdeutlichen. Beide Clusterungen haben ein Intercluster Conductance Wert von 0,875. Die rechte Clusterung scheint aber eine bessere Aufteilung des Graphen zu sein.

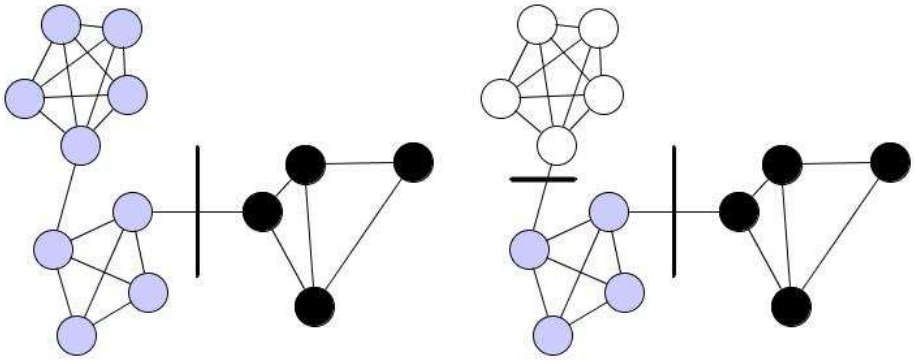

Abbildung 3.3: links eine schlechte und rechts eine gute Clusterung

Eine Clusterung mit hoher Intercluster Conductance Bewertung sagt also anscheinend nicht viel über die Beschaffenheit der einzelnen Cluster aus, sodass dieser Index als alleiniges Bewertungsmaß für Clusterungen suboptimal erscheint.

Des Weiteren treten Probleme bei inhomogenen Clustergrößen auf. Da wir aber Graphen mit homogenen Clustergrößen untersuchten, betrachten wir diese Problematik hier nicht.

Intercluster Conductance gewichtet Man kann analog einen gewichteten Intercluster Conductance Index definieren. Hierfür führen wir zunächst

die gewichtete Leitfähigkeit eines Schnittes $\mathcal{C} = (C, V \setminus C)$ ein.

$$\phi_g(C) := \begin{cases} 1 & \text{für } C \in \{\emptyset, V\} \\ 0 & \text{für } C \notin \{\emptyset, V\} \text{ und } \overline{w}(\mathcal{C}) = 0 \\ \frac{\overline{w}(\mathcal{C})}{\min\left(\sum_{v \in C} \deg_g v, \sum_{v \in V \setminus C} \deg_g v\right)} & \text{sonst} \end{cases}$$

Hierbei entspricht $\deg_g v = \sum_{\{v,u\} \in E, u \in V} w(\{v, u\})$ dem gewichteten Knotengrad des Knoten v.

Somit ist der gewichtete Intercluster Conductance Index $\delta_g(\mathcal{C})$ definiert:

$$\delta_g(\mathcal{C}) := 1 - \max_{i \in \{1,\ldots,k\}} \phi_g(C_i)$$

3.4 Intracluster Conductance

Der Intracluster Conductance Index hat das Ziel, solchen Clusterungen hohe Werte zuzuordnen, deren Cluster sehr dicht sind. Zu diesem Zweck wird für jeden Cluster ein Schnitt gesucht, der minimale Leitfähigkeit hat. Hierzu benötigten wir eine Erweiterung der ϕ Funktion aus Kapitel 3.3 auf einen Graphen G.

$$\phi(G) := \min_{C \subseteq V} \phi(C)$$

Somit ist $\phi(G)$ das Minimum aller Schnitte, die man in G bilden kann. Nun können wir den Intracluster Conductance Index $\alpha(\mathcal{C})$ folgend definieren:

$$\alpha(\mathcal{C}) := \min_{i \in \{1,\ldots,k\}} \phi(G[C_i])$$

Eine Clusterung besitzt also einen hohen Intracluster Conductance Index, wenn jeder Cluster eine hohe Dichte besitzt und keine Flaschenhälse innerhalb der Cluster existieren. Der Intracluster Conductance Index hat aber zwei Nachteile.

1. Der Intracluster Conductance Index macht keine Aussage darüber, wie die Cluster miteinander verbunden sind, da nur die durch die Cluster induzierten Subgraphen betrachtet werden. Die Intercluster Kanten werden also für die Berechnung nicht berücksichtigt.

2. Einem Graphen mit sehr großen Cliquen wird ein Wert von 0,5 zugeordnet, da der Schnitt mit minimaler Leitfähigkeit für eine Clique $\mathcal{C} = (C, V \setminus C)$ mit $|C| = \lfloor n/2 \rfloor$ ist. Für $n \to \infty$ nähert sich der Wert dann 0,5 an. Eine Clique mit drei Knoten wird hingegen mit 1 bewertet. Von diesem Index werden also Clusterungen mit ausschließlich kleinen Cliquen besser als Clusterungen mit ausschließlich großen Cliquen bewertet.

Aus diesen Gründen ist Intracluster Conductance als alleiniger Index für eine Clusterung nicht zu empfehlen, denn auf Grund Punkt 1 und 2 ergibt sich ein unerwünschter Effekt. Eine Clique mit sechs Knoten wird schlechter bewertet, als wenn diese Clique in zwei Cliquen mit je drei Knoten aufgeteilt wird. Dieser Effekt soll durch Abbildung 3.4 verdeutlicht werden.

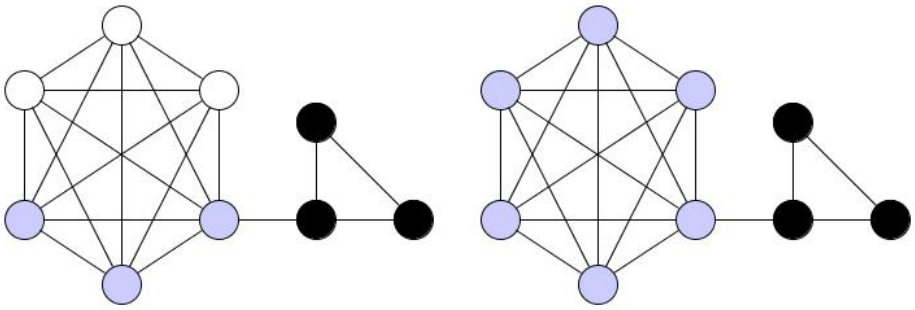

Abbildung 3.4: links eine schlechte und rechts eine gute Clusterung

Die linke Clusterung des Graphen wird mit 1 während die intuitiv bessere rechte Clusterung mit 0,6 bewertet wird.

Gewichtete Intracluster Conductance Wie bei den anderen drei Indizes existiert auch für den Intracluster Conductance Index eine gewichtete Version. Man erweitert, ähnlich zur ungewichteten Version, die gewichtete ϕ-Funktion auf einen gewichteten Graphen G.

$$\phi_g(G) = \min_{C \subseteq V} \phi_g(C)$$

Analog zur ungewichteten Version erhält man somit den gewichteten Intracluster Conductance Index $\alpha_g(\mathcal{C})$:

$$\alpha_g(\mathcal{C}) := \min_{i \in \{1,\ldots,k\}} \phi_g(G[C_i])$$

Auch die gewichtete Version des Intracluster Conductance Index besitzt die Nachteile, die die ungewichtete Version hat.

NP-Vollständigkeit Es sei angemerkt, dass die Berechnung von $\phi(G)$ für einen Graphen G im Allgemeinen NP-schwer ist. Somit ist auch das Berechnen des Intracluster Conductance Index NP-schwer. Wir wollen an dieser Stelle nur die Idee eines poly-logarithmischen Approximationsalgorithmus vorstellen, für genauere Ausführungen siehe [6].

Das Problem bei der Berechnung des Schnittes mit minimaler Leitfähigkeit für einen Graphen ist, dass man sämtliche denkbaren Knotenmengen betrachten muss. Der Approximationsalgorithmus berechnet nun nicht die Leitfähigkeit aller möglichen Schnitte, sondern betrachtet nur eine in der Knotenanzahl lineare Anzahl von Schnitten.

Zunächst wird der zweitgrößte Eigenwert der normalisierten Adjazenzmatrix $M(G)$ des Graphen berechnet. Der dazugehörige Eigenvektor impliziert eine Sortierung der Knoten. Für jede Aufteilung dieser Sortierung in zwei Knotenmengen wird nun der Schnitt im Graphen G berechnet. Der Schnitt mit der minimalen Leitfähigkeit ist nun der approximierte Wert von $\phi(G)$ mit poly-logarithmischer Güte.

Kapitel 4

Graphgeneratoren

Wir haben verschiedene Typen von Graphen generiert, bei denen man eine intuitive Clusterung durch die Art und Weise, wie der Graph generiert wird, angeben kann. Solche Graphen bezeichnen wir als prägeclusterte Graphen. Die Generatoren dieser Graphen stellen wir in diesem Kapitel vor.

4.1 Attraktoren

Die Idee bei einem Attraktorgraphen ist, dass man eine diskrete Aufteilung - wir benutzten eine Gitterstruktur - des Raumes hat. In dieser Gitterstruktur werden einige Knoten zufällig mit einem Mindestabstand verteilt. Diese Knoten sollen nachher die so genannten Clusterknoten bilden.

Generierung Zunächst werden c Knoten zufällig in einem Gitter verteilt, wobei die Knoten einen gewissen Mindestabstand zueinander haben sollten. Diese Knoten sollen als Clusterknoten c_i mit $i = 1, \ldots, n$ bezeichnet werden und werden jeweils einem eigenen Cluster zugeordnet. Daraufhin wird nun für jede Koordinate des Gitters Folgendes untersucht:

1. suche denjenigen Clusterknoten c_j, der dieser Koordinate am nähesten ist
2. bestimme den Abstand d von der Koordinate zu diesem Clusterknoten
3. erstelle mit einer Wahrscheinlichkeit von $1/d$ an dieser Koordinate einen Knoten u und verbinde diesen Knoten mit dem Clusterknoten c_j

4. ordne den Knoten u dem Cluster von c_j zu

5. setze das Gewicht der hinzugefügten Kante auf eins

Nachdem nun alle Koordinaten abgearbeitet worden sind, werden alle Knoten des Graphen verbunden, die bezüglich des Gitters einen Abstand kleiner p haben. Hierbei wird das Gewicht der Kanten $e = \{u, v\}$ mit $1/d(u, v)$, wobei $d(u, v)$ der Abstand der beiden Punkte voneinander ist, festgelegt. Algorithmus 1 zeigt die Generierung des Graphen in Pseudocode.

Algorithm 1: AttraktorGraph(c, $minDis$, $maxDis$, g)

```
erzeuge c Knoten im Gitter g mit minimalen Abstand von minDis
foreach Knoten u do
    cluster(u) ← index(u)
foreach Gitterkoordinate p do
    u ← p.nähesterClusterKnoten()
    d ← δ(p, u) {Distanz von p und u}
    erzeuge mit einer Wahrscheinlichkeit 1/d einen Knoten v an Koordinate p
    if v ist an p erzeugt worden then
        cluster(v) ← cluster(u)
        erzeuge Kante e zwischen v und u
        gewicht(e) ← 1
foreach Nodes u,v do
    if δ(u, v) < maxDis then
        erzeuge Kante e zwischen u und v
        gewicht(e) ← 1/δ(u, v)
```

Man erhält also einen Graphen mit c Clustern, die je nach dem zugrunde liegenden Gitter und den gewählten Parametern mehr oder minder dicht sind. Dadurch, dass jeder Knoten, der nicht ein Clusterknoten ist, beim Einfügen in den Graphen mit einem Clusterknoten verbunden wird, sind die Cluster in jedem Falle zusammenhängend. Die Güte der intuitiven Clusterung hängt stark mit den gewählten Parametern zusammen, doch bei geschickter Wahl der Parameter erhielten wir - zumindestens für zweidimensionale Gitter - Graphen, dessen Clusterung intuitiv sehr gut erschien.

Beispiel Die Abbildung 4.1 zeigt die Generierung eines Attraktorgraphen mit einem zweidimensionalen Gitter der Größe 5×5, 3 Clusterknoten, einem

Mindestabstand der Clusterknoten von 3 und einem Abstand von 2, mit dem nach Einfügen aller Knoten, die Knoten verbunden werden. Die roten Knoten stellen die Clusterknoten dar.

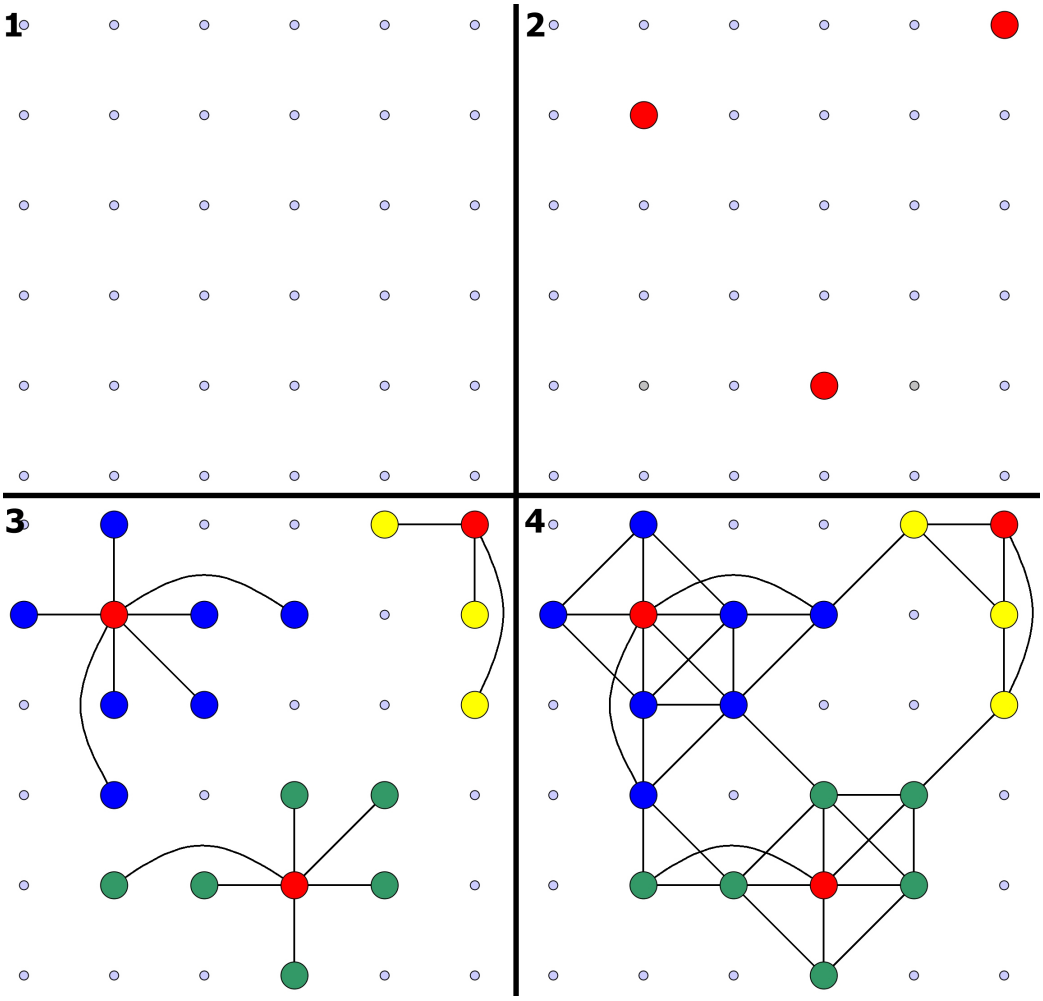

Abbildung 4.1: Zunächst erzeugen wir eine Gitterstruktur der Größe 6×6 (1). Daraufhin werden die Clusterknoten im Gitter verteilt (2). Nun werden die zusätzlichen Knoten eingefügt und mit den Clusterknoten verbunden, zu dem sie den geringsten Abstand besitzen (3). Als letztes werden nun noch Kanten zwischen den Knoten eingefügt, die einen Abstand kleiner 2 haben (4).

Wir beschränkten uns bei unseren Untersuchungen auf zwei- und dreidimensionale Gitterstrukturen. Außerdem untersuchten wir Attraktoren, denen ein Hyperwürfel als Gitterstruktur zugrunde liegt.

4.1.1 2D Clusterknoten

Bei einem 2D Clusterknoten Graphen handelt sich um einen Attraktor, dem ein zweidimensionales Gitter zugrunde liegt. Hierdurch können Knoten eines Clusters, sofern $maxDis$ kleiner $minDis$ ist, mit maximal acht paarweise verschiedenen Clustern durch eine Kante verbunden sein. Tests mit Instanzen, die eine signifikante Clusterung aufwiesen, zeigten aber, dass ein Cluster meist maximal mit fünf anderen Clustern direkt verbunden war.

4.1.2 3D Clusterknoten

Analog zum 2D Clusterknoten Graphen liegt dem 3D Clusterknoten Graphen ein dreidimensionales Gitter zugrunde. Aus diesem Grunde können mehr Intercluster Kanten entstehen und ein Cluster kann, sofern $maxDis$ kleiner $minDis$ gilt, mit maximal 26 anderen Clustern verbunden sein. Aber auch hier zeigten Tests, dass bei den meisten Graphen mit signifikanter Clusterung, die so generiert wurden, dieser Wert bei maximal 10 lag.

4.1.3 Hyperwürfel

Mit der allgemeinen Form der Gitterstruktur lässt sich auch ein Hyperwürfel als zugrunde liegende Struktur erzeugen. Hierfür setzt man die Größe jeder Dimension auf zwei fest und variiert die Größe der Gitterstruktur ausschließlich durch die Veränderung der Anzahl k der Dimensionen. Dadurch erhält man ein Gitter mit 2^k möglichen Positionen für Knoten. Auf Grund der Gegebenheiten des Hyperwürfels entstehen bei diesem Graphtyp sehr dichte Cluster, da ein Knoten zu bis zu $2k$ anderen Knoten einen Abstand von eins haben kann. Dies hat aber auch zur Folge, dass, je nach Wahl der Parameter, entweder sehr viele oder so gut wie gar keine Intercluster Kanten entstehen. Man erhält also sehr dichte Cluster, die in den meisten Fällen entweder gar nicht oder sehr stark mit anderen Clustern verbunden sind.

4.2 Zufallscluster Graphen

Wir stellen unsere Variante von Zufallscluster Graphen vor, die ähnlich in [7] und [8] beschrieben sind. Bei diesen Graphen generiert man zunächst eine Partition P und verbindet dann zwei Knoten, die dem gleichen Cluster zugehören, mit einer Wahrscheinlichkeit p. Zwei Knoten unterschiedlicher Cluster

werden mit einer Wahrscheinlichkeit q verbunden. Das Kantengewicht einer Intracluster Kante wird zufällig kleiner p gesetzt, das einer Intercluster Kante zufällig kleiner q. Durch diese Wahl der Kantengewichte erwartet man, dass die induzierte intuitive Clusterung im gewichteten Falle besser als im ungewichteten ist.

Algorithm 2: ZufallsClusterGraph($P[]$, p, q)

for $i = 0$; $i < P.length$; i++ **do**
  erzeuge $P[i]$ Knoten
  setze Clusterindex dieser Knoten auf i
Verbinde zwei Knoten gleicher Cluster mit Wahrscheinlichkeit p
Setze Kantengewicht zufällig $< p$
Verbinde zwei Knoten verschiedener Cluster mit Wahrscheinlichkeit q
Setze Kantengewicht zufällig $< q$

Durch die Art der Verteilung der Größen für die Partition kann man mit diesem Generator verschiedenartige Graphen erstellen. Wir wählten zwei Arten der Verteilung. Zum einen eine Normalverteilung, zum anderen eine Gleichverteilung.

4.2.1 Normal verteilte Zufallscluster Graphen

Die zugrunde liegende Partition ist in diesem Falle normal verteilt, wobei man sowohl den Mittelwert als auch die Standardabweichung für die Größe der Cluster festlegen kann. Man möchte also möglichst viele Cluster von der Größe des Mittelwertes haben. Bei einer kleinen Standardabweichung kriegt man fast ausschließlich Cluster von der Größe des Mittelwertes.

Algorithmus 3 zeigt einen Generator für eine normal verteilte Partition mit n Einträgen, dem Mittelwert M und der Varianz V.

Algorithm 3: ErzeugeNormalPartition(n, M, V)

$P \leftarrow$ new Array[n]
for $i = 0$; $i < P.length$; i++ **do**
  $d \leftarrow$ normal verteilter Double mit Mittelwert 0 und Varianz 1
  $P[i] = (d \cdot V) + M$

Die so erzeugte Partition kann dem Algoritmus 2 übergeben werden, sodass der so erhaltene Zufallscluster Graph eine Normalverteilung der Clustergrößen aufweißt.

4.2.2 Gleich verteilte Zufallscluster Graphen

Hier sind die Größen der Cluster gleich verteilt, wobei man einen Maximalwert und einen Minimalwert für die Größen der Cluster festlegen kann. Analog zu 4.2.1 erstellt Algorithmus 4 eine Partition, deren Einträge einer Gleichverteilung entsprechen.

Algorithm 4: ErzeugeGleichPartition(n, min, max)

```
P ← new Array[n]
for i = 0; i < P.length; i++ do
    I ← gleich verteilter Interger ∈ [min, max]
    P[i] = I
```

Diese Partition kann ebenfalls Algorithmus 2 als Partition übergeben werden, sodass wir Graphen erhalten, dessen Clustergrößen einer Gleichverteilung entsprechen.

4.3 s-let Graphen

Der s-let Graph ist aus dem Beweis für die NP-schwere des Cluster Editing Problems [5] motiviert. Wir erklären zunächst diesen Graphen und werden danach unsere Erweiterungen vorstellen.

1. Zunächst erzeugen wir n Cliquen, die jeweils aus drei Knoten bestehen und paarweise knotendisjunkt sind. Wir bezeichnen diese Knoten als u Knoten. Für jede Clique i wird eine weitere Clique mit m Knoten (im Folgenden als S-Knoten bezeichnet) erzeugt. Diese beiden Cliquen werden nun vollständig miteinander verbunden. Unser Graph besteht nun aus n Cliquen von je $m + 3$ Knoten, die paarweise knotendisjunkt sind. Jede Zusammenhangskomponente wird einem eigenen Clusterindex zugeordnet.

2. Nun werden zufällig drei der u Knoten ausgewählt, wobei diese drei Knoten im Graphen noch nicht cliquenweise verbunden sein dürfen. Es wird eine Clique der Größe m erzeugt und mit den drei zufälligen Knoten vollständig verbunden. Die S-Knoten dieser Clique erhalten einen eigenen Clusterindex.

3. Schritt 2 darf so lange wiederholt werden, bis keiner der drei ausgewählten Knoten mehr als drei Verbindungen zu verschiedenen S-Knoten mit paarweise verschiedenem Clusterindex besitzt.

Abbildung 4.2 zeigt die Generierung solch eines Graphen mit $n = 2$ und $m = 10$. Schritt 2 wurde nur einmal durchgeführt.

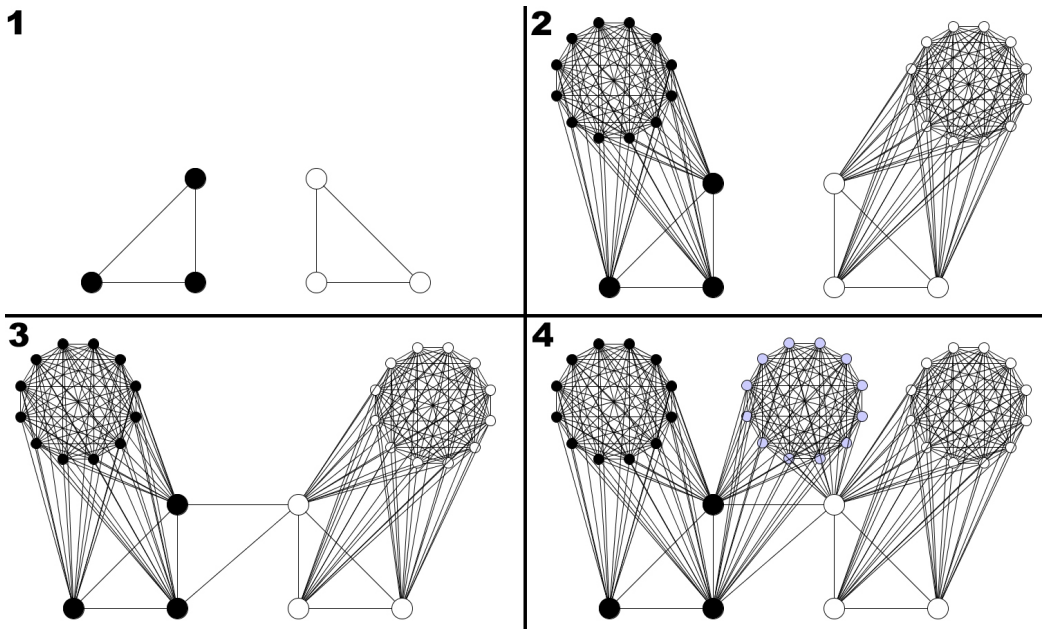

Abbildung 4.2: Zunächst erzeugen wir zwei Cliquen mit je drei Knoten (1). Diese sechs Knoten sind die u-Knoten des Graphen. Für jede dieser Cliquen erzeugen wir nun eine Clique mit je 10 Knoten, die so genannten S-Knoten, und verbinden diese mit ihren zugehörigen u-Knoten (2). Daraufhin werden drei der u-Knoten ausgewählt und cliquenartig verbunden (3). Als letztes werden für diese neu entstandene Clique 10 neue S-Knoten erzeugt und cliquenartig verbunden (4).

In dem NP-Vollständigkeitsbeweis [5] entsprechen je drei u-Knoten einem Triplet des 3-Exact-3-Cover Problems (3X3C) [9, SP2]. Die S-Knoten werden eingefügt, damit der Graph hinreichend groß ist, um Cluster Editing auf 3X3C zu reduzieren.

Erweiterter Generator Wir erweiterten dieses Prinzip um folgende Eigenschaften:

- Die Größe s der initialen Cliquen ist frei wählbar. Es werden dann in Schritt 2 auch nicht drei sondern s Knoten ausgewählt.
- Schritt 3 wird mindestens $n/2$ und höchstens n mal ausgeführt. Ausserdem dürfen die s ausgewählten u-Knoten Kanten zu s verschiedenen

S-Knoten besitzen.

- Man kann die Dichte der S-Knoten festlegen. In jedem Schritt werden die S-Knoten nicht cliquenartig verbunden, sondern jede Kante wird mit einer Wahrscheinlichkeit p eingefügt.
- Ebenso kann eine Wahrscheinlichkeit festgelegt werden, die angibt, mit welcher Wahrscheinlichkeit jede Kante zwischen u- und S-Knoten eingefügt wird.

Diese Erweiterungen beheben das Problem, dass der Graph nur aus Cliquen besteht. Außerdem kann man mit diesem Generator besser die Anzahl der Cluster und Kanten zu einer vorgegebenen Zahl Knoten beeinflussen. Algorithmus 5 zeigt den nun erhaltenenen Generator in Pseudocode.

Algorithm 5: SletGraph(s, n, m, p, q)

```
c = 0 {Der Clusterindexzähler}
for i = 0; i < n; i++ do
    erzeuge Clique mit s Knoten {die u-Knoten}
    erzeuge m Knoten {die S-Knoten}
    setze Clusterindex dieser Knoten auf c
    c++
    foreach s, m do
        erzeuge Kante (s, m) mit Wahrscheinlichkeit q
    foreach Knotenpaar m_1, m_2 der S-Knoten do
        erzeuge Kante {m_1, m_2} mit Wahrscheinlichkeit p
for i = 0; i < s − 1; i++ do
    U = ∅
    U ← drei zufällige u-Knoten u_1, u_2, u_3
    if Clusterindex der Knoten von U verschieden then
        verbinde U Knoten cliquenartig
        erzeuge m Knoten {die S-Knoten}
        setze Clusterindex dieser Knoten auf c
        c++
        foreach u ∈ U, m do
            erzeuge Kante (u, m) mit Wahrscheinlichkeit q
        foreach Knotenpaar m_1, m_2 der S-Knoten do
            erzeuge Kante {m_1, m_2} mit Wahrscheinlichkeit p
```

4.4 Rekursive Graphen

Bei den rekursiven Graphen ist die Idee, rekursiv Subgraphen miteinander zu verbinden. Hierbei sind sehr viele Varianten denkbar. Wir beschränkten uns, um den Rahmen der Tests nicht zu sprengen, auf zwei prinzipielle Vorgehensweisen. In beiden Fällen erzeugen wir mehrere Cliquen, die zunächst keine Verbindungen untereinander haben. Jede dieser Cliquen fasst man im nächsten Schritt als einzelnen Knoten auf. Nun verbindet man diese Knoten miteinander, wobei wir zwei Arten der Verbindungen wählten:

1. cliquenartiges Verbinden der Knoten
2. kreisförmiges Verbinden der Knoten

Die neu entstandenen Zusammenhangskomponenten können nun je zu einem Knoten zusammengefasst werden. Diese Knoten werden dann erneut cliquen- oder kreisartig miteinander verbunden. Dies führt man solange durch, bis der Graph zusammenhängend ist. Es sei angemerkt, dass dieses Verfahren nur dann funktioniert, wenn die Anzahl der Cliquen im ersten Schritt passend gewählt wird. Algorithmus 6 stellt den Graphgenerator als Pseudocode dar.

Algorithm 6: RekursiverGraph(n, c, j, w, *type*)

```
n ← c · j^i {Knotenanzahl wird so erhöht, dass die Anzahl der Cliquen
auf unterster Ebene passend sind}
erzeuge n/c Cliquen
setze Kantengewichte aller Kanten auf 1
l ← 1 {der aktuelle Clusterlevel}
while Graph nicht zusammenhängend do
    foreach Zusammenhangskomponente k do
        Wähle zufälligen Knoten n_k aus k
    for i = 0; i < n/(c · j^l); i++ do
        if type == true then
            verbinde n_i, n_{i+1}, ..., n_{j-1} cliquenartig
            setze Kantengewicht auf l · w
        else
            verbinde n_i, n_{i+1}, ..., n_{j-1} kreisförmig
            setze Kantengewicht auf l · w
    l++
```

Im Gegensatz zu den anderen Graphgeneratoren erhält man hier nicht nur eine vermeintlich signifikante Clusterung pro Graph, sondern mehrere. Man kann jedes Level der Erzeugung des Graphen als Clusterung auffassen. Diese Level wollen wir als Clusterlevel bezeichnen. Außerdem kann bei diesem Graphtypen der Intracluster Conductance Index sehr gut approximiert werden, da ein Schnitt der Cluster mit geringer Leitfähigkeit bestimmt werden kann.

Beispiel Abbildung 4.4 zeigt einen rekursiven Graphen mit $c = j = 5$ und einem Clusterlevel von 2.

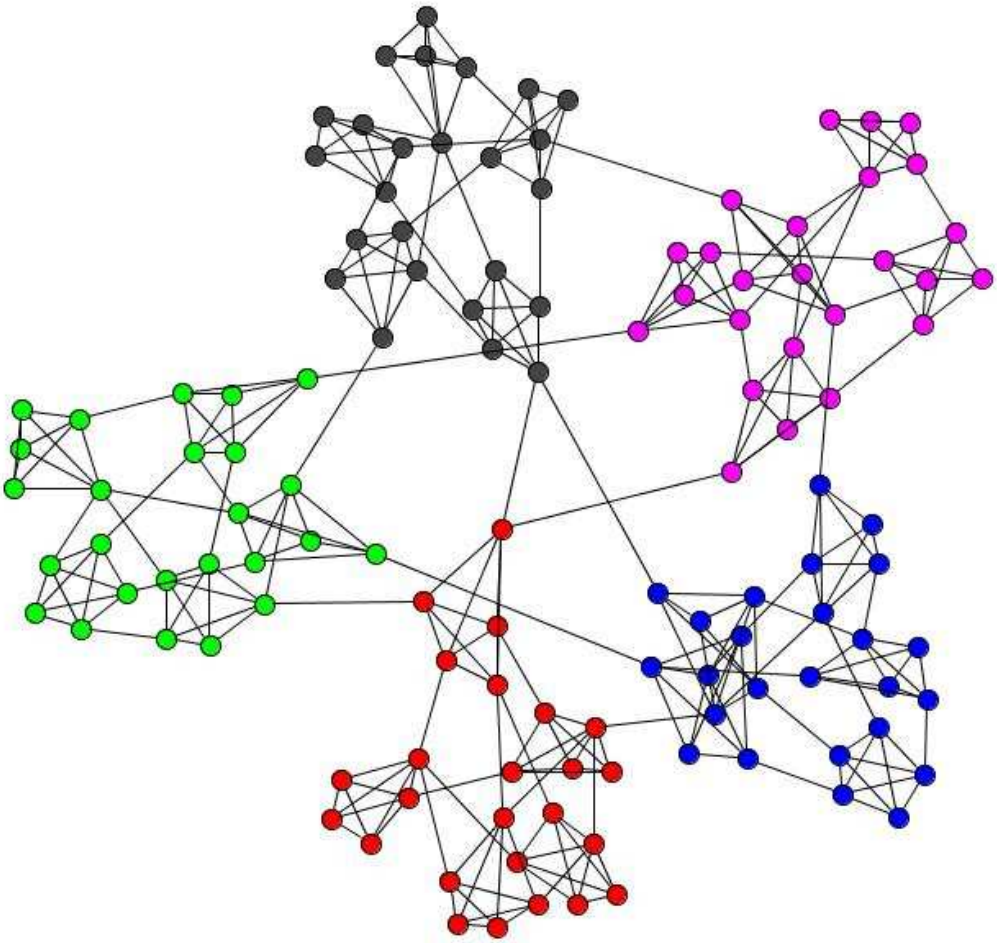

Abbildung 4.3: Rekursiver Graph mit 125 Knoten, 5 Clustern und Clusterlevel 2

Berechnung der Intracluster Conductance Bei den Varianten der von uns betrachteten rekursiven Graphen ist die Berechnung der Intracluster Conductance ohne Bestimmung der Eigenvektoren möglich. Allerdings kann im Allgemeinen nicht zugesichert werden, dass es sich hierbei um den exakten Wert handelt.

Jeder Cluster C_i besteht aus k gleichartigen Komponenten K_j. Der Schnitt $\mathcal{C} = (C, V \setminus C)$ mit $C = \{v \mid v \in K_j, j = 1, \ldots \lfloor (k/2) \rfloor\}$ ist ein Schnitt des Graphen $G[C_i]$ mit geringer Leitfähigkeit. Mit Hilfe dieses Schnittes kann man nun den Intracluster Conductance Wert für diesen Cluster C_i bestimmen. Da alle Cluster ähnliche Struktur besitzen, ist dieser Wert auch der Intracluster Conductance Wert für die gesamte Clusterung. Abbildung 4.4 zeigt hierfür ein Beispiel.

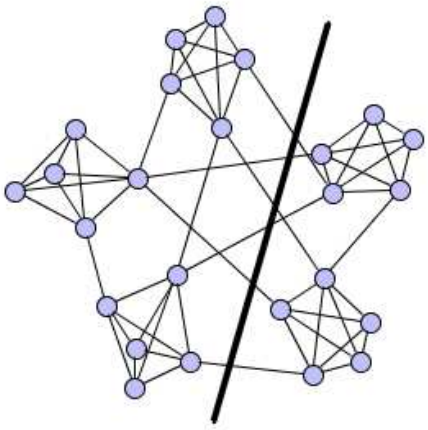

Abbildung 4.4: Ein Schnitt mit geringer Leitfähigkeit für diesen Graphen

Man kann sich gewichtete Graphen erstellen, bei denen dieser Schnitt nicht der Schnitt mit minimaler Leitfähigkeit ist. Bei den Graphen, die wir untersuchten, zeigte sich aber, dass der so gefundene Schnitt sich nur in seltenen Fällen von dem Schnitt unterschied, den der Approximationsalgorithmus aus Kapitel 3.4 berechnete. Aus diesen Gründen nutzen wir schließlich nur den hier vorgestellten Schnitt für die Berechnung der Intracluster Conductance.

Kapitel 5

Experimente

Wir untersuchten mehrere Instanzen der Generatoren aus Kapitel 4. Für jede Instanz wurden, sofern es sich um gewichtete Graphen handelte, die gewichtete Version der Indizes berechnet. Zunächst untersuchten wir die signifikanten Clusterungen, allerdings verzichteten wir auf die rekursiven Graphen aus Kapitel 4.4, da wir diese gesondert in Kapitel 5.2 betrachten wollen. Für die anderen generierten Graphen wählten wir folgende Vorgaben:

- Graphen mit 500 Knoten, ca. 3000 Kanten und 8-12 Clustern
- Graphen mit 1000 Knoten, ca. 7500 Kanten und 13-18 Clustern
- Graphen mit 5000 Knoten, ca. 100000 Kanten und 25-35 Clustern

Ferner sollten alle Cluster ungefähr die gleiche Größe haben. Diese Einschränkungen nahmen wir vor, um die signifikanten Clusterungen der Graphen der verschiedenen Generatoren miteinander vergleichen zu können.

Zusätzlich untersuchten wir gesondert die rekursiven Graphen. Die getrennte Analyse war auf Grund der Tatsache notwendig, dass wir bei den rekursiven Graphen mehrere Clusterungen untersuchen wollten. Wir wollten feststellen, welcher Index welchem Clusterlevel den besten Wert zuordnet. Außerdem wollten wir analysieren, ob ein Index dem intuitiv besten Clusterlevel den maximalen Wert zuordnet. Bei diesen Graphen untersuchten wir Graphinstanzen mit ungefähr 100.000 Knoten. Für die Kanten- und Clusteranzahl wählten wir keine Vorgabe, da diese durch die Wahl der Parameter sehr stark schwankten. Außerdem sollte bei diesen Experimenten der Schwerpunkt auf der Analyse der Clusterlevel liegen. Wir untersuchten bei den rekursiven Graphen sowohl solche, die cliquenartige Verklebung aufweisen, als auch solche, die kreisförmig verklebt werden.

Im Rahmen dieser Studienarbeit implementierten wir die Generatoren und Indizes in `Java 1.4.2` mit den Libraries `yfiles 2.2` und `colt 1.03`. Eine geeignete Cluster Datenstruktur war schon vorhanden, sodass wir diese nicht neu implementieren mussten. Für unsere Experimente stand uns ein Pentium 4 2,8 GHz mit 2 GB RAM und Windows XP zur Verfügung.

Alle durchgeführten Experimente dienen die Wechselwirkung von künstlich erzeugten prägeclustereten Graphen und Indizes zu untersuchen. Da auch größere Graphen mit ungefähr 100.000 Knoten betrachten wurden, wurde auf strenge statistische Signifikanz verzichtet.

5.1 Graphen mit signifikanter Clusterung

5.1.1 2D Clusterknoten

Bei dem 2D Clusterknoten Attraktor wählten wir mehrere Parametertupel, die jeweils Graphen generierten, die den Vorgaben entsprachen. Die Parameter in der Tabelle entsprechen den Variablen des Algorithmus' 1 in Kapitel 4.1.

Knoten	c	g	$minDis$	$maxDis$
≈ 500	8	60×60	17	4
≈ 500	9	50×50	8	4
≈ 500	10	50×50	11	4
≈ 1000	16	80×80	17	4
≈ 1000	15	50×50	10	4
≈ 5000	25	290×290	13	30
≈ 5000	30	300×300	12	30

Durch die Wahl dieser Parameter erhielten wir Graphen, die man intuitiv als gut clusterbar bezeichnen würde. Abbildung 5.1 zeigt die Werte der einzelnen Indizes für jede Graphinstanz, getrennt nach 500, 1000 und 5000 Knoten. Auf der x-Koordinate sind jeweils die Knoten eingetragen, die y-Koordinate gibt den jeweiligen Indexwert für die intuitive Clusterung an. Dabei entspricht blau dem Coverage, rot dem Perfomance, gelb dem Intercluster Conductance und grün dem Intracluster Conductance Index.

Wir stellen die Ergebnisse für 500, 1000 und 5000 Knoten zusammen vor, da ausgiebige Tests ergaben, dass die Bewertung der intuitiven Clusterung durch die Indizes für unsere jeweiligen Vorgaben unabhängig von der Graphgröße sind.

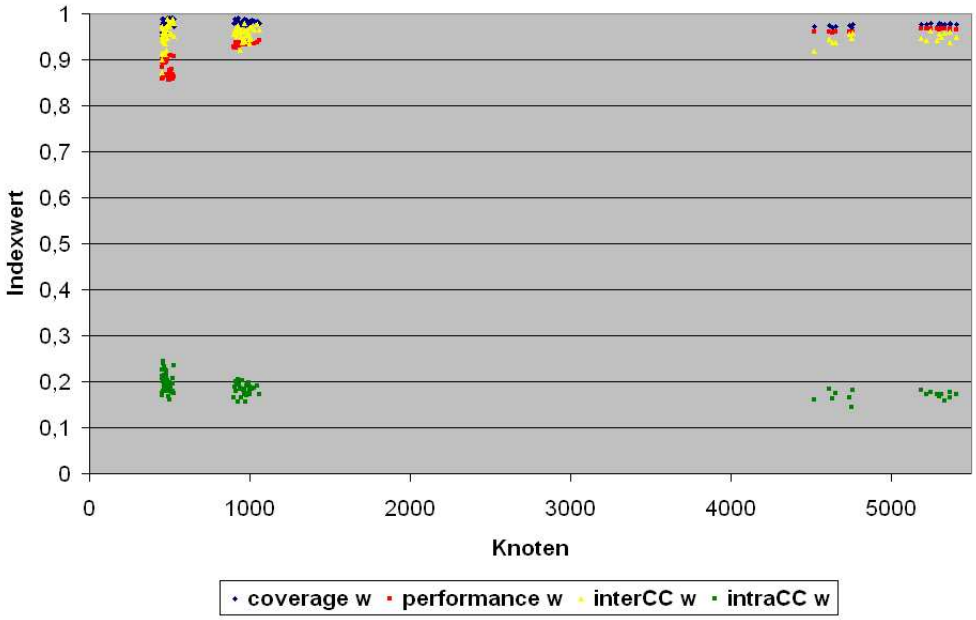

Abbildung 5.1: 2D Clusterknoten Graphen mit ungefähr 500, 1000 und 5000 Knoten

Auf diesen Graphen bewerteten die Indizes die intuitive Clusterung mit guten Werten, wobei Coverage und Intercluster Conductance sehr nahe bei 1 liegen. Außerdem ist der Intracluster Conductance Wert mit 0,2 niedrig, sodass zumindestens ein Cluster einen Flaschenhals aufweisen muss.

5.1.2 3D Clusterknoten

Auf Grund der dreidimensionalen Gitterstruktur entstehen bei den 3D Clusterknoten Graphen prozentual mehr Intercluster Kanten als bei den 2D Clusterknoten Graphen. Um zu untersuchen, in welchen Indizes sich dieser Unterschied bemerkbar macht untersuchten wir verschiedene Parametertupel, sodass Graphen entstanden, die den Vorgaben entsprachen. Auch hier entsprechen die Parameter den Variablen des Algorithmus' 1.

Knoten	c	g	$minDis$	$maxDis$
≈ 500	9	$11 \times 11 \times 11$	5	2
≈ 500	9	$11 \times 11 \times 11$	2	1
≈ 1000	15	$15 \times 15 \times 15$	5	2
≈ 1000	15	$16 \times 16 \times 16$	4	2
≈ 5000	30	$30 \times 30 \times 30$	3.2	8.5
≈ 5000	25	$29 \times 29 \times 29$	3.5	8.5

Auch die durch diese Parameter erhaltenen Graphen schienen intuitiv gut clusterbar, wobei im Gegensatz zu den 2D Clusterknoten Graphen die Anzahl der Intercluster höher war. Abbildung 5.2 zeigt die Bewertungen der Indizes für diese Graphen.

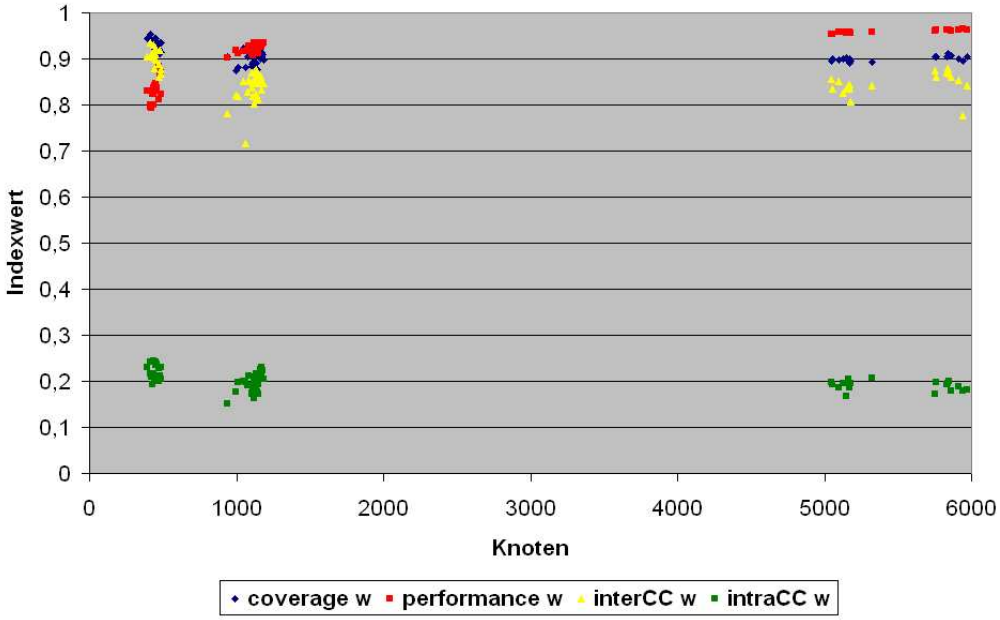

Abbildung 5.2: 3D Clusterknoten Graphen mit ungefähr 500, 1000 und 5000 Knoten

Hier fällt auf, dass im Vergleich zu den 3D Clusterknoten und 2D Clusterknoten Graphen der Performance und Intracluster Conductance Index ähnliche Werte besitzt, die Werte für Intercluster Conductance und Coverage aber niedriger sind, was darauf schließen lässt, dass diese beiden Indizes sensibler auf die Anzahl der Intercluster Kanten reagieren.

5.1.3 Hyperwürfel

Durch die Mehrdimensionalität der zugrundeliegenden Gitterstruktur, entstehen noch mehr Interclusterkanten. Wir untersuchten Hyperwürfel, die mit folgenden Parametertupeln generiert wurden.

Knoten	c	g	$minDis$	$maxDis$
≈ 600	9	2^{10}	4	1
≈ 1300	13	2^{11}	4	1
≈ 4100	6	2^{14}	6	2

Durch die starke Verbindung der Cluster untereinander war es uns nicht möglich, Graphen zu erzeugen, die den Vorgaben entsprachen. Um zumindestens die Kantenanzahl beizubehalten, variierten wir die Knoten- und Clusteranzahl.

Diese Parameter erzeugten Graphen, die man intuitiv als nicht besonders gut clusterbar bezeichnen würde, da die Verbindungen der Cluster untereinander stark waren. Abbildung 5.3 zeigt die Ergebnisse für diese Graphen.

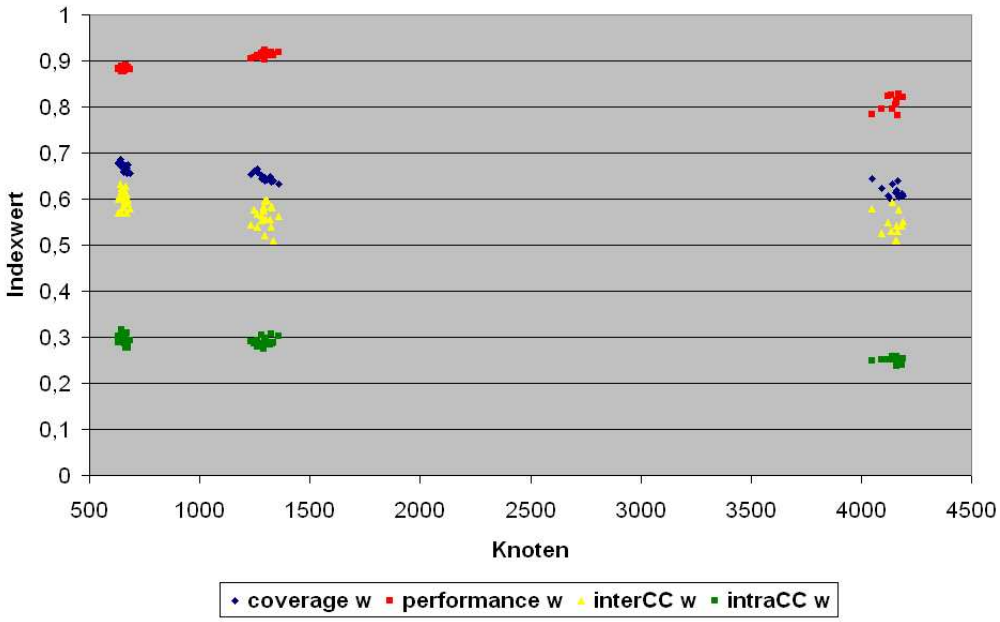

Abbildung 5.3: Hyperwürfel Graphen mit ca. 600, 1300 und 5000 Knoten

Bei den Hyperwürfeln wird die vorgegebene Clusterung von Intracluster Conductance und Performance gut bewertet, wohingegen Coverage und Interclu-

ster Conductance die Clusterung recht schlecht bewerten. Vor allem ist auffällig, dass die Werte für Intracluster Conductance und Performance ungefähr denen von 2D und 3D Clusterknoten Graphen entsprechen, wohingegen die anderen beiden Indizes signifikant niedriger als bei den 2D und 3D Graphen sind.

5.1.4 Zufallscluster Graphen

Durch die Vorgabe, dass die Cluster ungefähr gleich groß sein sollten, waren die normal verteilten Zufallscluster Graphen sehr ähnlich zu den gleich verteilten. Aus diesem Grund stellen wir stellvertretend für die Zufallsclustergraphen nur die Ergebnisse der normal verteilten Graphen vor.

Wir wählten folgende Parameter, wobei M der Mittelwert, V die Varianz, p die Wahrscheinlichkeit für Intracluster Kanten und q für Intercluster Kanten repräsentieren:

Knoten	Cluster	M	V	p	q
≈ 500	8	64	10	0.2	0.0005
≈ 500	8	64	10	0.15	0.002
≈ 500	10	50	8	0.3	0.001
≈ 500	10	50	8	0.2	0.002
≈ 500	12	40	5	0.3	0.002
≈ 1000	13	75	13	0.18	0.002
≈ 1000	13	75	13	0.2	0.0003
≈ 1000	13	75	13	0.22	0.0001
≈ 1000	15	65	9	0.2	0.0003
≈ 1000	15	65	9	0.18	0.0015
≈ 1000	18	55	5	0.25	0.002
≈ 1000	18	55	5	0.3	0.0003
≈ 5000	25	200	25	0.2	0.0001
≈ 5000	25	200	25	0.19	0.0005
≈ 5000	30	165	20	0.25	0.0002
≈ 5000	35	140	15	0.28	0.0003

Da wir die Wahrscheinlichkeit für Intercluster Kanten (q) sehr gering wählten, erhielten wir Graphen, die sehr gut clusterbar erschienen. Allerdings waren die Cluster selbst nicht sehr stark verbunden, da wir p ebenfalls verhältnismäßig niedrig wählten. Abbildung 5.4. zeigt die Ergebnisse für diese Graphen, wobei hier noch eine zweite Grafik vorhanden ist, die eine feinere Auflösung für Coverage und Intercluster Conductance zeigt.

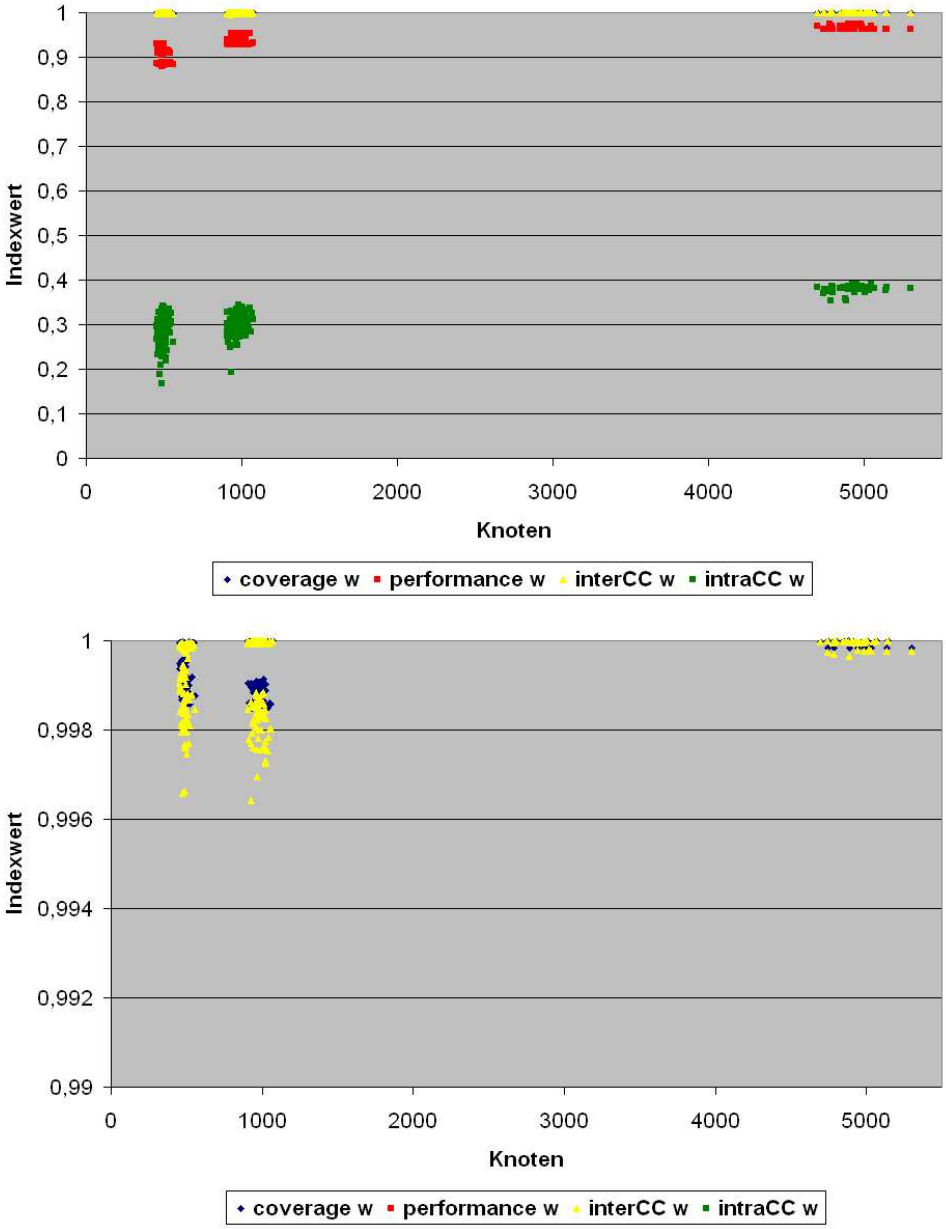

Abbildung 5.4: Zufallscluster Graphen mit gröberer und feinerer Auflösung

Wie man sieht, bewerteten Intercluster Conductance und Coverage die signifikante Clusterung bei jeder Instanz mit Werten sehr nahe bei eins. Dies läßt sich durch die Wahl von kleinen Werten für den Parameter q erklären. Performance und Intracluster Conductance bewerten die Clusterung zwar auch gut, aber nicht so hoch wie die beiden anderen Indizes.

5.1.5 s-let Graphen

Bei den s-let Graphen wählten wir die Parameter ebenfalls so, dass wir Graphinstanzen erhielten, die den Vorgaben entsprachen. Außerdem wählten wir die Parameter so, dass wenig Intercluster Kanten entstanden. Wieder entsprechen die Parameter den Variablen des zugehörigen Generatoralgorithmus' (5).

Knoten	s	n	m	p	q
≈ 500	4	6	40	0.25	0.12
≈ 500	6	6	40	0.2	0.1
≈ 1000	8	7	55	0.15	0.08
≈ 1000	10	7	55	0.12	0.05
≈ 5000	9	5	160	0.23	0.1
≈ 5000	11	5	150	0.25	0.1

Abbildung 5.5 zeigt die Auswertungen für die s-let Graphen, die sich in einem zu den vorherigen Auswertungen unterscheidet. Bei den s-let Graphen handelt es sich um ungewichtete Graphen. Deshalb nutzten wir in diesem Falle jeweils die ungewichtete Version der Indizes.

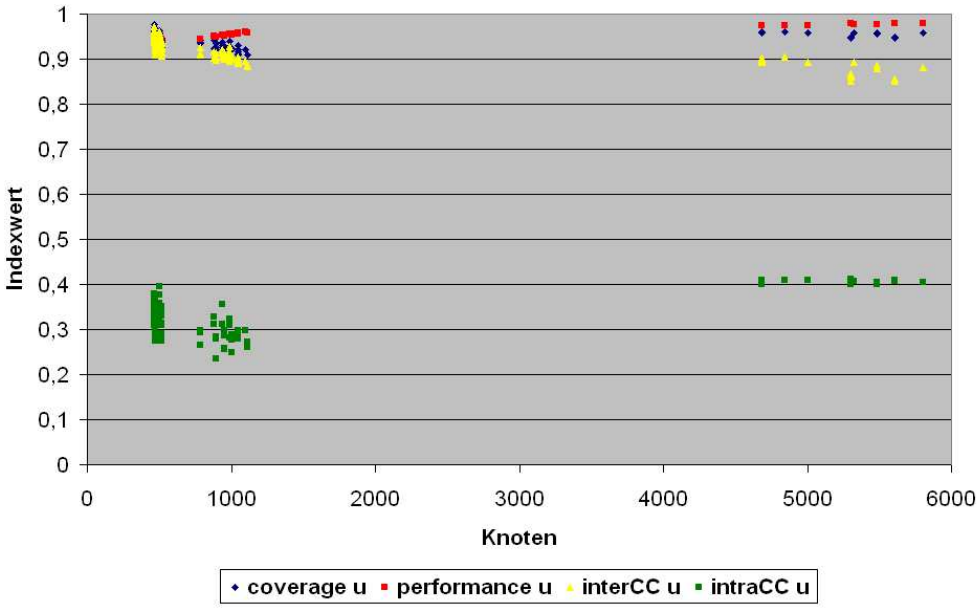

Abbildung 5.5: Auswertungen der s-let Graphen

Bei diesen gewählten Parametern bewerteten alle Indizes die vorgegebene Clusterung gut.

5.2 Die Level der rekursiven Graphen

Zuletzt untersuchten wir rekursive Graphen, wobei hier die verschiedenen Indizes nicht nur für eine Clusterung, sondern für sämtliche Clusterlevel berechnet wurden. Wir teilten dabei die rekursiven Graphen in drei Typen auf:

Typ 1: Die Größe c der Cliquen auf der untersten Ebene ist kleiner als die Anzahl j der Subgraphen, die auf jeder Ebene verbunden werden.

Typ 2: Hier sind c und j gleich groß.

Typ 3: Bei dem letzten Typ wählten wir c größer j.

Für alle drei Typen untersuchten wir Graphinstanzen von circa 100.000 Knoten. Durch die Wahl der Parameter ist sowohl die Anzahl der Cluster, als auch die Anzahl der Kanten für jede Graphinstanz unterschiedlich. Da wir in diesem Abschnitt aber die einzelnen Ebenen analysieren wollten, nahmen wir diese Einschränkung in Kauf.

Wir beschränkten uns bei diesen Experimenten auf ungewichtete Graphen, da Tests ergaben, dass sich bei den von uns betrachteten gewichteten Graphen die Ergebnisse sich nicht von den hier vorgestellten unterschieden.

5.2.1 Cliquenartige Verklebung

Zunächst untersuchten wir alle drei Typen bei cliquenartiger Verklebung der Subgraphen.

Typ 1

Für Typ 1 wählten wir folgende Parameter, wobei hier wieder die Parameter den Variablen des Algorithmus' 6 entsprechen.

	Knoten	c	j	max. Level
Instanz 1	≈ 100.000	3	14	5
Instanz 2	≈ 100.000	3	33	4
Instanz 3	≈ 100.000	6	26	4
Instanz 4	≈ 100.000	8	23	4

Bei diesen Instanzen ist intuitiv die beste Clusterung auf Clusterlevel 2, da die kleinen Cliquen unterster Ebene erneut cliquenartig verbunden sind. Bereits

auf Clusterlevel 3 ist die Dichte der Cluster aber zu gering, um von einer guten Clusterung zu sprechen.

Abbildung 5.6 zeigt die Ergebnisse für diese Graphen, wobei hier die Grafiken nach Index getrennt sind. Auf der x-Koordinate sind die Clusterlevel, auf der y-Achse die Werte für den jeweiligen Index, abgetragen. Jede Linie steht für die verschiedenen Clusterlevel einer Graphinstanz.

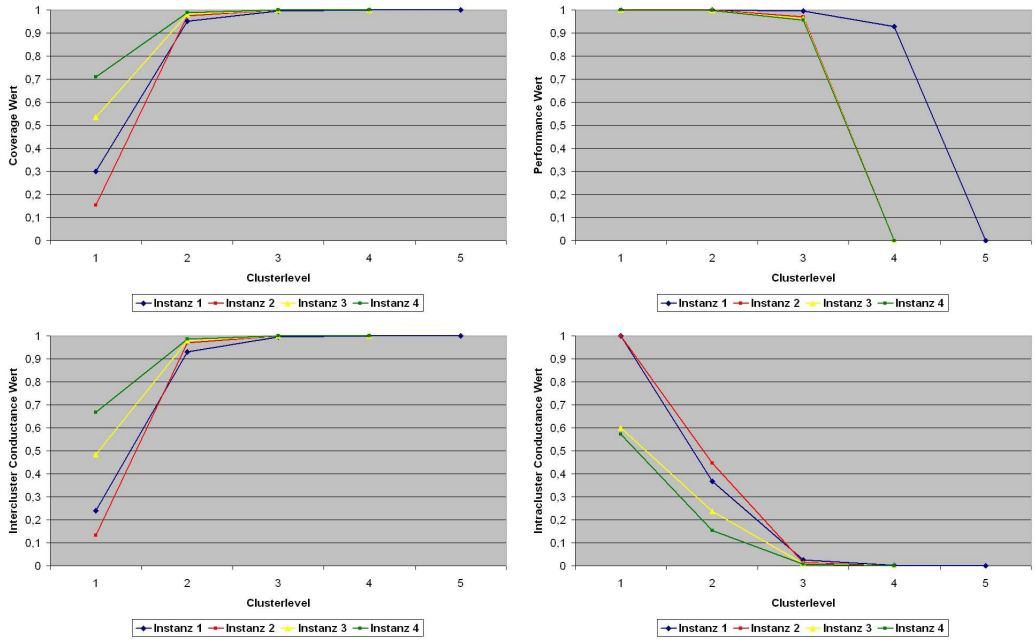

Abbildung 5.6: Coverage, Peformance, Intra- und Intercluster Conductance Werte der Clusterlevel für rekursive Graphen des Typ 1 mit cliquenartiger Verklebung

Es ist zu erkennen, dass Coverage und Intercluster Conductance monoton steigend sind. Dieser Effekt tritt ein, da die Anzahl der Intercluster Kanten mit jeder Erhöhung des Levels abnimmt. Performance und Intracluster Conductance hingegen sind monoton fallend, da die Dichte der Cluster mit steigendem Clusterlevel abnimmt.

Außerdem fällt auf, dass alle Indizes, bis auf Intracluster Conductance, fast alle Clusterlevel sehr gut bewerten. Darüber hinaus besitzt kein Index sein Maximum auf dem intuitiv besten Clusterlevel.

Typ 2

Bei Typ 2 wählten wir folgende Parameter:

	Knoten	c	j	max. Level
Instanz 1	≈ 100.000	3	3	11
Instanz 2	≈ 100.000	7	7	6
Instanz 3	≈ 100.000	10	10	5
Instanz 4	≈ 100.000	18	18	4

Intuitiv ist es bei diesen Graphen schwer zu entscheiden, welcher Clusterlevel die beste Clusterung ist. Sowohl Level 1 als auch Level 2 sind als bestes Clusterlevel möglich. Wobei mit steigendem c Clusterlevel 1 immer besser erscheint. Bei Instanz 1 handelt es sich um den Hanoi-Graphen, den man intuitv nicht als gut clusterbar bezeichnen würde.

Die Ergebnisse für diese Graphen zeigt Abbildung 5.7.

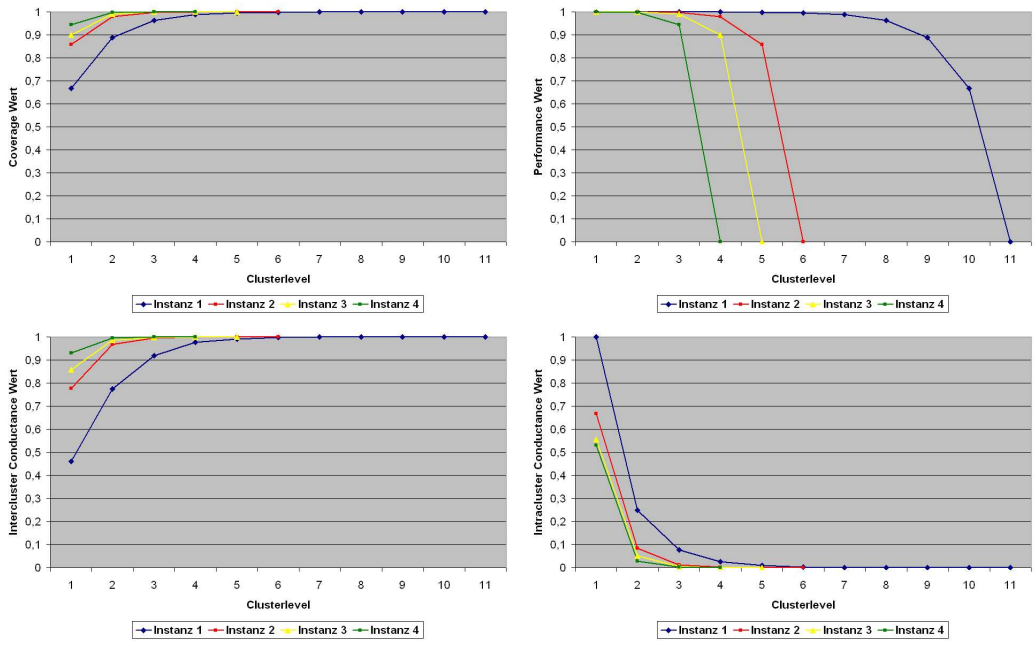

Abbildung 5.7: Typ 2 mit cliquenartiger Verklebung

Hier fällt zum einen auf, dass für Instanz 1 Intercluster Conductance erst ab Clusterlevel 3 gute Werte erreicht, Intracluster Conductance aber nur die Level 1 und 2 akzeptabel bewertet. Für die anderen Instanzen gilt wieder, dass Intracluster Conductance und Performance eher niedrige, die anderen beiden Indizes eher höhere Level, preferieren.

Typ 3

Bei Typ 3 wählten wir folgende Parameter:

	Knoten	c	j	max. Level
Instanz 1	≈ 100.000	16	3	9
Instanz 2	≈ 100.000	33	5	6
Instanz 3	≈ 100.000	42	7	5
Instanz 4	≈ 100.000	48	3	8

Bei diesen Instanzen ist intuitiv die beste Clusterung das unterste Clusterlevel, da zunächst große Cliquen gebildet werden, von denen dann wenige verbunden werden. Aus diesem Grunde ist die Dichte der Cluster bereits auf Clusterlevel 2 nicht mehr hoch genug, um von einer guten Clusterung sprechen zu können.

Abbildung 5.8 zeigt die Ergebnisse für diese Graphen.

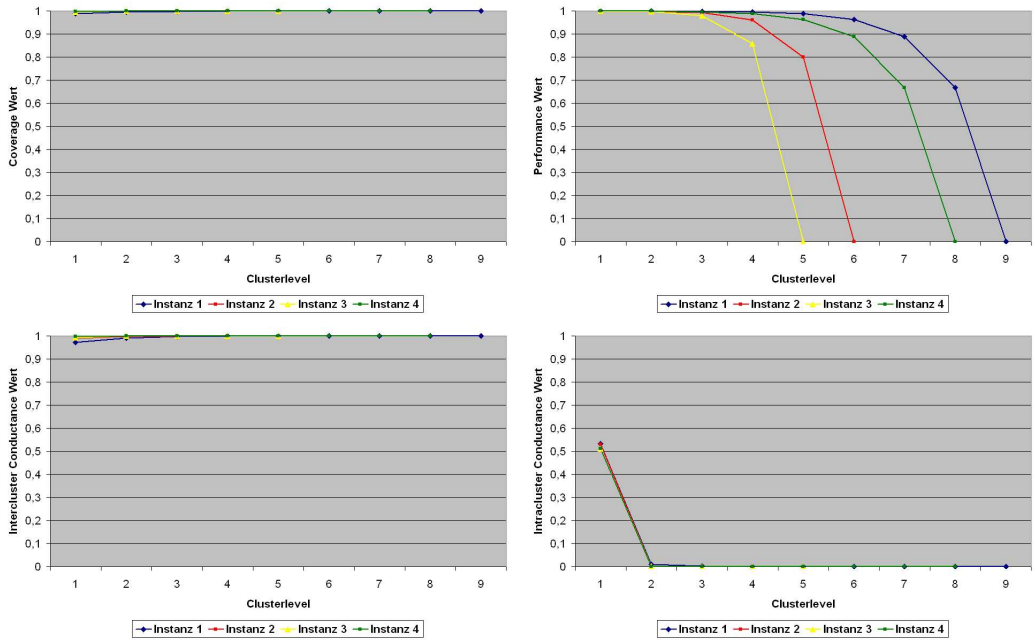

Abbildung 5.8: Typ 3 mit cliquenartiger Verklebung

Hier fällt auf, dass Intercluster Conductance und Coverage alle Clusterlevel sehr gut bewerten, was der Intuition widerspricht. Aber auch Performance bewertet mittlere Level zu hoch. Zwar besitzt Performance sein Maximum auf Level 1, jedoch ist der Unterschied zu Level 2 sehr gering. Nur Intracluster

Conductance bewertet die Clusterlevel intuitiv richtig. Hier wird Level 1 mit einem sehr guten Wert von ungefähr 0,5 bewertet, alle anderen Level werden mit Wert sehr nahe bei 0 bewertet.

5.2.2 Kreisartige Verklebung

Wir untersuchten noch rekursive Graphen mit kreisartiger Verklebung, da bei diesen Graphen die Cluster höherer Level auf Grund der kreisartigen Verklebung signifikant weniger dicht sind, als dies bei rekursiven Graphen mit cliquenartiger Verklebung der Fall ist. Als Instanzen wählten wir die gleichen Parameter wie bei der cliquenartigen Verklebung in Kapitel 5.2.1.

Typ 1

Bei diesem Typ ist das unterste Clusterlevel die intuitiv beste Clusterung für den Graphen, da die Cliquen der untersten Ebene nur kreisförmig verbunden werden, sodass die Cluster des Clusterlevel 2 bereits nicht besonders dicht sind. Allerdings sind die Cliquen bei Instanz 1 und 2 so klein, dass diese Graphen intuitiv nicht als gut clusterbar gelten.

Abbildung 5.9 zeigt die Ergebnisse für diesen Typ.

Hier fällt auf, dass Intracluster Conductance nur Level 1 gut bewertet. Bei Instanz 1 und 2 wird dieses Level aber von der Intercluster Conductance mäßig bewertet. Auch hier liefern Coverage und Performance unbefriedigende Ergebnisse, da sie fast alle Clusterlevel sehr gut bewerten.

Typ 2

Bei diesem Typ von rekursiven Graphen ist das unterste Clusterlevel für die Instanzen 2,3 und 4 die beste Clusterung, da ab Level 2 die Subgraphen nur noch kreisförmig verbunden werden. Instanz 1 entspricht Instanz 1 vom Typ 2 in Kapitel 5.2.1, da bei einer Verklebung von 3 Subgraphen pro Ebene eine cliquenartige Verbindung einer kreisartigen entspricht. Aus diesem Grunde erwartet man für diese Instanz ein bestes Clusterlevel von 2.

Abbildung 5.10 zeigt die Ergebnisse für diese Graphen.

Für Instanz 2, 3 und 4 besitzt nur Intracluster Conductance ein eindeutiges Maximum bei dem intuitiv besten Clusterlevel, wohingegen bei Instanz 1 kein Index den intuitiv besten Clusterlevel 2 am höchsten bewertet.

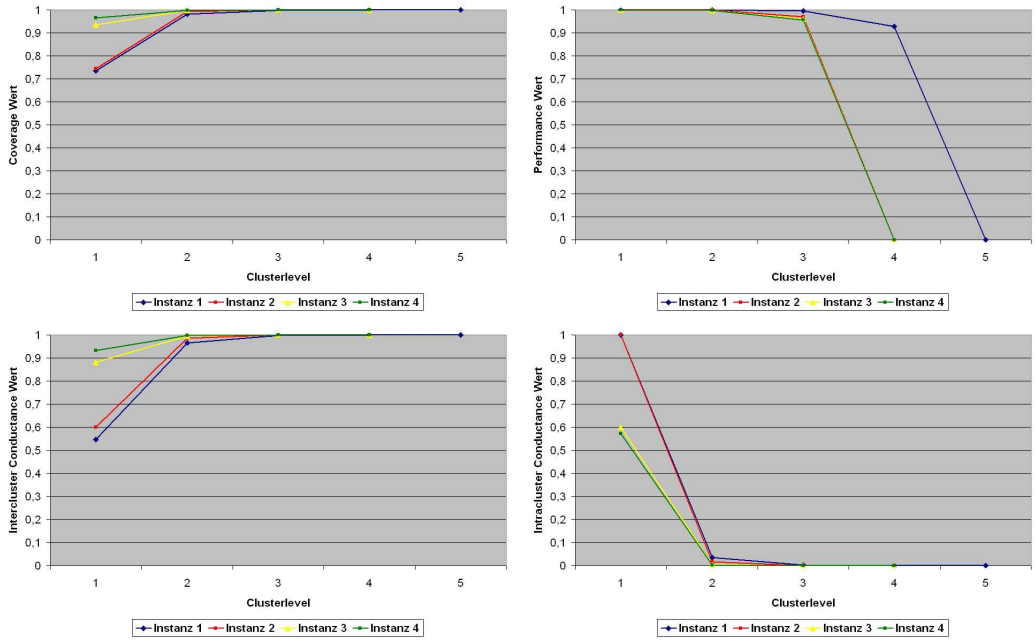

Abbildung 5.9: Typ 1 mit kreisartiger Verklebung

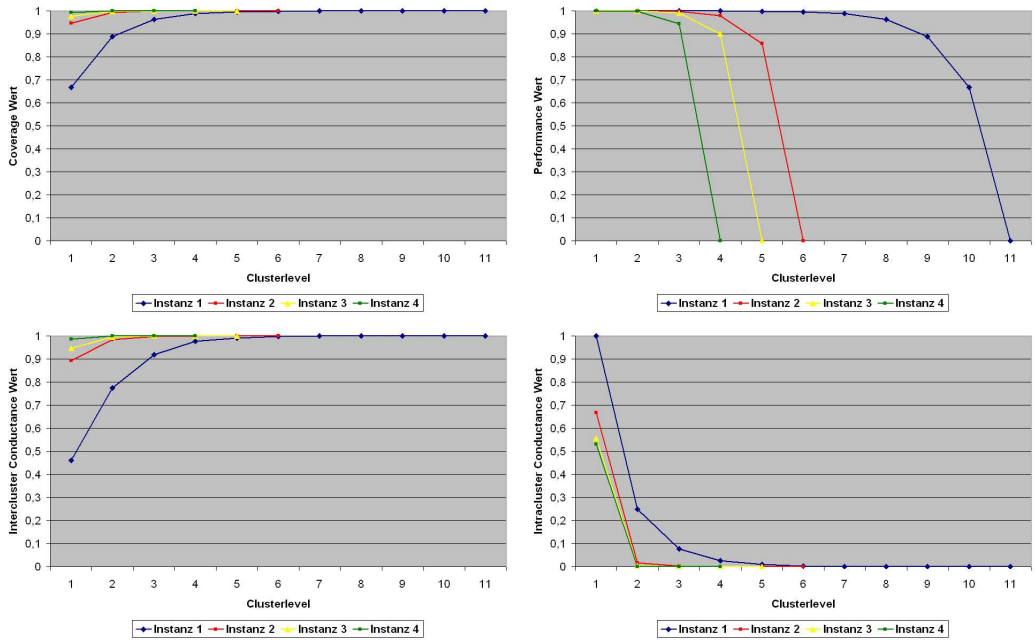

Abbildung 5.10: Typ 2 mit kreisartiger Verklebung

Typ 3

Bei Typ 3 der rekursiven Graphen mit kreisförmiger Verklebung ist Clusterlevel 1 immer die intuitiv beste Clusterung, da bei diesem Level die Cluster die Cliquen bilden und die Anzahl der Interclusterkanten auf Grund der kreisartigen Verbindung gering ist.

Abbildung 5.11 zeigt die Ergebnisse für diese Graphen.

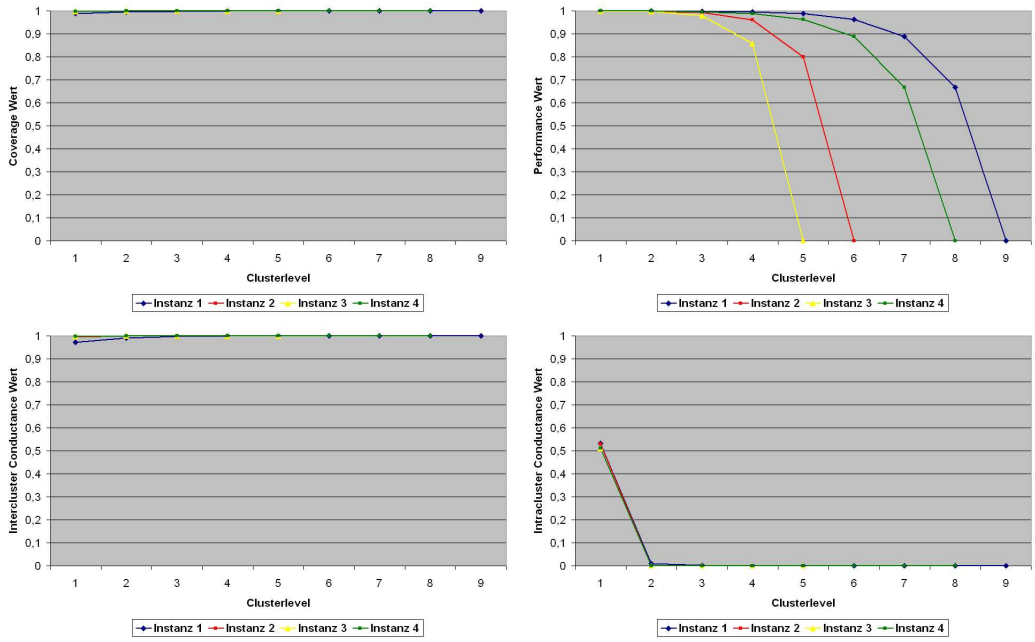

Abbildung 5.11: Typ 3 mit kreisartiger Verklebung

Coverage und Intercluster Conductance scheinen bei diesem Graphtypen unbrauchbar, da sie allen Leveln sehr hohe Werte zuordnen. Performance bewertet alle niedrigeren Level sehr gut, nur Intracluster Conductance bestätigt die Intuition.

Kapitel 6

Ergebnisse

Wir stellen in diesem Abschnitt nun die Ergebnisse vor, wobei wir einen Index vorstellen möchten, der auf allen von uns untersuchten Graphen gute Bewertungen liefert.

Prägeclusterte Graphen Bei der Analyse der Attraktorgraphen ist hervorzuheben, dass 2D und 3D Clusterknoten Graphen intuitiv gute prägeclusterte Graphen sind, wohingegen Hyperwürfel auf Grund ihrer vielen Intercluster Kanten als nicht besonders gute Clustergraphen zu bezeichnen sind. Dass dies fast ausschließlich an den Intercluster Kanten liegt, zeigt sich daran, dass Performance und Intracluster Conductance für alle drei Typen ähnliche Werte liefern, wohingegen Coverage und Intercluster Conductance schlechte Werte für Hyperwürfel und gute für 2D beziehungsweise 3D Clusterknoten Graphen liefern. Aus diesem Grund sollte man nicht ausschließlich Performance oder Intracluster Conductance als alleinigen Index für die Bewertung von Clusterung benutzen.

Man könnte annehmen, dass vielleicht Intercluster Conductance oder Coverage als alleiniger Index ausreicht. Gegen Coverage sprechen aber die gravierenden Nachteile aus Kapitel 3.1. Auch für Intercluster Conductance haben wir in Kapitel 3.3 einen Graphen vorgestellt, bei dem dieser Index nicht optimale Ergebnisse liefert.

Rekursive Graphen Vor allem bei den rekursiven Graphen wird deutlich, dass keiner der vier in Kapitel 3 vorgestellten Indizes als alleiniger Index brauchbar ist. Nur bei Typ 3, sowohl bei cliquenartiger, als auch kreisförmiger Verklebung, bewertet Intracluster Conductance den intuitiv besten Cluster-

level aller Instanzen maximal. Bei Typ 1 und 2 bewertet dieser Index aber auch nicht den intuitiv besten Clusterlevel am höchsten. Somit reicht auch bei den rekursiven Graphen ein alleiniger Index, um Clusterungen zu bewerten, nicht.

Was zudem negativ auffällt, ist die Tatsache, dass Intercluster Conductance und Coverage mit steigendem Clusterlevel monoton steigend, Intracluster Conductance und Performance monoton fallend sind. Somit bewerten Coverage und Intercluster Conductance immer den höchsten Level als die beste Clusterung, Performance und Intracluster Conductance immer den niedrigsten. Dies entspricht aber in den meisten Fällen nicht der intuitiv besten Clusterung. Auch erkennt keiner der Indizes einen schlecht clusterbaren rekursiven Graphen, da zumindestens ein Level hoch bewertet wird. Außerdem bewerten alle Indizes, bis auf Intracluster Conductance, zu viele Level mit hohen Werten.

Verwendung von mehreren Indizes Ein vielversprechender Ansatz für die Bewertung der Clusterungen scheint der Folgende zu sein. Man bildet einen gewichteten Durchschnitt von Inter- und Intracluster Conductance. Diesen Durchschnitt wollen wir mit *Conductance Index* $\gamma(\mathcal{C})$ bezeichnen. Er berechnet sich wie folgt.

$$\gamma(\mathcal{C}) = x \cdot \alpha(\mathcal{C}) + y \cdot \delta(\mathcal{C}) \text{ mit } x, y \geq 0, x + y = 1$$

Bei gewichteten Graphen sollte man jeweils die gewichtete Version des Index benutzen. Mit den Variablen x und y kann man beeinflussen, ob man Intercluster Conductance stärker als Intracluster Conductance gewichten möchte, oder umgekehrt. Ersteres führt dazu, dass die Clusterung wenig Intercluster Kanten besitzen soll, dafür die Cluster aber nicht besonders dicht sein müssen. Im Gegensatz dazu führt eine stärkere Gewichtung der Intracluster Conductance zu Clustern, die zwar stark miteinander verbunden sein müssen, aber noch recht viele Intercluster Kanten besitzen dürfen.

Es sei noch zu bedenken, dass Intracluster Conductance sehr gute Clusterungen, die nicht ausschließlich aus sehr kleinen Clustern bestehen, mit ungefähr 0,5 bewertet. Werte größer 0,6 scheinen auf genügend großen Graphen nicht sinnvoll. Von daher sollte man, damit Dichte der Cluster und Verbindungen der Cluster untereinander ungefähr gleich wichtig sind, Intra- und Intercluster Conductance im Verhältnis 2:1 bewerten. Ferner sollten Clusterungen, deren Intracluster Conductance Wert über 0,6 liegt, nur in Ausnahmefällen akzeptiert werden.

Der Conductance Index liefert auf den von uns untersuchten prägeclusterten Graphen mit einer signifikanten Clusterung sehr gute Ergebnisse. Die signifikanten Clusterungen der intuitiv gut clusterbaren Graphen werden hoch bewertet, wohingegen die signifikanten Clusterungen der Hyperwürfel niedrige Werte erhalten.

Auch auf den rekursiven Graphen scheint der Conductance Index sehr gut zu funktionieren, da dieser Index die beiden „gegenläufig“ verlaufenden Indizes vereint. Demnach wird nur dann ein Clusterlevel hoch bewertet, wenn sowohl Inter- als auch Intracluster Conductance hinreichend hohe Werte haben. Es scheint, als ob durch die Verwendung eines eher Intercluster Kanten abhängigen und eines Intracluster Kanten abhängigen Indizes bessere Bewertungen der Clusterlevel entstehen.

Intracluster Conductance Es wäre von Vorteil, anstatt des Intracluster Conductance Index, den Performance Index für den Conductance Index zu benutzen. Zum einen auf Grund der Problematik, dass Intracluster Conductance aufwendig berechnet wird, zum anderen durch die in Kapitel 3.4 beschriebene Anomalie für die Bewertung von Cliquen.

Der Nachteil ergibt sich aus den Untersuchungen mit den rekursiven Graphen. Bei Verwendung des Performance Index würden auch schlechte Clusterlevel (größer als Level drei) sehr gut bewertet. Das Paar Inter- und Intracluster Conductance liefert hier bessere und eindeutigere Ergebnisse.

Kapitel 7

Abschließende Bemerkungen

7.1 Zusammenfassung

Wir untersuchten mehrere Instanzen verschiedener Graphtypen, bei denen eine signifikante Clusterung bekannt und gegeben war. Diese Clusterung ließen wir jeweils von den Clusterindizes Coverage, Performance, Intercluster Conductance und Intracluster Conductance bewerten. Dabei stellten wir fest, dass Coverage und Intercluster Conductance stärker die Intercluster Kanten bewerten, wohingegen Performance und Intracluster Conductance eher die Intracluster Kanten zum Bewerten der Clusterung benutzen. Die vorgegebenen signifikanten Clusterungen derjenigen Graphen, die man intuitiv als gut clusterbar bezeichnen würde, wurden von allen Indizes gut bewertet, wohingegen die Instanzen des Hyperwürfels, die nicht sehr gut clusterbar erschienen, nur von Intercluster Conductance und Coverage nicht gut bewertet wurden.

Ferner untersuchten wir rekursive Graphen, bei denen die intuitiv beste Clusterung nicht bekannt, aber die Anzahl der möglichen „guten“ Clusterungen stark begrenzt war. Bei diesen Graphen untersuchten wir, wie die verschiedenen Indizes die unterschiedlichen Clusterlevel bewerten. Hier stellten wir fest, dass kein Index als alleinige Bewertung einer Clusterung ausreicht, da Coverage und Intercluster Conductance immer das höchste Level, Performance und Intracluster Conductance immer das niedrigste Level als die beste Clusterung bewerteten.

Hieraus folgerten wir, dass ein alleiniger Index als Bewertungsfunktion für Clusterungen nicht ausreicht. Daher stellten wir den Conductance Index vor, der die Vorteile des Inter- und Intracluster Conductance Index vereinigt. Die-

ser Index scheint ein vielversprechender Amsatz für eine Bewertungsfunktion für Clusterungen bei Verwendung einiger Sukzessiv-Verfahren zu sein.

7.2 Ausblick

Als mögliche Weiterführung wäre nun interessant, die signifikanten Clusterungen der hier vorgestellten Graphtypen mit den von Sukzessiv-Verfahren gefundenen Clusterungen zu vergleichen. Hier könnte man untersuchen, inwieweit die verschiedenen Indizes für die Bewertung der Clusterungen die Ergebnisse verändern und ob bei Bewertung mit dem von uns vorgestellten Conductance Index die beste Clusterung gefunden wird. In diesem Zusammenhang wäre auch interessant, ob ein Indexpaar Intercluster Conductance und Performance ähnlich gute Werte liefert. Dies wäre in sofern interessant, da die Berechnung des Intracluster Conductance Index - auch mit den gegebenen Approximationsalgorithmen - aufwendiger als die Berechnung des Performance Index ist.

Außerdem wäre es interessant, Graphen zu untersuchen, die eine intuitiv gute Clusterung besitzen, die aber von dem Conductance Index schlecht bewertet werden. Ferner wäre dann zu untersuchen, aus welchen Gründen die Clusterung schlecht bewertet wird. Natürlich ist auch der umgekehrte Fall von Interesse. Also eine Clusterung eines Graphen, der schlecht clusterbar ist, und somit keine intuitiv gute Clusterung besitzt, die aber von dem Conductance Index gut bewertet wird. Auch hier wären die Ursachen für die schlechte Bewertung interessant.

Wir untersuchten nur ungerichtete Graphen. Man könnte natürlich auch gerichtete Graphen untersuchen, wobei bei diesen die vier hier vorgestellten Indizes entsprechend angepasst werden müssten.

Zudem könnte man noch Eigenschaften von einigen Graphen nutzen. Hier sei als Beispiel die Planarität gegeben. Hieran soll verdeutlicht werden, dass unsere Ergebnisse nur begrenzte Aussagekraft für planare Graphen haben, da man in diesem Falle zunächst noch klären müsste, was im planaren Fall ein „guter“ Cluster ist. Jedenfalls ist unsere Definition eines optimalen Clusters (eine Clique) auf planaren Graphen sicherlich nicht haltbar, da bereits eine Clique mit 5 Knoten nicht mehr planar eingebettet werden kann.

Man kann abschließend sagen, dass es in dem Bereich der Clusterung noch einiges zu erforschen gibt.

Literaturverzeichnis

[1] S. Li, C.M. Armstrong, N. Bertin: *A map of the interactome network of the metazoan C. elegans* Science. 2004 Jan 23;303(5657):540-3. Epub 2004 Jan 02.

[2] A.K. Jain, M.N. Murty, P.J. Flynn: *Data clustering: a review* ACM Computing Surveys 31 (1999) 264-323

[3] D. Jungnickel: *Graphen, Netzwerke und Algorithmen* 1994, BI-Wissenschaftsverlag; 3. Auflage

[4] I. Wegener: *Komplexitätstheorie: Grenzen der Effizienz von Algorithmen* 2003, Springer.

[5] R. Shamir, R. Sharan, D. Tsur: *Cluster Graph Modification Problems* In Proc. of 28th WG, number 2573 in LNCS, pages 379-390, 2002, Springer.

[6] R. Kannan, S. Vempala, A. Vetta: *On Clusterings: Good, Bad, Spectral* In Foundations of Computer Science 2000. (2000) 367-378

[7] U. Brandes, M. Gaertler, D. Wagner: *Experiments on Graph Clustering Algorithms* Proc. 11th Europ. Symp. Algorithms (ESA '03), Springer LNCS, to appear

[8] S. van Dongen: *Graph Clustering by Flow Simulation* PhD thesis, Universiteit Utrech, 2000. http://www.library.uu.nl/digiarchief/dip/diss/1895620/full.pdf

[9] M.R. Garey, D.S. Johnson: *Computer and Intractability: A Guide to the Theory of NP-Completeness* W. H. Freeman (1979)

[10] M. Gaertler: *Clustering with spectral methods* Master's thesis, Universität Konstanz, 2002. http://i11www.informatik.uni-karlsruhe.de/algo/people/gaertler/docs/thesis.pdf

[11] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein: *Introduction to Algorithms (Second Edition)* MIT Press (2001)

[12] M.R. Garey, D.S. Johnson, L.J. Stockmeyer: *Some simplified NP-complete graph problems* Theoretical Computer Science 1 (1976) 237-267