Clustering of dynamic graphs

Hauptseminar Networkvisualisation

Moritz Hamann

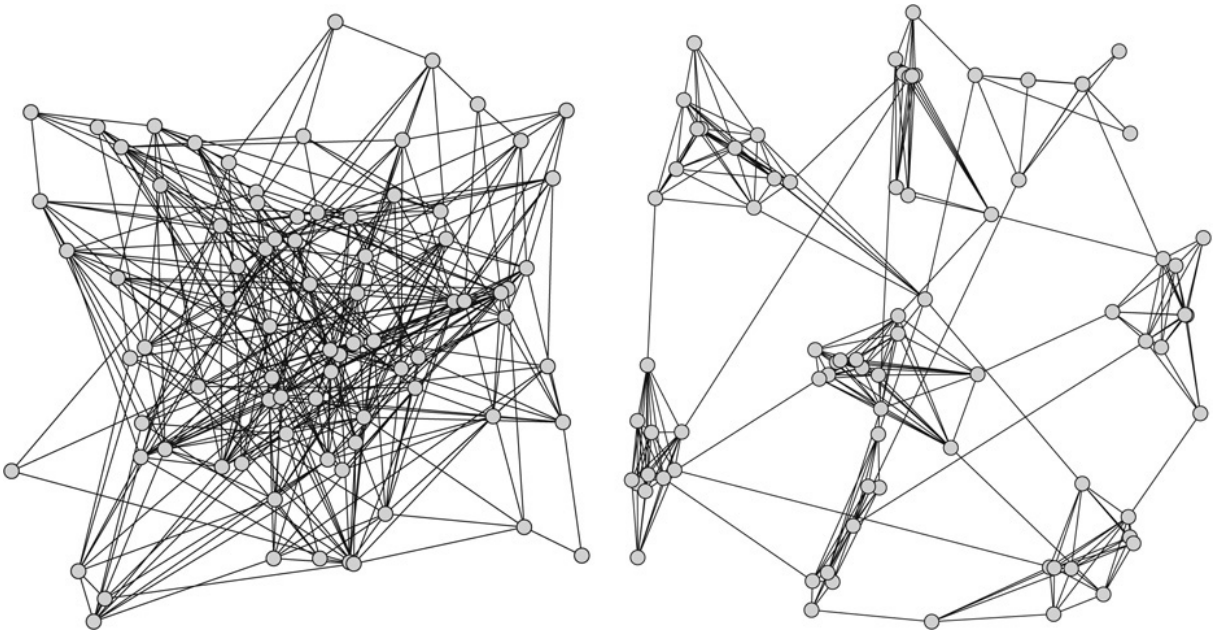

Abb. 1. Zwei Graphen mit je 84 Knoten und 358 Kanten, welche beide mit einem Spring-Force Algorithmus visualisiert wurden. Man sieht deutlich die lokalen Inhomogenitäten im rechten Graph, der eine “Relaxed Caveman” Struktur hat. Der linke Graph ist ein zufälliger Random Graph [12]

Kurzbeschreibung—Betrachtet man die Graphen realer Netzwerke stellt man fest, dass die Kantenverteilung lokal oftmals stark variiert. In solchen Graphen sind vor allem diejenigen Teilgraphen von Interesse, welche eine hohe Kantendichte haben und deren Knoten stark verknüpft sind. Diese Teilgraphen werden auch Cluster genannt.
In diesem Artikel wird eine Zusammenfassung verschiedener Verfahren gegeben, welche solche Cluster in einem Graph finden. Neben dem Clustering klassischer statischer Graphen, werden auch Verfahren vorgestellt, die auf zeitveränderlichen dynamischen Graphen operieren. Hierzu werden im ersten Teil die wichtigsten Eigenschaften von Cluster, k-Cliques und dynamischen Graphen erläutert. Anschließend wird eine Übersicht verschiedener Clusteringverfahren für statische Graphen präsentiert, ihre grundlegenden Funktionsweisen erläutert und anhand dieser eine Einteilung in verschiedene Klassen vorgenommen. Es werden exemplarische die Modularitätsoptimierung, sowie die Clique Percolation Method als zwei Vertreter dieser Algorithmen genauer erläutert.
Im letzten Teil werden zwei Verfahren vorgestellt, mit denen Cluster auch in dynamischen Graphen gefunden werden können. Neben einer Erweiterung der Clique Percolation Method, wird ein sogenanntes Time-Step Clustering präsentiert, welches ermöglicht beliebige statische Clusteringverfahren auch auf dynamische Graphen anzuwenden.

1 Motivation

Das mathematische Konzept der Graphen ist ein essentielles Modellierungswerkzeug in der Informatik. Damit lassen sich nicht nur verschiedenste Datenstrukturen anschaulich dargestellten, sondern sie eignen sich auch zur Modelierung und Untersuchung von Beziehungen zwischen einzelnen Objekten oder Prozessen in einem Netzwerk. Aus diesem Grund sind sie heutzutage nicht nur in der klassischen Informatik sowie in der Mathematik zu finden, sondern haben auch in vielen anderen Wissenschaften ihren Einzug erhalten. Sie werden genutzt um die Gruppendynamik in biologischen Netzwerken zu beschreiben, dienen als Kontrollalgorithmen für Multiagenten Systeme [9] und beschreiben Kommunikationsmuster in sozialen Netzwerken.

Um die Eigenschaften sehr großer Netzwerke analytisch untersuchen zu können, werden häufig Zufallsgraphen nach dem Model von Edgar Gilbert oder Erdos-Renyi verwendet. Diese Graphen haben die Besonderheit, dass die Wahrscheinlichkeit für eine Verbindung zwischen je zwei Knoten im gesamten Netzwerk konstant ist. Dadurch entsteht ein gleichmäßiger Graph, dessen Gradverteilung binomial verteilt sind, und somit die meisten Knoten die gleiche Anzahl an Kanten haben. Mit Hilfe der Wahrscheinlichkeitstheorie, lassen sich nun die Eigenschaften dieser Graphen auch für eine sehr hohe Anzahl an Knoten bestimmen und untersuchen.

Allerdings haben Untersuchungen realer Netze gezeigt, dass sich diese in den meisten Fällen von Zufallsgraphen unterscheiden. Reale Netzwerke sind häufig sogenannte Skalen freie Netze (im Englischen ’Scale-free networks’), in denen die Anzahl der Verbindungen pro Knoten nicht binomial verteilt ist, sonder einem Potenzgesetz folgt. Dadurch einsteht ein Netzwerk, in dem einzelne wenige Knoten eine große Anzahl an Verknüpfungen aufweisen, doch die Mehrzahl der Knoten weniger stark verknüpft ist. Weiterhin ist die Kantenverteilung zwischen den Knoten auch lokal sehr inhomogen, so dass sich Teilgraphen ausbilden, deren Knoten untereinander sehr stark bis komplett

Institut für Visualisierung und Interaktive Systeme, Universität Stuttgart. Sommersemester 2013.

verknüpft sind. Diese Teilgraphen werden auch 'Cluster' genannt.

Diese Cluster spielen in viele Anwendungsgebieten eine wichtige Rolle. Betrachtet man zum Beispiel den Graph der Freundschaftsbeziehungen in einem sozialen Netzwerk, lassen sich mithilfe von Angaben anderer Benutzer, sowie lokaler Cluster, unter anderem Rückschlüße auf gemeinsame Interessen, Wohnorte oder Freunde der einzelnen Benutzer ziehen. Diese Informationen bieten dem soziale Marketing eine bis vor kurzem unbekannte Menge an Möglichkeiten ihre Produkte zielgerichteter und persönlicher zu vermarkten.

2 GRUNDLAGEN

Dieses Kapitel erläutert die wichtigsten Eigenschaften und Merkmale von Clustern und dynamischen Graphen, und führt in das Konzept der k-Cliques ein.

2.1 Eigenschaften von Clustern

Der Artikel von S.E. Schaeffer [12] bietet eine umfassende Zusammenfassung über bisherige Clustering Verfahren, und versucht Cluster anhand gewünschter Eigenschaften zu definieren.

Betrachtet man den Teilgraphen Ω eines kompletten Graphen Υ, so müssen mehrere Bedingungen erfüllt sein, damit aus Ω ein guter Cluster wird. Mithilfe des Grads eines Knoten lässt sich eine erste wichtige Eigenschaft definieren. Der Grad $d(v)$ eines Knoten v ist definiert als die Anzahl der Kanten zu anderen Knoten im Graphen Υ. Ist nun $v \in \Omega$, so lässt sich der Grad in einen externen und internen Teil unterscheiden. Dabei ist der interne Grad die Anzahl der Kanten von v zu anderen Knoten aus Ω, der externe Grad die Anzahl der Kanten von v zu Knoten aus $\Upsilon \setminus \Omega$. Dabei gilt:

$$d_{int}(v,\Omega) = |\Gamma(v) \cap \Omega| \tag{1}$$
$$d_{ext}(v,\Omega) = |\Gamma(v) \cap (\Upsilon \setminus \Omega)| \tag{2}$$
$$d(v) = d_{int}(v,\Omega) + d_{ext}(v,\Omega) \tag{3}$$

wobei $\Gamma(v)$ die direkten Nachbarn von v sind. Für einen Cluster sollte nun gelten, dass das Verhältniss von internem zu externem Grad für alle Knoten $v \in \Omega$ hoch ist, d.h. die Knoten eins Cluster haben untereinander wesentlich mehr Verknüpfungen als zu den Knoten des restlichen Graphen.

Eine weiteres Kriterium für die Qualität eines Clusters ist die sogenannte interne Clusterdichte. Die allgemeine Dichte eines Graphen $\Upsilon = (V,E)$ mit der Knotenmenge V und der Kantenmenge E ist definiert als das Verhältnis der Summe aller Kanten durch die Anzahl aller möglichen Kanten in Graph:

$$\rho(\Upsilon) = \frac{|E|}{\binom{|V|}{2}} = \frac{2|E|}{|V|(|V|-1)} \tag{4}$$

Somit lässt sich die interne Clusterdichte eines Clusters Ω definieren als

$$\rho_{int}(\Omega) = \frac{|\{\{u,v\}|u,v \in \Omega\}|}{|\Omega|(|\Omega|-1)} \tag{5}$$

wobei $\{u,v\}$ eine Kante zwischen den Knoten u und v darstellt. Die fehlende 2 im Zähler im Vergleich zur allgmeinen Graphendichte ist dadurch zu erklären, dass $\{u,v\}$ und $\{v,u\}$ zwar die gleiche Kante darstellen, aber zwei unterschiedliche Elemente sind, wodurch die Anzahl der Kanten in $\{\{u,v\}|u,v \in \Omega\}$ verdoppelt wird.

Zusätzlich zur internen Clusterdichte, existiert noch eine sogenannte externe Clusterdichte zwischen verschiedenen Clustern Ω_i eines Graphen Υ. Sie ist definiert als das Verhältnis der Summe aller Kanten zu der Summe aller möglichen Kanten zwischen den verschiedenen Clustern Ω_i:

$$\rho_{ext}(\Upsilon|\Omega_1 ... \Omega_k) = \frac{|\{\{v,u\}|v \in \Omega_i\, u \in \Omega_j, i \neq j\}|}{|V|(|V|-1) - \sum_{l=1}^{k} |\Omega_k|(|\Omega_k|-1)} \tag{6}$$

wobei $|\Omega_i|$ die Anzahl der Knoten des Teilgraphen Ω_i darstellt. Im Allgemeinen sollte für ein gutes Clustering auf einem Graph Υ gelten:

$$\rho_{int}(\Omega_i) > \rho(\Upsilon) > \rho_{ext}(\Upsilon|\Omega_1 ... \Omega_k) \quad \forall i = 1...k \tag{7}$$

Abb. 2 zeigt drei verschiedene Cluster unterschiedlicher Qualität im Vergleich. Dabei representieren die schwarz hervorgehobenen, beliebig gewählten Teilgraphen jeweils einen Cluster. Der linke Cluster weist eine sehr hohe interne Dichte auf, und hat kaum Kanten mit Knoten außerhalb. Daher ist Qualität dieses Clusters sehr hoch. Der mittlere Cluster hat zwar die gleiche Anzahl an internen Kanten, weist aber im Gegensatz zum Linken eine wesentliche höhere Kantenanzahl zu Knoten außerhalb des Cluster auf. Zwar hat der rechte Cluster nur wenige Kanten nach außen, allerdings ist aber die Kantendichte innerhalb des Clusters minimal, was ihn zum schlechtesten Cluster der drei macht.

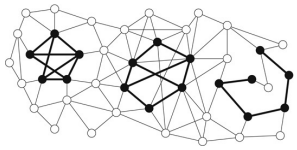

Abb. 2. Drei verschiedene Cluster unterschiedlicher Qualität [12]

2.2 k-Cliques

Einen Teilgraphen Ω mit k Knoten nennt man k-Clique, falls alle k Knoten dieses Teilgraphen direkt miteinander verbunden sind, und Ω somit vollständig ist [4]. Die entstehende Topologie des Teilgraphen Ω ist natürlich vom Parameter k abhänging. Abb. 3 zeigt k-Cliques für verschiedene Werte von k.

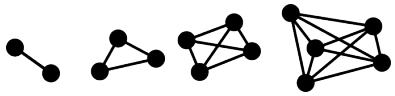

Abb. 3. Topologien für k-Cliques mit k=2,3,4,5

Da jede k-Clique vollständig ist, ist die in Kapitel 2.1 definierte interne Graphdichte mit $\rho_{int}(\Omega) = 1$ maximal. Somit stellen k-Cliques theoretisch gute Kandidaten für Cluster da. Beschränkt man sich allerdings bei der Clusterfindung auf einzelne k-Cliques, werden in den meisten Fällen nur Cliques für geringe Werte von k gefunden, da Forderung an einen vollständig verknüpften Cluster zu restriktiv ist.

2.3 Dynamische Graphen

Ein klassicher Graph $\Omega = (V,E)$ ist eine Menge an Knoten V und Kanten E zu einem bestimmten Zeitpunkt t. Für viele Anwendungen ist es aber wichtig das Netzwerk über einen Zeitraum mit mehreren Zeitschritten t_i zu betrachten und zu untersuchen.

Das Konzept der *dynamischen Graphen*[11] stellt die zeitliche Veränderung eines Graphen als geordnete Folge von statischen Teilgraphen für jeden Zeitpunkt $t = 1,...,n$ da:

$$\Upsilon = \{\Omega_1 = (V_1,E_1), \Omega_2 = (V_2,E_2), ..., \Omega_n = (V_n,E_n)\} \tag{8}$$

wobei Ω_i der Konfiguration des dynamischen Graphen Υ zum Zeitpunkt i entspricht. Da alle V_i, E_i für alle Werte von i unabhängig sind, ist es mögich das zu jedem Zeitschritt sowohl Kanten als auch Knoten hinzugefügt oder verschwinden können.

3 CLUSTERING IN STATISCHEN GRAPHEN

Viele Clustering Verfahren für dynamische Graphen basieren auf Verfahren zur Clusterung klassischer, statischer Graphen. Da dynamische Graphen aufgrund der Zeitanhängigkeit weitere Komplexität einführen, ist es sinnvoll als erstes diese statsichen Clusteringmethoden zu

betrachten. In diesem Kapitel wird versucht die Vielzahl verschiedener Verfahren anhand ihrer grundsätzlichen Eigenschaften in Kategorien zu unterteilen, sowie eine kurze Erklärung ihrer Funktionsweise zu geben.

Nach Schaeffer [12] lassen sich Clustering Verfahren grundsätzlich in lokale oder globale Methoden einteilen. Hierbei werden die Verfahren entweder global auf den ganzen Graph angewendet, oder nur lokal auf einen Teilgraphen. Entsprechend benötigen globale Verfahren Informationen über die Topologie des gesammten Graphen, während bei bei lokalen Verfahren nur rekursiv die Nachbarschaft eines einzelnen Knotens betrachtet wird, somit auch Netzwerke betrachtet werden können, die a priori nicht komplett bestimmt sind.

Somit bieten lokale Verfahren eine bessere Skalierbarkeit als Globale, falls das Clustering nur auf einen Teilgraphen angewendet werden soll, da die Topologie des restlichen Graphen nicht bekannt sein muss. Weiterhin haben diese Verfahren den Vorteil, dass das Clustering nur von der lokalen Struktur abhängt und eine lokale Änderung im Graphen auch nur das Clusterung in deren Umgebung beeinflusst. Deshalb eignen sich lokale Verfahren in Anwendungen, bei denen die Nachbarschaft einzelner Knoten schnell und häufig untersucht werden soll.

Tabelle 1. Einteilung der Clusteringverfahren für statische Graphen

	Globale Verfahren	Lokale Verfahren
Top-Down	Spektrale Methoden Random Walk Methoden Maximaler Fluss Methoden	
Bottom-Up	Modularitätsoptimierung Nächste Nachbarn Methode	CPM

Globale Verfahren

Die Familie der globalen Verfahren lässt sich noch mal in sogenannte *Top-Down*, sowie *Bottom-Up* Methoden [12] unterteilen. Bei Top-Down Verfahren wird ein großer Startcluster rekursiv anhand verschiedener Kriterien in immer kleinere Teilcluster unterteilt, während bei Bottom-Up Verfahren viele kleinere Cluster sukzessive zu größeren zusammen gefasst werden, bis das Clustering einem Abbruchkriterium genügt.

Ein Vertreter der Top-Down Methoden, sind die sogenannten *Spektralen Methoden*. Sie basieren auf den Eigenwerten und Vektoren der Laplace-Matrix eines Graphen $\Omega = (V,E)$. Diese ist definiert als:

$$L(\Omega) = D(\Omega) - A(\Omega) \tag{9}$$

wobei $D(\Omega) = diag(\{d(v_1),...,d(v_n)\})$ die Degreematrix von Ω ist [9]. $A(\Omega)$ ist die sogenannte Adjacency Matrix von Ω, und definiert als

$$[A]_{ij} = \begin{cases} 1 & \text{falls zwischen } v_i \text{ und } v_j \text{ eine Kante besteht} \\ 0 & \text{sonst} \end{cases} \tag{10}$$

Da die Laplace Matrix eine symmetrische, positiv semidefinite Matrix ist [9], sind alle Eigenwerte $\lambda_i \geq 0$ und die korrespondierenden Eigenvektoren bilden ein orthogonales System. Ordnet man die Eigenwerte in aufsteigender Reihenfolge $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, so folgt aus der positiv Semidefinitheit von L dass $\lambda_1 = 0$. Ist weiterhin $\lambda_2 > 0$ so existieren keine isolierten Teilgraphen im kompletten Graph[9]. Die Algorithmen der spektralen Clustering Methoden benutzen nun typischerweise die Komponenten des Fiedler Vektors -den Eigenvektor von λ_2- um die Knoten eins Graphen in zwei verschiedene Cluster zu unterteilen[12].

Ein weitere Gruppe von Methoden die den Top-Down Ansatz verfolgen, sind die sogenannten *Random Walk* oder *Markov Ketten Methoden*. Diese Verfahren basieren auf einem zufälligen Weg ξ fester Länge durch den Graph. Dabei wird ξ aufgebaut, indem rekursiv, beginnend von einem Startknoten v_{start}, jeweils ein Knoten der direkten Nachbarn hinzugefügt wird. Aufgrund der höheren Dichte innerhalb eines Clusters, wird der Weg ξ die Knoten des Clusters von v_{start} häufiger besuchen als Knoten ausserhalb des Clusters. Abb. 4 zeigt einen Beispielgraph mit zwei Clustern. Wird ein Weg ausgehend von einem weisen Knoten aufgebaut, ist es für einen zufälligen Weg nur auf einem Knoten möglich den linken Cluster zu verlassen, und somit werden die weisen Knoten des linken Clusters mit einer höheren Wahrscheinlichkeit besucht.

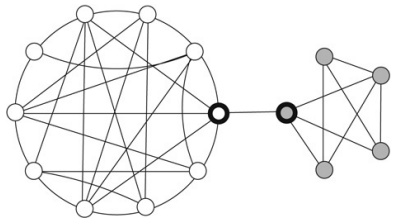

Abb. 4. Graph mit zwei Clustern. Beginnt man einen Random Walk bei einem Knoten des weisen Clusters ist die Wahrscheinlichkeit im nächsten Schritt einen weiteren weisen Knoten dieses Clusters zu erreichen höher, als den Cluster zu verlassen [12]

Für gewichtete Graphen eignen sich auch die *Maximaler Fluss Methoden* um ein Clustering duchzuführen. Sie gehören ebenfalls zu der Gruppe der Top-Down Verfahren, und versuchen einen minimalen Schnitt [12] des Graphen zu finden. Dazu werden Strömungsberechnungen auf dem Graphen durchgeführt, wobei ein Fluss zwischen zwei Knoten nur über eine Verbindungskante bestehen kann. Da aufgrund des *Minimum cut, maximum Flow Theorems* der Schnitt eins Graphen beim maximalen Fluss am geringsten ist, lässt sich dieser über den maximalen Fluss auf den Kanten des Graphen finden. Flake et al.[6] berechneten hierzu den Fluss mithilfe künstlich hinzugefügter Senken, und erzeugten daraus einen Minimum-cut Tree[7] um das Clustering durchzuführen.

Beispiele für Methoden, die einen Bottom-Up Ansatz verwenden sind unter anderem die *Modularitätsoptimierung* welche in Kaptiel 3.1 genauer behandelt wird. Eine weitere Vertreter ist die sogenannte *Nächste Nachbarn Methode*[12], welche häufig auch bei allgmeinen Klassifizierungsproblmen angewendet wird. Um die Nächste Nachbarn Methode auf Graphen anwenden zu können, muss eine Ähnlichkeit zwischen zwei Knoten definiert werden. Eine solche Ähnlichkeit kann z.B. anhand von vorhandenen Metadaten jedes Knoten erfolgen, oder anhand der Schnittmenge der direkten Nachbarn zweier Knoten. Im ersten Schritt wird für jeden Knoten des Graphen derjenige Nachbar gesucht, für welchen die Ähnlichkeit am grössten ist. Diese beiden Knoten bilden nun einen Cluster. In den darauffolgenden Schritten, werden nun die bereits gefunden Cluster auf Ähnlichkeit untersucht, und zu grösseren Clustern zusammengefügt, bis das Clustering beendet ist.

3.1 Modularitätsoptimierung

Das Verfahren der Modularitätsoptimierung ist ein weiterer Vertreter der Bottom-Up Ansätze. Es versucht eine Partitionierung des Graphen zu finden, für die die Modularität maximal wird. Hierbei ist die Modularität ein Maß, in wie fern die gewählte Partitionierung $C(\Omega)$ eines Graphen Ω einem guten Clustering entspricht. Mathematisch ist die Modularität $Q(C)$ definiert als [11]:

$$Q(C) = \frac{1}{2|E|} \sum_{u,v \in V} \left[A_{uv} - \frac{k_v k_u}{2|E|} \delta\left(c(u), c(v)\right) \right] \tag{11}$$

wobei $A_{uv} = 1$ falls eine Kannte zwischen u und v existiert, ansonsten 0. $k_u = d(u)$ ist der Grad des Knoten, und $c(u)$ diejenige Partition die den Knoten u enthält. Weiterhin ist $\delta\left(c(u), c(v)\right) = 1$ falls $c(u) = c(v)$, ansonsten 0. Somit repräsentiert die Modularität die Summe aller Kanten innerhalb einer Partition minus der Anzahl der Kanten, falls diese zufällig verteilt wären. Daraus folgt, das die Modularität in einem Random Graph für jede Partitionierung nahe 0 ist, und > 0, falls die Kan-

tendichte innerhalb einer Partition größer als eine zufällige, gleichmäßige Verteilung ist.

Der trivialste Ansatz diejenige Partitionierung $C(\Omega)$ zu finden welche die Modularität maximiert, wäre die Berechnung aller möglichen Partitionen und ihrer dazugehörigien Modularität. Es ist aber offensichtlich, dass bei steigender Graphgröße die Anzahl möglicher Partitionen rasant steigt. Weiterhin wurde gezeigt[2], dass das Problem eine optimale Partition zu finden NP-Vollständig ist.

Eine optimale Partitionierung, und somit ein Clustering kann mithilfe der Heuristik von Blondel et al. [1] approximiert werden. Der wesentliche Bestandteil der Heuristik ist die Tatsache, das die Änderung der Modularität ΔQ, falls ein Knoten v_i in den Cluster C verschoben wird, lokal effizient berechnet werden kann:

$$\Delta Q = \left[\frac{\sum_{in} + k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right] \tag{12}$$

wobei $\sum_{in}$ die Summe der Gewichte aller Kanten des Clusters C ist, $\sum_{tot}$ die Summe der Gewichte aller Kanten welche mit Knoten des Cluster C verbunden sind, k_i ist die Summe der Gewichte aller Kanten des Knoten v_i, $k_{i,in}$ ist die Summe der Gewichte aller Kanten welche den Knoten i mit Knoten aus C verbinden, und m die Anzahl aller Kanten im Graph. Hierbei bleibt zu erwähnen, das sogenannte *Selfloops*, also Kanten von v_i nach v_i erlaubt sind, und sogar ein wesentlicher Teil der Heuristik darstellen.

Die Heuristik läuft nun wie folgt ab:

1. Für jeden Knoten wird ein einzelner Cluster erstellt, der nur diesen einen Knoten enthält
2. Berechnung der ΔQ für das Verschieben eines Knoten v_i in die Cluster seiner direkten Nachbarn. Anschließend wird v_i in den Cluster desjenigen Nachbarn v_j verschoben, dessen ΔQ am größten ist. Ist ΔQ für jeden Nachbarn negativ, so bleibt der Knoten v_i in seinem Cluster. Dieser Schritt wird für alle Knoten solange wiederholt, bis sich ein lokales Maxima der Modularität einstellt. Dabei ist es durchaus möglich das ein Knoten mehrmals betrachtet wird.
3. Ist ein lokales Maxima erreicht, wird ein neuer Graph aufgebaut. Dabei entsprechen die bisherigen Cluster den Knoten des neuen Graph. Diese sind verbunden falls mindestens eine Kante zwischen je einem Knoten der beiden Cluster existiert und das Gewicht dieser Kante ist die Summe der Gewichte aller Kanten zwischen diesen Clustern. Kanten innerhalb eines Cluster führen zu einem Selfloop dessen Gewicht die Summe der Gewichte dieser Kanten ist.
4. Führe Schritte 1-3 wiederum auf den neuen Graph aus, solange bis die Modularität nicht weiter erhöht werden kann.

3.2 Clique Percolation Method (CPM)

Alle bisher vorgestellten Methoden waren globale Clusteringverfahren. Ein lokales Verfahren ist die *Clique Percolation Method (CPM)*[4], welche auf den in Kap. 2.2 vorgestellten k-Cliques beruht. Hierzu wird eine Nachbarschaftsbeziehung zwischen k-Cliques definiert, bei der zwei k-Cliques benachbart sind, falls diese sich $k-1$ Knoten teilen. Ein Cluster in der CPM ist definiert als eine maximale Menge von k-Cliques, welche über eine Nachbarschaft erreichbar sind. Dabei wird die CPM jeweils für ein festgelegtes k durchgeführt. Es exisitieren verschiedene Algorithmen mithilfe derer eine CPM durchgeführt werden kann, z.B. der Bron-Kerbosch Algorithmus.

Abb. 5 zeigt die gefundenen 3-Clique Cluster einer CPM für einen willkürlichen Graphen. Im linken sowie rechten Graph werden jeweils zwei Cluster gefunden, welche durch graue und schwarze Einfärbung hervorgehobenen sind. Im rechten Graph ist zu erkennen, dass sich Cluster, welche mithilfe einer CPM gefunden wurden, ein oder mehrere Knoten teilen können. Somit ist es auch möglich überlappende Cluster zu finden, welche vor allem in sozialen Netzwerken auftreten.

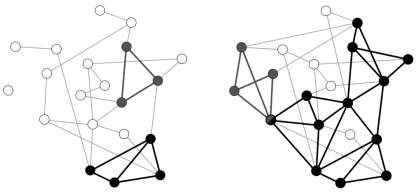

Abb. 5. Gefundene 3-Clique Cluster zweier zufälliger Graphen. In beiden Graphen werden jeweils zwei Cluster gefunden, welche schwarz und grau hervorgehoben sind [4]

4 Clustering in dynamischen Graphen

Beim Clustering eines dynamischen Graphens, ist neben der Identifikation der Cluster ansich auch deren zeitlichen Verlauf von Interesse. Hierzu müssen die bisher vorgestellten Clusteringverfahren für statische Graphen erweitert oder modifiziert werden, damit die gefundenen Cluster über die Zeitschritte verfolgt werden können.

Clusteringverfahren dynamischer Graphen lassen sich in zwei Klassen einteilen. *Evolutionäre Clustering Verfahren*[3] verwenden Graphinformationen mehrerer konsekutiver Zeitschritte um ein Clustering durchzuführen. Zum Beispiel verwendet die Erweiterung der CPM in Kap. 4.2 Graphinformationen des aktuellen, sowie des nächsten Zeitschritt. Ziel der evolutionären Clusteringverfahren ist es, ein über mehrere Zeitschritte konsistentes Clustering zu finden, in welchem weiche Übergänge zwischen Clustern der einzelnen Zeitschritte bestehen.

Die andere Klasse der Clusteringverfahren dynamischer Graphen wird hier *Time-step Clustering* genannt. Im Gegensatz zu den evolutionären Clusteringverfahren, wird hier der statische Graph jedes einzelnen Zeitschritts separat geclustert. Dies hat den Vorteil das statische Clusteringverfahren, somit auch die in Kap. 3 vorgestellten Verfahren, ohne Modifikation verwendet werden können. Da allerdings nur die Informationen des aktuellen Zeitschritt verwendet werden, können sich die gefunden Cluster in konsekutiven Zeitschritten sehr stark ändern.

Für Verfahren beider Klassen ist es notwendig gefundene Cluster verschiedener Zeitschritte zu vergleichen, um spezifische Cluster zeitlich zu verfolgen. Dazu kann der *Jaccard-Koeffizient* verwendet werden, mithilfe dessen zwei Mengen auf Ähnlichkeit untersucht werden. Für die Mengen A und B ist er definiert als [8]:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{13}$$

wobei $|A|$ die Anzahl der Elemente in A ist. Somit ist der Wertebereich des Jaccard-Koeffizient $[0,1]$ mit 1 als maximaler Übereinstimmung.

4.1 Evolution eins Clusters

Betrachtet man den zeitlichen Verlauf eines Clusters, so können verschiedene Ereignisse auftreten. Zu jedem Zeitpunkt kann ein neuer Cluster entstehen, ebenso kann sich ein bestehender Cluster auflösen. Weiterhin können neue Cluster entstehen, falls sich bestehende große Cluster teilen. Ebenso ist das Gegenteil möglich, und mehrere kleinere Cluster formieren sich im nächsten Zeitschritt zu einem Großen. Im einfachsten Fall bleib ein Cluster im nächsten Zeitschritt konstant, oder er wächst oder schrumpft indem neue Knoten hinzukommen oder wegfallen. In Abb. 6 sind alle möglichen Ereignisse noch einmal graphisch dargestellt.

4.2 Erweiterung der CPM

Palla et al. haben in [10] die in Kaptiel 3.2 vorgestellte Clique Percolation Method erweitert, um sie auf dynamische Graphen anzuwenden. Dabei wird ein evolutionärer Ansatz verwendet, welcher den Graphen Ω_t zum Zeitpunkt t, als auch Ω_{t+1} benutzt.

Im ersten Schritt wird mithilfe der CPM für statische Graphen jeweils ein Clustering für Ω_t und Ω_{t+1} durchgeführt. Anschließend wird

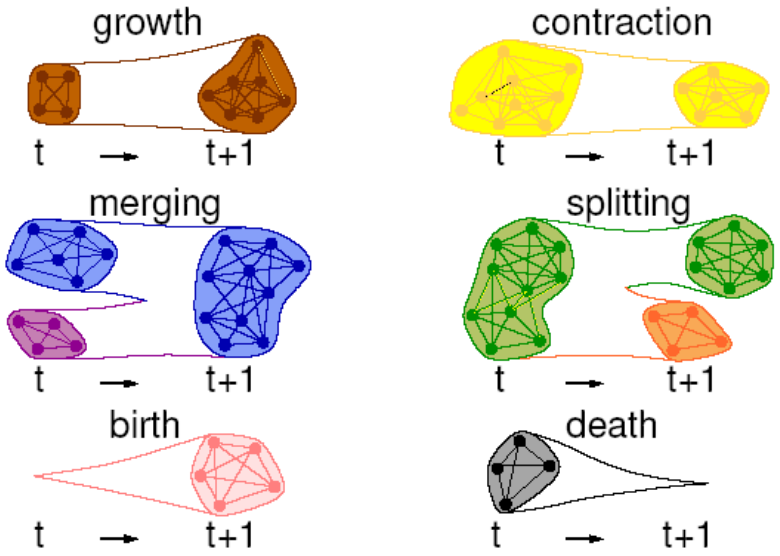

Abb. 6. Mögliche Ereignisse im zeitlichen Verlauf eines Clusters [10]

ein Verbundgraph $\Omega_{t,t+1} = \Omega_t \cup \Omega_{t+1}$ konstruiert, welcher alle Knoten und Kanten aus Ω_t und Ω_{t+1} enthält. Auf diesen Verbundgraph wird wiederum eine CPM angewendet. Da bei der Vereinigung zweier Graphen keine Kanten oder Knoten entfernt werden, existiert für jeden Cluster in Ω_t und Ω_{t+1} genau ein Cluster in $\Omega_{t,t+1}$. Enthält ein Cluster im Verbundgraph jeweils einen einzelnen Cluster aus Ω_t und einen einzelnen Cluster aus Ω_{t+1}, so können diese identifiziert werden. Falls ein Cluster des Verbundsgraph mehrere Cluster der aus Ω_t oder Ω_{t+1} enthält, werden die Cluster anhand des relativen Knoten Überlapps identifiziert. Details des Verfahren in [10].

In Abb. 7 werden die einzelnen Schritte der CPM Erweiterung graphisch dargestellt. Der Graph zum Zeitpunkt t wird in blau dargestellt, der Graph zum Zeitpunkt $t+1$ in gelb. Die Kreise zeigen die jeweils gefundenen Cluster an. Der Verbundgraph ist in der Mitte dargestellt. Grün gefärbte Knoten und Kanten sind dabei in beiden Zeitschritten vorhanden, blau und gelbe nur in den Zeitschritten entsprechender Farbe.

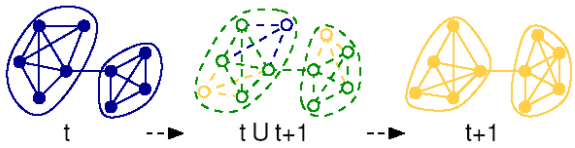

Abb. 7. Zwei Zeitschritte eines dynamischen Graphen. Der mittlere Graph ist die Vereinigung der Graphen beider Zeitschritte [10]

4.3 Dynamic Communities

Ein wichtiger Teil eines Time-Step Clusteringverfahrens ist neben dem eigentlichen Clustering, das Identifizieren gleicher Cluster über mehrere Zeitschritte. Hierzu haben Greene et al. in [8] ein generelles Framework vorgestellt, welches auf den in Kap. 4.1 gezeigten Ereignissen basiert. Da es weiterhin unabhängig vom verwendeten Clustering Verfahren ist, lässt es sich mit allen in Kap. 3 vorgestellten Verfahren kombinieren.

Betrachtet man einen dynamischen Graphen $\Upsilon = \{\Omega_1, \ldots, \Omega_n\}$ mit statischen Teilgraphen Ω_t zum Zeitpunkt t und wendet ein Time-Step Clustering an, erhält man ein Clustering $\mathbb{C}_t = \{C_{t,1}, \ldots, C_{t,k}\}$ für jeden einzelnen Zeitschritt.

Ein Kernkonzept des in [8] vorgestellten Verfahrens sind sogenannte *Dynamic Communities*. Diese Dynamic Communities D_i repräsentieren den zeitlichen Verlauf eines einzelnen Cluster C_i im Graph. Sie lassen sich als eine zeitlich geordnete Folge $D_i = \{C_{1,i}, \ldots, C_{n,i}\}$ einzelner Konfigurationen des Clusters C_i zu unterschiedlichen Zeitpunkten darstellen. Abb. 8 zeigt den Verlauf dreier Dynamic Communities über drei Zeitschritte. Die zeitlich letzte bekannte Konfiguration einer Dynamic Community wird die Front F_i genannt, welche für verschiedene D_i nicht unbedingt zum selben Zeitpunkt sein muss.

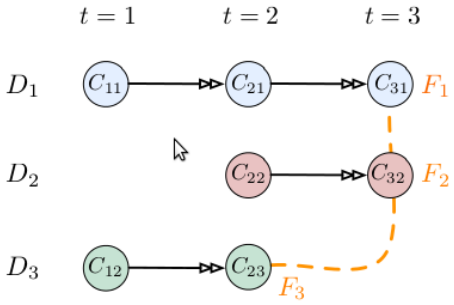

Abb. 8. Drei Dynamic Communities über drei Zeitschritte, bei denen Geburt und Todesereignisse auftreten.[8]

Die Dynamic Communities in Abb 8 sind:

- $D_1 = \{C_{1,1}, C_{2,1}, C_{3,1}\}$
- $D_2 = \{C_{2,2}, C_{3,2}\}$
- $D_3 = \{C_{1,2}, C_{2,3}\}$

Für jeden Zeitschritt t werden die gefundenen Cluster $C_{t,i}$ des Clustering $\mathbb{C}_t$ mit den Fronten der Dynamic Communities verglichen um gleiche Cluster in konsekutiven Zeitschritten zu finden. Hierzu wird der in Kap. 4 vorgestellte Jaccard-Koeffizient verwendet. Dabei sind die Front F_j und der Cluster $C_{t,i}$ gleich falls $J(F_j, C_{t,i})$ größer einem bestimmten Wert θ ist. Da es in dynamischen Graphen zu Teilungen und Zusammenschlüßen von Clustern kommen kann, ist es durchaus möglich, dass ein Cluster $C_{t,i}$ keiner, einer oder mehreren Fronten zugeordnet werden kann. Dabei gelten folgende Regeln für die einzelnen Ereignisse im zeitlichen Verlauf eines Clusters:

- **Geburt:**
  Falls ein Cluster $C_{t,i}$ zum Zeitpunkt t keiner bisherigen Front

zugeordnet werden kann, wird eine neue Dynamic Community D_i angelegt, welche $C_{t,i}$ enthält. Die Dynamic Community D_2 in Abb. 8 wird zum Zeitpunkt $t = 2$ geboren.

- **Tod:**
  Eine bestehende Dynamic Community D_i kann sich auch zu jedem Zeitschritt auch auflösen. Dies ist der Fall, falls in d konsekutiven Zeitschritten kein Cluster finden lässt, der der Front F_i entspricht. Nach dem Tod von D_i werden die gefunden Cluster nicht mehr mit F_i verglichen. Ist $d > 1$ so kann eine Dynamic Community mehrere Zeitschritte ohne einen entsprechenden Cluster des aktuellen Zeitschritts bestehen. Ein Beispiel für den Tod einer Dynamic Community ist D_3 in Abb. 8, falls für $t > 3$ keine Cluster gefunden werden welche F_3 entsprechen.

- **Zusammenschluss:**
  Zwei Dynamic Communities fallen zusammen, falls ein Cluster $C_{t,m}$ zwei Fronten F_i und F_j zugeordnet werden kann. Beiden Dynamic Communities D_i und D_j wird der Cluster $C_{t,m}$ hinzugefügt. Ein Beispiel hierfür sind D_1 und D_2 in Abb. 9.

- **Trennung:**
  Lassen sich zwei Cluster $C_{t,i}$ und $C_{t,j}$ der gleichen Front F_m zuordnen, so teilt sich die Dynamic Community im aktuellen Zeitschritt. Dabei wird D_m einer der beiden Cluster $C_{i,t}$ oder $C_{j,t}$ hinzugefügt, und eine neue Dynamic Community D_l angelegt, welche alle bisherigen Elemente von D_m enthält, und zusätzlich den anderen Cluster. Ein Beispiel sind die Dynamic Communities D_3 und D_4 in Abb. 9.

Somit sind die Dynamic Communities in Abb. 9:

- $D_1 = \{C_{1,1}, C_{2,1}, C_{3,1}\}$
- $D_2 = \{C_{1,2}, C_{2,1}, C_{3,1}\}$
- $D_3 = \{C_{1,3}, C_{2,2}, C_{3,2}\}$
- $D_4 = \{C_{1,3}, C_{2,3}, C_{3,3}\}$

Folgende Schritte werden somit im [8] vorgestellten Algorithmus durchgeführt:

1. Anwenden eines Clusteringverfahren auf den statischen Teilgraphen Ω_0 zum Zeitpunkt $t = 0$ des dynamischen Graphen Υ. Für jeden gefundenen Cluster wird eine Dynamic Community angelegt, welche diesen Cluster enthält.

2. Führe ein Clustering des nachfolgenden Zeitschritt aus, um ein Clustering $\mathbb{C}_t$ zu erhalten.

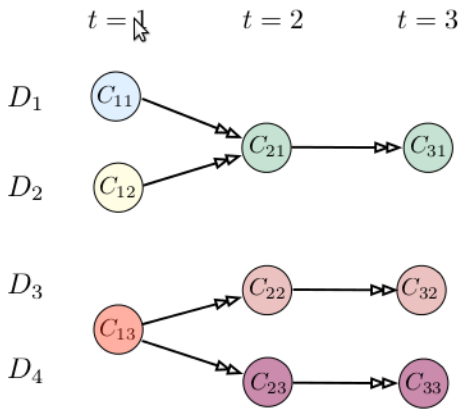

Abb. 9. Drei Dynamic Communities, bei denen ein Zusammenschluss und Trennungsereigniss auftritt [8]

3. Wende auf jeden Cluster $C_{i,t} \in \mathbb{C}_t$ folgende Schritte an:
   (a) Finde alle Dynamic Communities D_j, für die $J(C_{i,t}, F_j) > \theta$ ist.
   (b) Falls es keine solche Dynamic Community gibt, lege eine neue an welche $C_{i,t}$ enthählt.
   (c) Falls Dynamic Communities gefunden wurden, füge Jeder $C_{i,t}$ hinzu.

4. Falls einer bestehenden Dynamic Community D_i mehere Cluster zugeordnet wurden, wird für jeden zusätzlichen Cluster eine Kopie von D_i erstellte, welcher jeweils der jeweilige Cluster hinzugefügt wurde. Anschließend werden die Fronten F_i aller aktiven Communties auf den letzten bekannten Cluster aktualisiert.

5. Wiederhole ab Schritt 2 bis alle Zeitschritte verarbeitet wurden.

Hierbei entspricht der Schritt 3. (b) der Geburt einer neuen Community, der Schritt 3. (c) einem Zusammenschluss mehrere Communities falls Anzahl der übereinstimmenden Fronten > 1 ist. In Schritt 4. wird der Fall behandelt, falls sich eine bestehende Dynamic Community in mehrere aufteilt.

5 Fazit

In diesem Artikel wurde eine Übersicht verschiedenster Graphclustering Verfahren gegeben. Das Forschungsgebiet des Clustering statischer Graphen ist sehr gut erforscht, was die Vielzahl der vorgestellten Verfahren bestätigt. Hierbei stellen die Verfahren aus Kap. 3 nur eine sehr grobe Übersicht dar, da oftmals Variationen und zusätzliche Heuristiken existieren. Eine detaillierte Beschreibung, sowie weiterführende Literatur ist in [12] gegeben. In den letzten Jahren hat sich der Fokus der Forschung auf diesem Gebiet in Richtung dynamischer Graphen verschoben. Zwar ist es mit dem in [8] vorgestellten Framework mögliche, jedes statische Clustering Verfahren auch für dynamische Graphen anzuwenden, allerdings haben Time-Step Verfahren den Nachteil, dass gefunden Cluster durchaus über mehrere Zeitschritte sehr stark schwanken können.

Persönlich sehe ich das größte Potential in lokalen Verfahren wie der CPM. Diese bieten den Vorteil das der komplette Graph a priori nicht vorhanden sein muss, dass das Clustering außschließlich auf bestimmte Bereiche angewendet werden kann und das bei lokalen Änderungen nicht das komplette Clustering neu berechnet werden muss. Zwar ist die CPM aufgrund der maximalen Graphdichte der k-Cliques sehr restriktiv, allerdings existieren weitere Ideen, welche aus gefunden k-Cliques, sowie dem relativen Knotenüberlapp benachbarter k-Cliques einen gewichteten Clique-Graph[5] aufbauen, und somit neue Möglichkeiten zum Clustering bieten.

Literatur

[1] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. Juli 2008.
[2] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. Maximizing modularity is hard. August 2006.
[3] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. 2006.
[4] I. Derényi, G. Palla, and T. Vicsek. Clique percolation in random networks. April 2005.
[5] T. Evans. Clique graphs and overlapping communities. *Journal of Statistical Mechanics*, September 2010.
[6] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis. Graph clustering and minimum cut trees. April 2004.
[7] R. Gomery and T. Hu. Multi-terminal network flows. *SIAM Journal 9*, 1961.
[8] D. Greene, D. Doyle, and P. Cunningham. Tracking the evolution of communities in dynamic social networks. August 2010.
[9] M. Mesbahi and M. Egerstedt. *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press, first edition, 2010.
[10] G. Palla, A.-L. Barabási, and T. Vicsek. Quantifying social group evolution.
[11] A. Sallaberry, C. Muelder, and K.-L. Ma. Clustering, visualizing, and navigating for large dynamic graphs. September 2012.

[12] S. E. Schaeffer. Graph clustering. May 2007.