Identifikation von Clustern in Graphen

Robert Görke · Tanja Hartmann
Andrea Kappes · Dorothea Wagner

Einführung

Infolge des rasch anwachsenden Umfangs elektronisch zugänglicher Daten werden Methoden zur Lokalisierung relevanter Information und deren intelligente Organisation immer wichtiger. Für die Algorithmik stellt sich die Herausforderung, effiziente und praktikable Algorithmen zur Clusterung von Daten zur Verfügung zu stellen. In vielen Anwendungen liegen die Daten in Form von Netzwerken vor, wie zum Beispiel bei der Analyse sozialer Netzwerkstrukturen. Daher hat sich unter der Bezeichnung *Graphclustern* die Suche nach sinnvollen Clusterungen in Netzwerken bzw. Graphen als eigene Problemstellung etabliert. Basierend auf der Intuition, dass Kanten in Graphen eine Beziehung zwischen den entsprechenden Knoten modellieren, sollen die als Cluster identifizierten Teilgraphen hierbei gleichzeitig stark zusammenhängen und untereinander nur lose verbunden sein. Abbildung 2 zeigt ein klassisches Beispiel eines sozialen Netzwerks. Hierfür wurde die Beziehungsstruktur der Mitglieder eines Karateclubs durch Befragung erhoben. Als eines Tages aufgrund eines Streits zwischen Manager und Trainer die Teilung des Clubs bevorstand, lies sich die konkrete Aufteilung der Mitglieder bereits am strukturellen Zusammenhang des Netzwerks erahnen [16]. Eine andere Anwendung ist in Abb. 3 illustriert. Hier wurde Graphclustern dazu benutzt, Kundenprofile in einem Drogeriemarkt zu extrahieren.

Eine algorithmische Herangehensweise an die Problemstellung des Graphclusterns erfordert die Festlegung formaler Eigenschaften der gesuchten Clusterungen. Hierin liegt jedoch auch eine der Schwierigkeiten beim Graphclustern. Eine Vielzahl

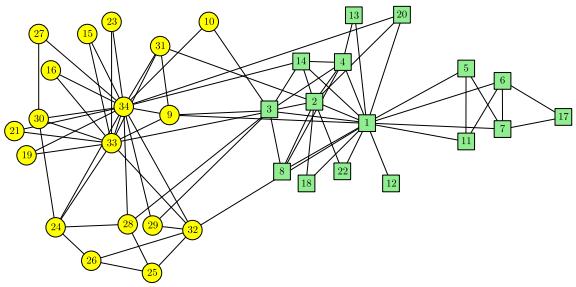

Abb. 2 Zacharys Karateclub. Die Knotenform kennzeichnet zwei stark zusammenhängende Gruppen

von Qualitätsmaßen versucht dieser Anforderung gerecht zu werden, jedoch gibt es trotz etablierter Maße und Algorithmen keinen allgemeingültigen Konsens bezüglich der Charakterisierung guter Clusterungen. Der wiederholten Einbeziehung der experimentellen Ergebnisse in den Entwurf und die Analyse von Algorithmen gemäß des Algorithm-Engineering-Kreislaufs kommt daher eine besondere Bedeutung bei der Entwicklung effizienter Verfahren zur Berechnung aussagekräftiger Clusterungen von Graphen zu.

Dieser Artikel beschränkt sich im Weiteren auf die Betrachtung disjunkter Cluster. Eine Clusterung $\mathcal{C}$ entspricht somit einer Partitionierung des Graphen $G = (V, E)$ ($|V| = n, |E| = m$). Der Einfachheit halber beziehen sich die folgenden

DOI 10.1007/s00287-013-0685-0

Robert Görke · Tanja Hartmann · Andrea Kappes
Dorothea Wagner
Karlsruher Institut für Technologie,
Institut für Theoretische Informatik,
Karlsruhe
E-Mail: {tanja.hartmann, andrea.kappes,
dorothea.wagner@kit.edu}

Zusammenfassung

Algorithm Engineering für Graphclustern beinhaltet mehr als die Entwicklung gut funktionierender Algorithmen für konkrete Anwendungen oder Datensätze. Es geht vielmehr um den systematischen Entwurf von Algorithmen für formal sauber gefasste Probleme und deren Analyse und Evaluation unter Betrachtung angemessener Qualitätsmaße. Die Wahl eines Qualitätsmaßes und eine dementsprechend saubere Formulierung eines Optimierungsproblems ist bereits für das intuitiv nahe liegende Paradigma eines starken Zusammenhangs innerhalb der Cluster gegenüber einem schwachen Zusammenhang zwischen den Clustern eine Herausforderung. Umso bedeutender ist der Erkenntnisgewinn, der aus der Methodik des Algorithm Engineering für Graphclustern erzielt werden kann. Viele Aspekte, die in diesem Artikel nur kurz angerissen werden, sind in der Arbeit [9] ausführlich beschrieben.

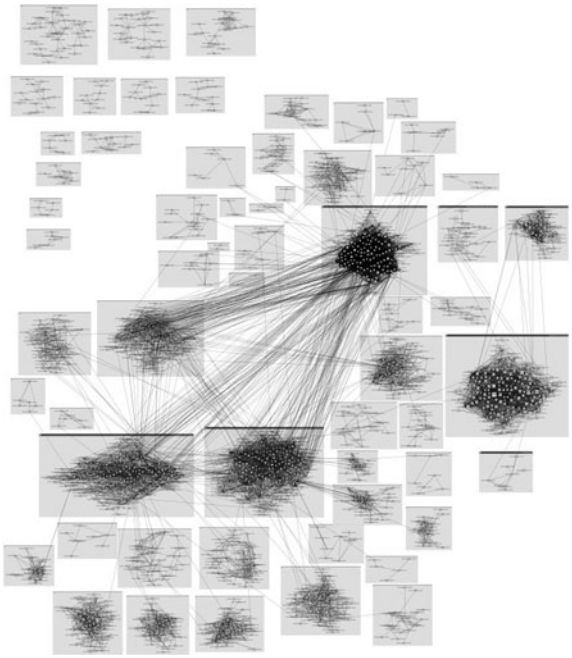

Abb. 3 Netzwerk der Ähnlichkeit von Kassenzetteln in einem Drogeriemarkt. Die Clusterung kann als Einteilung bezüglich Kundenprofilen aufgefasst werden

Ausführungen größtenteils nur auf ungewichtete Graphen. Die meisten hier vorgestellten Qualitätsmaße und Algorithmen lassen sich jedoch relativ leicht für die Anwendung auf gewichtete Graphen verallgemeinern.

Qualitätsmaße

Das im Bereich des Graphclusterns verfolgte Paradigma benennt als Merkmale einer gesuchten Clusterung *Intra-Cluster Density*, das heißt den starken Zusammenhang der Knoten innerhalb eines Clusters sowie *Inter-Cluster Sparsity*, also den schwachen Zusammenhang der Cluster untereinander. Daraus resultiert eine Fülle von Verhaltensweisen, die ein sinnvolles Qualitätsmaß intuitiv besitzen sollte. So sollte zum Beispiel die Bewertung einer Clusterung besser ausfallen, je mehr Kanten innerhalb der Cluster verlaufen und je weniger Kanten zwischen verschiedenen Clustern bestehen. Weiterhin sollten sich zusammenhängende Cluster positiv auf die Bewertung auswirken, während die Aufspaltung von Cliquen (vollständig verbundene Teilgraphen) in verschiedene Cluster eher negativ zu bewerten ist. Eine Clusterung bestehend aus unverbundenen Cliquen sollte mit maximaler Qualität bewertet werden. Dagegen würde man erwarten, dass zufällige Clusterungen eher schlecht abschneiden. Bei großen Graphen scheint es plausibel, ein Qualitätsmaß zur fordern, dessen Bewertung für einen lokalen Teil der Clusterung unverändert bleibt, auch wenn in einem anderen Teil eine neue Einteilung der Knoten in Cluster vorgenommen wird. Auch sollten sich die Bewertungen verschiedener Zusammenhangskomponenten nicht gegenseitig beeinflussen. Insbesondere sollte eine Clusterung, die aus zwei identischen Clusterungen auf zwei Kopien desselben Graphen besteht, weder schlechter noch besser eingestuft werden als die Einzelclusterungen der Kopien.

Bestehende und neue Maße müssen sich an diesen Kriterien messen lassen. Einige nahe liegende Maße setzen sich außerdem aus zwei getrennten Maßen für den Zusammenhang innerhalb und zwischen den Clustern zusammen, was eine Betrachtung des Graphclusterns als bikriterielles Optimierungsproblem nahe legt. Die nachfolgend beschriebenen Maße sind die nahe liegendsten und in der Literatur am häufigsten betrachteten Qualitätsmaße für das Graphclustern entsprechend dem eingangs formulierten Paradigma.

Abstract
Algorithmics is faced with the challenge of providing efficient and feasible algorithms for clustering relational data, i. e., graph clustering. This does not only encompass the development of algorithms that work well for specific applications or sets of data, but also the systematic design of algorithms for formally and rigorously defined problems, and their analysis and evaluation with respect to adequate criteria of quality. However, the choice of adequate quality measures and rigorous problem definitions are still difficult tasks even under consideration of the simple and intuitive paradigm of intra-cluster density vs. inter-cluster sparsity. The methodology of algorithm engineering can help to overcome this problem and result in important new insights in graph clustering.

Ansätze zur Zusammenhangsbewertung

Zur Bewertung des Zusammenhangs innerhalb und zwischen Clustern gibt es zwei grundlegende Ansätze. Der erste Ansatz assoziiert einen Zusammenhang mit der Anzahl vorhandener Kanten im Vergleich zur Anzahl potenziell möglicher Kanten in verschiedenen Kontexten. So beschreibt zum Beispiel die *Dichte* $\mathrm{dens}(G) = m/q$ eines Graphen G die Anzahl der vorhandenen Kanten m im Verhältnis zur Anzahl $q := 1/2(n(n-1))$ der Knotenpaare in G, also der Anzahl potenzieller Kanten. Daraus ergibt sich direkt die Dichte eines Clusters, indem man diesen als Teilgraph auffasst. Den Zusammenhang der Cluster untereinander, also die Dichte zwischen den Clustern, erhält man analog, indem man die Anzahl $\widehat{m}$ der tatsächlichen Kanten zwischen Clustern im Verhältnis zur Anzahl p der potenziellen Kanten zwischen Clustern betrachtet. Entsprechend gibt $p - \widehat{m}$ die Anzahl der nicht vorhandenen Kanten und damit richtig klassifizierten Knotenpaare zwischen den Clustern an. Das recht simple Maß *Coverage* $\mathrm{cov}(\mathcal{C}) = \overline{m}/m$ betrachtet nur die Anzahl $\overline{m}$ der von Clustern abgedeckten und damit richtig klassifizierten Kanten im Verhältnis zur Gesamtanzahl m an Kanten in G. Letzteres kann als maximal mögliche Anzahl der von Clustern abgedeckten Kanten bei Wahl einer beliebigen Clusterung interpretiert werden. Entsprechend wird Coverage offensichtlich genau bei der Wahl der trivialen Clusterung, die alle Kanten in einem einzigen Cluster vereint, maximal. Ohne eine zusätzliche Forderung an die Anzahl der Cluster ist Coverage alleine also kein sinnvolles Maß. Als Baustein in komplexeren Maßen findet Coverage jedoch durchaus Anwendung. Demgegenüber bewertet das Maß *Performance* $\mathrm{perf}(\mathcal{C}) = (\overline{m} + p - \widehat{m})/q$ die Gesamtqualität der Clusterung, indem es die Anzahl der insgesamt richtig klassifizierten Knotenpaare ins Verhältnis zur Gesamtzahl der Knotenpaare im Graph setzt. Ein Manko dabei ist, dass in Graphen mit wenigen Kanten, zu denen viele Echtweltdaten gehören, der Wert $p - \widehat{m}$ stark dominiert und somit sehr feine Clusterungen optimale Werte erzielen.

Der zweite Ansatz betrachtet die Größe von Schnitten in der Graphstruktur und bewertet deren Bedeutung anhand der Größe der getrennten Seiten. Schnitte können als Flaschenhälse interpretiert werden, die innerhalb von Clustern weit (viele Schnittkanten) und zwischen Clustern eng (wenige Schnittkanten) sein sollen. Auf dieser Sichtweise basierende Maße betrachten zum Beispiel die sogenannte *Conductance* geeigneter Schnitte. Die Conductance des Schnitts, der die Knotenmengen A und B trennt, setzt die Anzahl Schnittkanten $|E(A,B)|$ zwischen A und B ins Verhältnis zum kleineren Volumen der beiden Seiten, das (im ungewichteten Fall) als Summe der Knotengrade $\mathrm{vol}(A) = \sum_{v \in A} \deg(v)$ in A (bzw. B) definiert ist. Damit ist also die Conductance des Schnitts, der A und B trennt gerade $\mathrm{con}(A,B) = |E(A,B)| / \min\{\mathrm{vol}(A), \mathrm{vol}(B)\}$. Der kleinste Conductance-Wert unter allen Schnitten in einem Cluster ist somit ein Mindestmaß für den Zusammenhang innerhalb des Clusters und der Cluster mit dem geringsten Zusammenhang induziert eine untere Schranke für die Qualität der gesamten Clusterung. Die Conductance-Werte der Schnitte $(C, V \setminus C)$ können andererseits als Maß für den Zusammenhang zwischen den Clustern $C \in \mathcal{C}$ aufgefasst werden, welcher möglichst gering sein soll. Analoge Maße zur Bewertung von Clusterungen unter Betrachtung von Schnitten erhält man, wenn man statt der Conductance die sogenannte *Expansion* zugrunde legt, bei der statt der Summe der Knotengrade einfach die Knotenanzahl der beiden Seiten im Nenner steht.

Modularity

Eines der Maße, die sowohl die Dichte innerhalb der Cluster als auch die Abgetrenntheit zwischen den Clustern in einem einzigen Maß vereinen, ist

die 2004 von Newman und Girvan eingeführte *Modularity* [13]. Modularity basiert auf der schon angesprochenen Coverage, vergleicht diese jedoch mit dem Erwartungswert für die Coverage derselben Clusterung in einem Zufallsgraphen mit ähnlicher Gradverteilung. Die Idee dahinter ist, einer Clusterung nur dann einen positiven Wert zuzuordnen, wenn ihre Coverage höher ist als erwartet, also eine gewisse Signifikanz besitzt. Der Zufallsgraph entsteht, indem alle Kanten neu im Graphen verteilt werden, wobei beide Endknoten mit einer Wahrscheinlichkeit proportional zum Ursprungsgrad $\deg(v)$ eines Knoten v gewählt werden.[1] Dies führt zu folgender erwarteter Coverage:

$$\mathbb{E}\big(\mathrm{cov}(\mathcal{C})\big) = \frac{1}{4m^2} \sum_{C \in \mathcal{C}} \left(\sum_{v \in C} \deg(v) \right)^2$$

Die Modularity $\mathrm{mod}(\mathcal{C})$ einer Clusterung $\mathcal{C}$ ist dann die Differenz zwischen beobachteter Coverage und ihrem Erwartungswert:

$$\mathrm{mod}(\mathcal{C}) = \mathrm{cov}(\mathcal{C}) - \mathbb{E}\big(\mathrm{cov}(\mathcal{C})\big)$$

Clusterungen mit hoher Modularity weisen oft eine hohe Übereinstimmung mit menschlicher Intuition auf, weshalb das Maß schnell weite Anwendung zum Beispiel in der Analyse von Ökosystemen, wissenschaftlicher Kollaboration und der Biochemie gefunden hat und dort bis heute populär ist. Ein Kritikpunkt ist, dass die Frage, ob ein Cluster in einer Modularity-optimalen Clusterung enthalten ist, nicht nur von seiner unmittelbaren Umgebung abhängt, sondern zusätzlich von der Dichte des Gesamtgraphen. Dies hat zur Folge, dass Modularity in manchen Fällen intuitive und gut abgetrennte Cluster zu größeren Clustern verschmilzt. Dies geschieht zum Beispiel in dem in Abb. 3 abgebildeten Cliquenring. Ein weiterer Kritikpunkt ist, dass der für einen Graph erreichbare absolute Wert der Modularity wenig darüber aussagt, wie gut sich dieser tatsächlich in Cluster zerlegen lässt; Graphklassen, die intuitiv keine signifikante Clusterstruktur besitzen, wie Gitter oder Bäume mit kleinem Maximalgrad, besitzen Clusterungen, deren Modularity für steigende Graphgrößen gegen das absolute Maximum konvergiert [4]. Auch Clusterungen von Zufallsgraphen mit geringer Dichte können eine hohe Modularity aufweisen [12].

[1] Dabei können Mehrfachkanten und Schleifen entstehen.

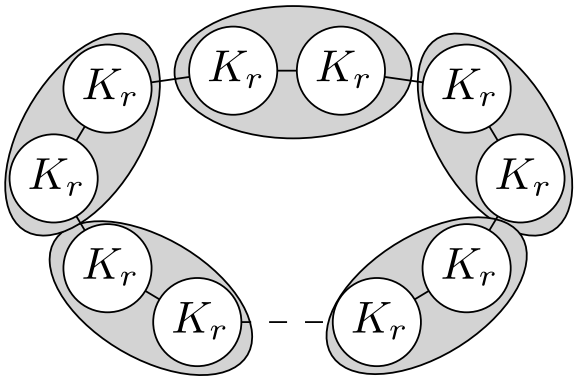

Abb. 3 Eine Menge von r-Cliquen, die jeweils mit einer Kante zu einem Ring verbunden sind. Wenn der Ring groß genug ist, werden in einer Modularity-optimalen Clusterung mehrere Cliquen zu einem Cluster zusammengefasst (aus [7])

Bikriterielles Optimierungsproblem

Statt der Suche nach guten Clusterungen bezüglich eines geschlossenen Qualitätsmaßes kann das Graphclustern auch als bikriterielles Optimierungsproblem aufgefasst werden. Die Idee hierbei ist, nur einen der beiden Aspekte, entweder den gewünscht starken Zusammenhang innerhalb der Cluster oder den gewünscht schwachen Zusammenhang zwischen Clustern, zu optimieren, während man für den jeweils anderen Aspekt eine gewisse Mindestqualität fordert. So könnte man zum Beispiel nur Clusterungen betrachten, bei denen alle Cluster eine gewisse untere Schranke für die Expansion oder Conductance erfüllen. Allerdings ist bereits die Berechnung der Expansion eines einzelnen Clusters NP-schwer, was die Überprüfung dieses Kriteriums schwierig macht. Ein einfacher abprüfbares Kriterium ist dagegen die Dichte der Cluster. Die Bewertung der gesamten Clusterung kann dann beispielsweise bezüglich eines globalen Dichtebegriffs oder auf Basis der minimalen Dichte oder der durchschnittlichen Dichte über alle Cluster vorgenommen werden. Für die Optimierung des Zusammenhangs zwischen den Clustern, der ja möglichst klein sein soll, bietet sich die Betrachtung von Trennschnitten zwischen den Clustern an. Interessant ist dabei einerseits der globale Schnitt, der die gesamte Partitionierung des Graphen in Cluster charakterisiert und andererseits sowohl Schnitte, die einzelne Cluster vom Rest des Graphen trennen als auch Schnitte, die Paare von Clustern trennen. Diese Schnitte können ebenfalls bezüglich verschiedener Maße, wie zum Beispiel Dichte zwischen Clustern, Expansion oder Con-

ductance, bewertet werden, was zu einer Vielzahl verschiedener Problemvarianten im Rahmen der bikriteriellen Optimierung führt. Eine systematische Untersuchung dieser Varianten wurde in [10, 11] mit dem Ergebnis vorgenommen, dass einzelne Kombinationen dieser Maße zu degenerierten Clusterungen führen, viele dieser Varianten sich aber ähnlich verhalten.

Algorithmen

Allen bisher angesprochenen formalen Definitionen für das Graphclustern liegt zugrunde, dass die zugehörigen Optimierungsprobleme inklusive der Optimierung von Modularity [2] NP-schwer sind. Um auch größere Graphen clustern zu können, bieten sich daher Heuristiken an, die in kurzer Zeit zwar keine optimalen, aber immerhin „gute“ Lösungen finden. Viele dieser Algorithmen lassen sich entweder als *divisive* oder als *agglomerative* Algorithmen charakterisieren.

Divisive Algorithmen verfahren nach dem *Top-Down*-Prinzip und teilen ausgehend von einem Cluster, das den gesamten Graphen enthält, die Knotenmenge in zwei oder mehrere Teile, wobei typischerweise nur wenige Kanten den Schnitt kreuzen und die einzelnen Teile nicht zu klein sein sollten. Diese Clusterung wird rekursiv *verfeinert*, bis eine dem betrachteten Optimierungsproblem entsprechende Clusterung erreicht ist. Agglomerative Algorithmen dagegen verfolgen das *Bottom-Up*-Prinzip und starten typischerweise mit einer Clusterung, in der jeder Cluster genau einen Knoten enthält. Diese Clusterung wird sukzessive *vergröbert*, indem jeweils Mengen von Clustern, die durch viele Kanten verbunden sind, verschmolzen werden. Auch hier wird dieser Prozess abhängig von der entsprechenden Zielfunktion irgendwann abgebrochen und die aktuelle Clusterung ausgegeben. Abbildung 4 veranschaulicht beide Vorgehensweisen.

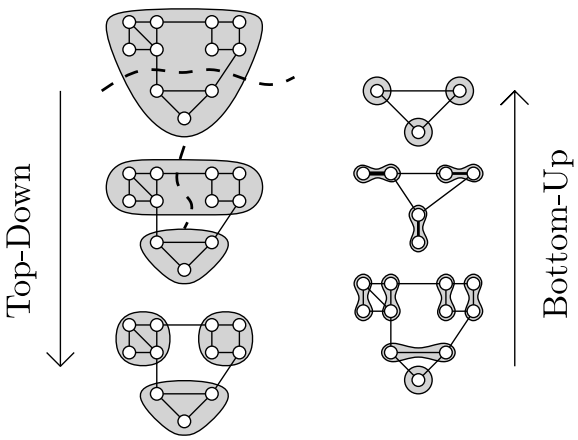

Abb. 4 Divisive (links) und agglomerative (rechts) Vorgehensweise

Greedy Merge

Eine grundlegende agglomerative Heuristik ist der *Greedy-Merge*-Algorithmus (manchmal auch als *Single-Linkage* bezeichnet), der iterativ Paare von Clustern verschmilzt. Eine naive Vorgehensweise würde zum Beispiel in jedem Schritt zwei Cluster, die mit einer maximalen Anzahl an Kanten verbunden sind, zusammenfassen. Da große Cluster jedoch typischerweise stärker verbunden sind als kleine Cluster, führt dies im Allgemeinen zu Clustern sehr unterschiedlicher Größe. Deshalb benutzt man üblicherweise Qualitätsmaße, um verschiedene Verschmelzungsoperationen zu bewerten. Beispielsweise wählt man zur Optimierung von Modularity in jedem Schritt zwei Cluster, deren Verschmelzung zu einer Clusterung mit möglichst hoher Modularity führt. Der Prozess bricht ab, sobald ein lokales Optimum erreicht ist. Dieses Vorgehen ist also ein klassischer gieriger Algorithmus. Beim Ablauf des Greedy-Merge-Algorithmus kann ein sogenanntes *Dendrogramm* erstellt werden, das die einzelnen Schritte dokumentiert. Dies ist ein binärer Wald, dessen Blätter den Knoten im Ausgangsgraphen entsprechen. Bei jedem Verschmelzungsschritt wird ein neuer Knoten eingefügt, der den neu entstehenden Cluster repräsentiert und mit den Ausgangsclustern verbunden wird. Abbildung 5 zeigt ein Beispiel für ein einfaches Dendrogramm. Das Dendrogramm ermöglicht dem Benutzer, Zwischenschritte des Algorithmus zu rekonstruieren und damit einzelne Cluster nach Wunsch weiter zu zerlegen.

Falls das zugrunde gelegte Qualitätsmaß die Eigenschaft hat, dass die qualitative Verbesserung, die mit einer Verschmelzung erzielt werden kann, nur von den beteiligten Clustern abhängt und nicht vom Rest der Clusterung, können potenzielle Verschmelzungen effizient in einer Prioritätswarteschlange verwaltet werden. In diesem Fall müssen nach jeder Verschmelzung $O(c)$ nicht mehr existierende Clusterpaare aus der Datenstruktur entfernt und $O(c)$ Paare, die durch den verschmolzenen

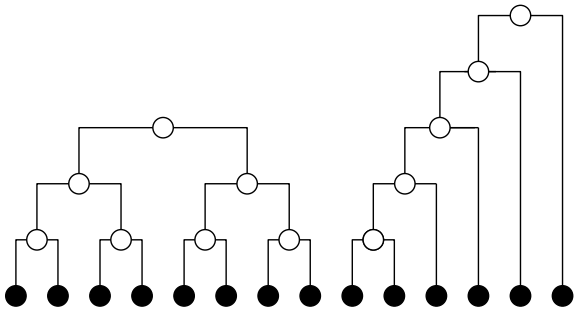

Abb. 5 Beispiel eines Dendrogramms der Höhe 5 mit zwei Clustern

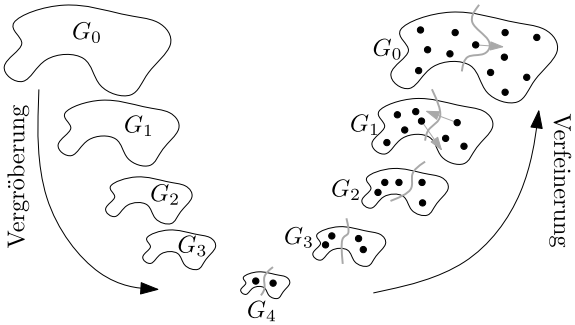

Abb. 6 Grobstruktur von Multilevel-Verfahren

Cluster entstehen, in die Datenstruktur eingefügt werden. Dabei bezeichnet c die Anzahl der Cluster vor der Verschmelzung. Falls zusätzlich die qualitative Verbesserung, die durch eine Verschmelzung erzielt wird, „einfach" zu berechnen ist, kann der komplette Algorithmus mit einer Worst-Case-Laufzeit von $O(n^2 \log n)$ implementiert werden. Einige Maße haben die Eigenschaft, dass das Verschmelzen von Clustern, zwischen denen keine Kanten bestehen, das Qualitätsmaß nie verbessern kann. In diesem Fall kann die Laufzeit auch durch $O(md \log n)$ abgeschätzt werden, wobei d die Höhe des Dendrogramms ist, das beim Ablauf des Algorithmus entsteht [3]. Dies trifft etwa für Modularity zu.

Prinzipiell kann dieses Verfahren auch für bikriterielle Optimierung benutzt werden, indem zum Beispiel eines der Maße unter der Einschränkung optimiert wird, dass das andere Maß einer gewissen Garantie genügen muss. Ein einfaches Beispiel wäre die Minimierung der Anzahl der Kanten zwischen Clustern, sodass die Dichte jedes Clusters einen gewissen Wert α nicht unterschreitet.

Multi-Level-Verfahren und Local Moving.

Analog zu Verfahren aus dem Bereich der Graphpartitionierung können agglomerative Clusteralgorithmen um eine *Verfeinerungsphase* erweitert werden und bilden damit ein *Multi-Level-Verfahren*[2] (s. Abb. 6). Dazu müssen zusätzlich zum Endergebnis des agglomerativen Verfahrens alle Zwischenergebnisse gespeichert werden, was eine Hierarchie von immer kleiner werdenden Graphen ergibt. In der Verfeinerungsphase wird dann ausgehend vom kleinsten Graphen in der Hierarchie, der dem Endergebnis des agglomerativen Verfahrens entspricht, die aktuelle Clusterung jeweils auf den nächstgrößeren Graphen projiziert. Im Anschluss daran wird diese Clusterung durch lokale Verbesserungsschritte weiter verbessert. Die Clusterung des größten Graphen ergibt dann das Endergebnis des Algorithmus. Im Kontext der Optimierung von Modularity hat sich dabei ein Multi-Level-Verfahren etabliert, bei dem beide Phasen auf lokalen Knotenverschiebungen beruhen. Dieses Verfahren schneidet in der Praxis sowohl bezüglich der Laufzeit als auch der Qualität besser ab als der entsprechende Greedy-Merge-Algorithmus [15]. Alternative Verfahren, die bezüglich Laufzeit und Qualität vergleichbar sind, basieren ebenfalls auf dem Bottom-Up-Prinzip [14, 15].

Cut Clustering

Das 2004 von Flake et al. vorgestellte *Cut-Clustering*-Verfahren [6] basiert auf der direkten Berechnung minimaler s-t-Schnitte in einem Graphen mit Kantengewichten. Zunächst wird in den Graphen G ein zusätzlicher Knoten t sowie Kanten von t zu jedem Knoten in G eingefügt. Diese zusätzlichen Kanten werden mit einem Wert α gewichtet, wobei α ein Eingabeparameter ist. Der Algorithmus berechnet nun auf dem erweiterten Graphen eine Menge von kreuzungsfreien minimalen s-t-Schnitten (s variabel, t fest), die in ihrer Gesamtheit t von allen Knoten in G trennen. Aus dem 1961 von Gomory und Hu vorgestellten Verfahren zur Berechnung eines Schnittbaums [8] folgt die Existenz einer solchen kreuzungsfreien Schnittmenge sowie eine einfache Berechnungsmethode. Nach Löschung von t induzieren diese Schnitte eine Clusterung von G. Die Cluster dieser Clusterung sind zum

[2] Im Gegensatz zur Graphpartitionierung fällt die Initiale Partitionierungsphase beim Clustern weg.

einen jeweils nur schwach zum Rest des Graphen verbunden, da sie durch minimale Schnitte induziert sind, zum anderen garantieren die eingefügten Zusatzkanten in G eine Mindestexpansion von α innerhalb der Cluster. Dies ist insbesondere interessant, da schon die direkte Berechnung der Expansion eines Clusters NP-schwer ist. In Abhängigkeit von der Wahl des Eingabeparameters α erhält man nicht nur Clusterungen unterschiedlicher Granularität, sondern für absteigende α-Werte bilden die zugehörigen Clusterungen sogar eine Hierarchie.

Andere Vorgehensweisen

Die hier vorgestellten Algorithmen bilden notgedrungen nur einen kleinen Ausschnitt der Literatur über das Graphclustern ab. Trotz der NP-Schwere der involvierten Clustermaße können kleine Instanzen in der Praxis durchaus optimal gelöst werden, beispielsweise mithilfe von ganzzahligen linearen Programmen. Auch wenn die Optimierung der genannten Maße oft schon dann NP-schwer ist, wenn man sich nur auf Clusterungen mit zwei Clustern einschränkt, können divisive Algorithmen in manchen Fällen zumindest eine gewisse Approximationsgüte garantieren, etwa für die Conductance.

Viele in der Literatur vorgestellte Algorithmen orientieren sich nicht an einem explizit formulierten Optimierungsproblem, sondern verfolgen stattdessen direkte Strategien, Cluster zu identifizieren. Ein möglicher Ansatz ist, iterativ Kanten zu entfernen, die „wahrscheinlich“ zwischen Clustern verlaufen. Dies können zum Beispiel Kanten sein, über die außergewöhnlich viele kürzeste Wege verlaufen. Die resultierenden Zusammenhangskomponenten identifizieren dann die Cluster. *Spektrale Algorithmen* hingegen bilden zunächst den Graphen auf eine Matrix ab und berechnen eine Menge von Eigenvektoren der Matrix. Die Knoten des Graphen werden dann mit Punkten identifiziert, deren Koordinaten durch die Einträge der Eigenvektoren bestimmt werden, und mit Standardmethoden aus dem Data Mining wie zum Beispiel *k-means* geclustert. Abhängig von der Art, wie die Matrix erstellt wird, kann der Erfolg von spektralen Methoden auf verschiedene Weise begründet werden, indem beispielsweise gezeigt werden kann, dass spektrales Clustern einer Relaxierung von schnittbasierten Maßen entspricht, oder Analogien zu Random Walks aufgezeigt werden. Random Walks können auch direkt benutzt werden, um Graphen zu clustern. *Markov Clustering* arbeitet beispielsweise direkt mit der Matrix der Übergangswahrscheinlichkeit der mit einem Random Walk im Graphen assoziierten Markovkette. Die Idee hierbei ist, Teilmengen an Knoten zu finden, aus denen ein Random Walker mit hoher Wahrscheinlichkeit lange nicht herausfindet. Einen detaillierteren Überblick über diese und andere Ansätze sowie die zugehörigen Referenzen finden sich in einem 2010 erschienenen Übersichtsartikel von Fortunato [7].

Aspekte des Algorithm Engineering

Wir haben in der Einführung bereits auf die Bedeutung des Algorithm Engineering für den Entwurf effizienter Methoden zum Graphclustern hingewiesen. Sowohl die Schwierigkeit, ein angemessenes Qualitätsmaß zu formulieren als auch die Tatsache, dass alle daraus resultierenden sinnvollen Optimierungsprobleme für das Graphclustern NP-schwer sind, legen den Einsatz von Methoden des Algorithm Engineering nahe. Da vielen Forschungsarbeiten in diesem Gebiet konkrete Anwendungen zugrunde liegen, wurden traditionell schon sehr häufig experimentelle Studien anhand von Anwendungsdaten durchgeführt, um die Qualität bzw. Effizienz von entsprechenden Heuristiken zumindest für diese Anwendungen zu belegen. Möchte man Graphclustern unabhängig von einer konkreten Anwendung als ein Optimierungsproblem bearbeiten, dient die experimentelle Evaluation im Sinne des Algorithm Engineering der Gewinnung neuer Einsichten über das Verhalten von Algorithmen, aber auch über die Eigenschaften von Qualitätsmaßen. Beispielsweise wurden die bereits im Abschn. „Modularity“ angesprochenen Erkenntnisse über das Verhalten von Modularity weitestgehend anhand experimenteller Arbeiten entdeckt.

Bedeutung von Benchmark-Daten

Beim Graphclustern ist die Wahl der für experimentelle Studien verwendeten Daten ein wichtiges und hochinteressantes Thema. Etwa die Einsicht, dass eine Optimierung der Modularity bei Graphen, die intuitiv keine signifikante Clusterstruktur besitzen (wie beispielsweise Gitter), zu sehr balancierten Clusterungen führt, wirft die Frage auf, anhand

welcher Daten experimentelle Studien über Algorithmen und Qualitätsmaße beim Graphclustern durchgeführt werden sollten. Die 2012 durchgeführte *10th DIMACS Implementation Challenge – Graph Partitioning and Graph Clustering*[3] und die daraus resultierende Arbeit über Benchmark-Daten [1] leistet einen entscheidenden Beitrag zur Beantwortung dieser Frage. Möchte man fundierte, anwendungsunabhängige Aussagen über Algorithmen zum Graphclustern treffen, sollte man nicht nur anhand umfangreicher und verschiedener Datensätze evaluieren, sondern auch künstlich generierte Datensätze verwenden. Letztere haben den Vorteil, dass man kontrollierte Experimente durchführen kann. So kann etwa die Größe der Graphen oder die Signifikanz der Clusterstruktur beliebig gewählt werden und die tatsächlich vorhandene Clusterstruktur, also die „Ground Truth", ist bekannt.

Vergleich von Clusterungen

Qualitätsmaße, wie wir sie hier diskutiert haben, ordnen einer Clusterung einen Wert zu. Möchte man die Ergebnisse verschiedener Algorithmen vergleichen, so ist die Aussagekraft dieser Werte allerdings sehr begrenzt. Vielmehr würde man gerne tiefere Einsichten über die berechneten Clusterungen gewinnen, indem man sie direkt miteinander vergleicht. Zum Vergleich verschiedener Clusterungen von (unstrukturierten) Datenpunkten stehen Vergleichsmaße zur Verfügung. Besonders bekannt ist der Index von *Rand*, der auf Abzählung von Paaren von Datenpunkten beruht, die in beiden Clusterungen gleich bzw. verschieden klassifiziert wurden. Ein weiteres Vergleichsmaß *Mutual Information* beruht auf dem Informationsgehalt einer Clusterung bzw. dem Verlust von Unsicherheit in einer Clusterung, wenn die andere Clusterung bekannt ist. Beim Vergleich von Graphclusterungen betrachten diese Vergleichsmaße allerdings nur die Zuordnung der Knoten zu Clustern, lassen aber die Graphstruktur außer Acht. Die Übertragung solcher Vergleichsmaße für Clusterungen von Datenpunkten auf Graphclusterungen stellt sich in der Arbeit [5] als eine eigenständige Engineering-Aufgabe heraus, die man ähnlich wie schon die Formulierung von Qualitätsmaßen für Graphclusterung an intuitiven Forderungen an solche Maße orientieren kann. So ist es etwa nahe liegend, zu fordern, dass der Vergleich zweier zufälliger Clusterungen desselben Graphen zu einem anderen Ergebnis führt als der Vergleich einer signifikanten Clusterung mit einer zufälligen Clusterung.

[3] http://www.cc.gatech.edu/dimacs10/

Ausblick

Die hier betrachtete Problemstellung des Graphclusterns setzt sich in viele Richtungen fort. Das Paradigma eines starken Zusammenhangs innerhalb der Cluster gegenüber einem schwachen Zusammenhang zwischen den Clustern lässt sich auch anders als hier diskutiert interpretieren. Statt der Forderung nach disjunkten Clustern, können überlappende Cluster angestrebt werden. Reale Netzwerke ändern sich über die Zeit, entsprechend sollten Graphclusterungen mit geringem Aufwand aktualisiert werden können, bei gleichzeitiger Kontrolle der Veränderung im Vergleich zur vorherigen Clusterung. Dies sind nur einige Beispiele an Fragestellungen, die in der aktuellen Forschung intensiv bearbeitet werden. Bei allen diesen Themen ist die Methodik des Algorithm Engineering ein Schlüssel zur Entwicklung praktikabler Algorithmen.

Danksagung

Diese Arbeit wurde in Teilen von der DFG im Rahmen der Projekte WA 654/19-1 und WA 654/15-1 finanziert.

Literatur

1. Bader DA, Meyerhenke H, Sanders P, Schulz C, Schumm A, Wagner D (2012) A benchmarking set for graph clustering and partitioning. In: Rokne J, Alhajj R (eds) Encyclopedia of Social Network Analysis and Mining. Springer, to appear
2. Brandes U, Delling D, Gaertler M, Görke R, Höfer M, Nikoloski Z, Wagner D (2008) On modularity clustering. IEEE T Knowl Data En 20(2):172–188
3. Clauset A, Newman MEJ, Moore C (2004) Finding community structure in very large networks. Phys Rev E 70(066111), online article
4. de Montgolfier F, Soto M, Viennot L (2011) Asymptotic modularity of some graph classes. In: Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC'11), Lecture Notes in Computer Science, vol 7074, Springer, pp 435–444
5. Delling D, Gaertler M, Görke R, Wagner D (2008) Engineering comparators for graph clusterings. In: Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management (AAIM'08), Lecture Notes in Computer Science, vol 5034, Springer, pp 131–142
6. Flake GW, Tarjan RE, Tsioutsiouliklis K (2004) Graph clustering and minimum cut trees. Internet Math 1(4):385–408
7. Fortunato S (2010) Community detection in graphs. Phys Rep 486(3–5):75–174
8. Gomory RE, Hu TC (1961) Multi-terminal network flows. Internet Math 9(4):551–570
9. Görke R (2010) An Algorithmic Walk from Static to Dynamic Graph Clustering. PhD thesis, Department of Informatics, Karlsruhe Institute of Technology, Karlsruhe, February
10. Görke R, Schumm A, Wagner D (2011) Density-constrained graph clustering. In: Proceedings of the 12th International Symposium on Algorithms and Data Structures (WADS'11), Lecture Notes in Computer Science, vol 6844, Springer, pp 679–690

11. Görke R, Schumm A, Wagner D (2012) Experiments on density-constrained graph clustering. In: Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX'12), SIAM, 2012, 1–15
12. Guimerà R, Sales-Pardo M, Nunes Amaral LA (2004) Modularity from fluctuations in random graphs and complex networks. Phys Rev E 70(025101), online article
13. Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69(026113):1–16
14. Ovelgönne M, Geyer-Schulz A (2010) Cluster cores and modularity maximization. In: 2010 IEEE International Conference on Data Mining Workshops (ICDMW), December 2010, IEEE Computer Society, pp 1204–1213
15. Rotta R, Noack A (2011) Multilevel local search algorithms for modularity clustering. ACM J Exper Algorithm 16:2.3:2.1–2.3:2
16. Zachary WW (1977) An information flow model for conflict and fission in small groups. J Anthropol Res 33(3–5):452–473