ALGORITHMISCHE METHODEN ZUR NETZWERKANALYSE

PROF. DR. HENNING MEYERHENKE

Vorwort

Dieses Skript dient nur der partiellen Ergänzung meiner Vorlesung. Es ist weder geeignet noch dazu gedacht, den Besuch der Vorlesung zu ersetzen.

1. Netzwerkeigenschaften

1.1. Gradfolgen.

1.2. Die k-Kern-Zerlegung eines Graphen.

Lemma 1. *Sei $G = (V, E)$ ein Graph, $(V_0, \ldots, V_k)$ seine Kern-Zerlegung und $G_i := (V_i, E_i) := G[V_i]$ sein i-Kern. Seien weiterhin $n_i := |V_i \backslash V_{i+1}|$ die Zahl der Knoten in der i-Schale und $m_i := |E_i \backslash E_{i+1}|$ die Zahl der Kanten, deren Endknoten mit minimaler Kernzahl zur i-Schale gehören. Per Definition gelte: $V_{k+1} := \emptyset, E_{k+1} := \emptyset$. Dann lässt sich die Größe der i-Schale beschränken durch:*

$$0 \leq n_i \leq |V| \tag{1.1}$$

$$\begin{cases} \lceil \frac{i \cdot n_i}{2} \rceil & , \text{ if } n_i > i \\ \binom{n_i}{2} + n_i(i - n_i + 1) & , \text{ if } n_i \leq i \end{cases} \leq m_i \leq \begin{cases} i \cdot n_i & , \text{ if } i < k \\ i \cdot n_i - \frac{i(i+1)}{2} & , \text{ if } i = k \end{cases} \tag{1.2}$$

Beweis. Ungleichung (1.1) ist trivial. Wir beweisen nun von (1.2) die erste Zeile. Seien dazu $V' := V_i \backslash V_{i+1}$ und $\deg^{(i)}(v) := \deg_{G_i}(v)$ der Knotengrad von v in G_i.

(1) Nach Konstruktion der Zerlegung muss $\deg^{(i)}(v) \geq i$ für alle $v \in V'$ gelten. Weiterhin muss $2|E_i \backslash E_{i+1}| \geq \sum_{v \in V'} \sum_{u \in N^{(i)}(v)} 1 = \sum_{v \in V'} \deg^{(i)}(v)$ gelten. Denn nur die Kanten mit beiden Endknoten in V_i werden nicht nur auf der linken Seite der Ungleichung, sondern auch rechts davon doppelt gezählt. Daraus folgt: $m_i \geq \frac{1}{2} \sum_{v \in V'} \deg^{(i)}(v) \geq \frac{1}{2} \sum_{v \in V'} i = \frac{i \cdot n_i}{2}$.

(2) Falls alle Kanten extern sind (also zwischen verschiedenen Schalen verlaufen), dann sind dies $i \cdot n_i$ Stück. Mehr als i externe Kanten kann ein Knoten der i-Schale nicht haben, da er sonst einer höheren Schale angehören würde.

□

Lemma 2. *Sei $G = (V, E)$ mit $|E| = m$ zusammenhängend, ungewichtet und schlicht. Dann kann der Algorithmus* CoreDecomposition*(G) in der Laufzeit $O(m)$ implementiert werden.*

Beweis. Das Berechnen und Speichern der Knotengrade in Zeile 3 ist offensichtlich in $O(m)$ Zeit möglich.

Damit die Schleife in Linearzeit läuft, brauchen wir eine geeignete Datenstruktur: Sie besteht zunächst aus einem Feld A der Länge n, wobei $A[i]$ ein Zeiger auf die

doppelt verkettete Liste L_i ist, die alle Knoten v mit $degree[v] = i$ speichert ($0 \leq i < n$). Außerdem gibt es eine Variable d_{min}, die das minimale i mit nicht-leerer Liste L_i speichert, und ein Feld $vptr$, dessen Einträge $vptr[v]$ auf den Eintrag von Knoten v in der passenden Liste L_i verweist. Vor dem Eintritt in die `while`-Schleife kann diese Datenstruktur in $O(n)$ Zeit konstruiert werden. Weil G zusammenhängend ist, gilt $m \geq n - 1$. Somit ist $\max\{n, m\} = O(m)$.

Der Pseudocode wird so erweitert, dass direkt nach Zeile 9 ein Knoten u in die Liste $L_{degree[u]}$ verschoben wird. Dies geht in konstanter Zeit. Ggf. muss zwar d_{min} auch verändert werden, aber dies geht nach Ergebnissen von Fiduccia und Mattheyses in amortisiert konstanter Zeit. Das Hauptargument hier ist, dass ein Ansteigen des Zeigers durch die Summe aller Knotengrade beschränkt werden kann, welcher $O(m)$ ist. In Zeile 6 können die Knoten mit Grad $< i$ jeweils in konstanter Zeit gefunden werden, indem man auf $L_{d_{min}}$ zugreift. Somit kostet ein Schleifendurchlauf für den Knoten v eine Laufzeit von $O(degree[v])$. Da jeder Knoten nur einmal betrachtet wird, ergibt sich insgesamt:

$$\sum_{v \in V} O(degree[v]) = \sum_{v \in V} O(deg(v)) = O(m).$$

□

1.3. **Clusterkoeffizienten.**

Theorem 3. *Für einen ungerichteten Graphen $G = (V, E)$ kann ein Wert $C_{local}^{approx}(G) \in [C_{local}(G) - \epsilon, C_{local}(G) + \epsilon]$ mit Wahrscheinlichkeit mindestens $\frac{\nu-1}{\nu}$ in der erwarteten Zeit $O(\frac{\log \nu}{\epsilon^2})$ berechnet werden.*

Beweis. Bzgl. Laufzeit siehe Tafel.

Um die Korrektheit unserer Wahl von k zu beweisen, benutzen wir Hoeffdings Schranke: Sei X_i eine unabhängige relle Zufallsvariable, die für alle i durch $0 \leq X_i \leq M$ beschränkt ist. Falls k die Größe der Stichprobe sowie $\epsilon > 0$ eine Fehlerschranke angeben, gilt:

$$\Pr\left(\left|\frac{1}{k}\sum_{i=1}^{k} X_i - \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right]\right| \geq \epsilon\right) \leq 2\exp\left(\frac{-2k\epsilon^2}{M^2}\right) \tag{1.3}$$

Wir müssen nun beweisen, dass der Erwartungswert $\mathbb{E}(l/k)$ gleich $C_{local}(G)$ ist und dass die gewünschten Schranken für unsere Wahl von k gelten.

Sei dazu $\Pi(G)$ die Menge aller Tripel in G und $\Pi(v)$ $[\Delta(v)]$ die Menge der Tripel [Dreiecke] mit v als Zentrum. Außerdem sei $\Upsilon \in \Pi(G)$ ein beliebiges Tripel von G. Dann ist $X(\Upsilon) = \begin{cases} 1 & \text{, falls } \Upsilon \text{ Dreieck} \\ 0 & \text{, sonst} \end{cases}$ eine Indikatorabbildung von $\Pi(G)$ nach $\{0, 1\}$. Weiterhin gilt:

$$\begin{aligned} C_{local}(G) &= \frac{1}{|V'|}\sum_{v \in V'} \frac{|\Delta(v)|}{|\Pi(v)|} \\ &= \frac{1}{|V'|}\sum_{v \in V'} \sum_{\Upsilon \in \Pi(v)} \frac{X(\Upsilon)}{|\Pi(v)|} \end{aligned}$$

Die Wahrscheinlichkeit, dass ein bestimmtes Tripel Υ mit Zentrum $v \in V'$ gezogen wird, ist $\frac{1}{|V'||\Pi(v)|}$. Also gilt wegen der Linearität des Erwartungswertes:

$$\begin{aligned}\mathbb{E}[l/k] &= \frac{1}{k}\sum_{i=1}^{k}\mathbb{E}[X(\Upsilon_i)] \\ &= \sum_{v\in V'}\sum_{\Upsilon\in\Pi(v)} X(\Upsilon)\cdot\frac{1}{|V'||\Pi(v)|} \\ &= \frac{1}{|V'|}\sum_{v\in V'}\frac{|\Delta(v)|}{|\Pi(v)|} \\ &= C_{local}(G).\end{aligned}$$

Da die Zufallsvariable $X(\Upsilon_i)$ eine Abbildung von $\Pi(G)$ nach $\{0,1\}$ ist, gilt $M = 1$ in Hoeffdings Schranke. Daher gilt mit unseren Werten für Ungleichung (1.3):

$$\begin{aligned}\Pr\left(\left|\frac{1}{k}\sum_{i=1}^{k}X_i - \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k}X_i\right]\right| \geq \epsilon\right) &\leq 2\exp\left(\frac{-2\lceil\ln(2\nu)/(2\epsilon^2)\rceil\epsilon^2}{1}\right) \\ &\leq 2\exp(-\ln(2\nu)) \\ &= \frac{1}{\nu}\end{aligned}$$

Daraus folgt die Behauptung für unsere Wahl von k. □

2. Eigenschaften von Knoten und Pfaden

3. Generierung von Graphen

Theorem 4. *(Erdös und Gallai) Eine* Partition $d_1 \geq \cdots \geq d_n > 0$ *einer natürlichen Zahl ist genau dann die* Gradfolge *eines schlichten ungerichteten Graphen, wenn*

$$\sum_{i=1}^{n} d_i \equiv 0 \mod 2 \tag{3.1}$$

und für alle $j \in [n]$ gilt:

$$\sum_{i=1}^{j} d_i \leq j(j-1) + \sum_{i=j+1}^{n}\min\{d_i, j\}. \tag{3.2}$$

Beweis. "⇐": (Fragment!) Falls keiner der Fälle 0-3 gilt, dann sind $v_1, \ldots, v_r$ paarweise adjazent und $d(v_k) = \min\{r, d_k\}$ für $k > r$. Da S unabhängig ist, gilt $\sum_{i=1}^{r} d(v_i) = r(r-1) + \sum_{k=r+1}^{n}\min\{r, d_k\}$. Wegen Ungleichung (3.2) ist $\sum_{i=1}^{r} d_i$ beschränkt durch die rechte Seite. Also haben wir das Defizit am Knoten r beseitigt und setzen iterativ mit dem nächsten kritischen Index fort. □

4. Clusteranalyse

4.1. **Zielfunktion Modularität.** Was die beste oder wenigstens eine gute Clusterung ist, kann sich deutlich von Problem zu Problem unterscheiden. Als Algorithmiker wollen wir aber natürlich Methoden entwickeln, die vielseitig und ohne großes Zutun eines Benutzers verwendbar sind. Ein vielfach genutztes Prinzip ist das Paradigma, dass Cluster intern *dicht* sein und nach außen nur wenige Verbindungen

haben sollen (*intra-dense vs inter-sparse paradigm*). Somit sollen Clustergrenzen möglichst wenige Kanten schneiden. Anders ausgedrückt sollen zwischen Knoten desselben Clusters viele Wege kurzer Länge verlaufen, zwischen Knoten verschiedener Cluster aber nur wenige solcher Wege.

Um auf algorithmischem Wege zu einer guten Clusterung zu kommen, formalisiert man die obigen abstrakten Vorstellung mit einer Zielfunktion. Dabei ist es auch wichtig zu beachten, dass die Cluster nicht völlig unterschiedliche Größe haben. Denn wenn man nur die Schnittgröße ohne jegliches Balancekriterium berücksichtigen würde, wäre das Abschneiden einiger Grad-1-Knoten immer optimal. Manche Zielfunktionen nehmen daher ein Balancekriterium explizit auf (wie etwa der normalisierte Schnitt), andere implizit.

Zu den Zielfunktionen, die dies implizit tun, gehört Modularität. Die Idee dahinter ist, den Anteil der Intra-Cluster-Kanten (diesen Ausdruck nennt man englisch auch *coverage*) mit dem erwarteten Anteil dieser Größe in einem Zufallsgraphen verglichen werden. Eine Clusterung, die aus nur einem Cluster besteht, hat dann Modularität 0, was ausdrückt, dass keine nennenswerte Clusterstruktur vorliegt. Dabei hat dieser Zufallsgraph dieselbe Gradverteilung wie der ursprüngliche Graph.

Definition 5. Sei $G = (V, E)$ ein Graph, dieser habe eine Clusterung $\mathcal{C}$, also eine Partition der Knotenmenge V. Die Modularität q von $\mathcal{C}$ auf G ist definiert als:

$$q(G, \mathcal{C}) := \sum_{C \in \mathcal{C}} \frac{|E(C)|}{m} - \left(\frac{\sum_{v \in C} \deg(v)}{2m} \right)^2 . \tag{4.1}$$

Bei der obigen Definition bezeichnet $E(C)$ die Menge der Intra-Cluster-Kanten des Clusters C. Modularität lässt sich alternativ auch so schreiben (der Nachweis der Äquivalenz beider Ausdrücke sei als Übungsaufgabe empfohlen):

$$q(G, \mathcal{C}) := \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{\deg(i) \deg(j)}{2m} \right) \delta(c(i), c(j)). \tag{4.2}$$

Der Cluster eines Knotens i wird dabei mit $c(i)$ bezeichnet. Außerdem stellt $\delta(\cdot, \cdot)$ das sogenannte Kronecker-Delta dar, welches genau dann 1 ist, wenn beide Parameter gleich sind und sonst 0. Modularität muss maximiert werden. Die Leserin kann sich leicht überzeugen, dass der Wert 1 das theoretische Maximum darstellt. Wie bereits angedeutet, weisen Werte um 0 auf eine fehlende Clusterstruktur hin. Das theoretische Minimum liegt nach Bisseling und Fagginger Auer bei $-\frac{1}{2}$.

Satz 6. *Modularität vergleicht die tatsächliche* coverage *und ihren Erwartungswert für jeden Cluster.*

Beweis. Dass der Teil links des Minuszeichens der coverage entspricht, ist in Formel (4.1) offensichtlich. Der Erwartungswert für den rechten Teil lässt sich wiederum leichter aus Formel (4.2) herleiten: Betrachten wir dazu eine beliebige Kante mit dem Ziel, zunächst die *Zahl* der erwarteten Intra-Cluster-Kanten herzuleiten.

Starte diese Kante am Knoten i. Wenn wir für diese Kante einen zufälligen anderen Endpunkt suchen, dann wird in unserem Nullmodell des Zufallsgraphen mit Wahrscheinlichkeit $\frac{\deg(j)}{2m}$ der Knoten j gewählt, weil die m Kanten von G insgesamt $2m$ Endpunkte haben. Wenn wir dieses Prozedere der Neuverdrahtung für alle Kanten des Knoten i durchführen, multipliziert man den Ausdruck noch mit $\deg(i)$. Wir erhalten also für alle Halbkanten: $\sum_i \deg(i) \sum_j \frac{\deg(j)}{2m}$.

Den Erwartungswert für den Anteil der Intra-Cluster-Kanten erhalten wir, indem wir für jedes geordnete Knotenpaar den Beitrag wie oben angegeben verwenden sowie eine Indikator-Variable, ob sie innerhalb eines Clusters verläuft. Letzteres erledigt der Ausdruck $\delta(c(i), c(j))$. Um jede Kante wiederum nur einmal zu zählen, brauchen wir noch die Normalisierung $\frac{1}{2}$ vor der Summe. Um *coverage*, also den *Anteil* der Intra-Cluster-Kanten, zu erhalten, teilen wir schließlich noch durch m. □

Theorem 7. *Das Problem Modularity ist streng $\mathcal{NP}$-vollständig.*

Beweis. (Rest) Wir zeigen, dass gilt: 3-Partition $\leq$ Modularity. Beachten Sie, dass $G(A)$, der in der Reduktion aus A konstruierte Graph, polynomielle Größe in der *unären* Kodierungsgröße von A hat. Da 3-Partition streng $\mathcal{NP}$-vollständig ist, reicht eine solche Transformation für unsere Zwecke aus.

Nach den Vorbetrachtungen an der Tafel und auf den Folien bleibt noch zu zeigen, dass sich aus der Lösung des einen Problems eine Lösung für das jeweils andere Problem ergibt.

"$\Leftarrow$" Falls es eine Clusterung $\mathcal{C}$ gibt mit

$$q(\mathcal{C}) \geq K(A) = 1 - \frac{2k-2}{k(a+1)} - \frac{ka^2(a+1)^2}{k^2a^2(a+1)^2} = \frac{(k-1)(a-1)}{k(a+1)},$$

dann wissen wir, dass diese Clusterung die Element-Knoten perfekt auf die k Cliquen-Cluster aufteilen muss. Weil jedes Element in genau einem Cluster enthalten ist, erzeugt dies eine Lösung für die 3-Partition-Instanz. Mit dieser Wahl von $K(A)$ ist also die Instanz $(G(A), K(A))$ von Modularity nur dann erfüllbar, falls auch die Instanz A von 3-Partition erfüllbar ist. Oder umgekehrt: Ist A nicht erfüllbar, dann auch nicht $(G(A), K(A))$.

"$\Rightarrow$" Für die andere Beweisrichtung nehmen wir an, dass die Instanz A von 3-Partition erfüllbar ist. Dann gibt es eine Partition in k Mengen derart, dass die Summe über jede Menge $\frac{1}{k} \cdot a = b$ ist. Wenn wir den zugehörigen Graphen clustern, indem wir die Element-Knoten jeder Menge mit einer anderen Clique vereinigen, erhalten wir eine Clusterung mit Modularität $K(A)$. Also ist dann auch die Instanz $(G(A), K(A))$ von Modularity erfüllbar. □

Die Definition von Modularität sowie das Ergebnis aus Theorem 7 lassen sich im Übrigen leicht auf gewichtete Graphen übertragen.