

Data Mining

Community Detection

Dr. Hanna Köpcke
Wintersemester 2020

Abteilung Datenbanken, Universität Leipzig
<http://dbs.uni-leipzig.de>

Übersicht

Hochdimensionale Daten

Clustering

Dimensions-
reduktion

Empfehlungs-
systeme

Assoziations-
regeln

Locality Sensitive
Hashing

Supervised ML

Graphdaten

Community
Detection

PageRank

Web Spam

Datenströme

Windowing

Filtern

Momente

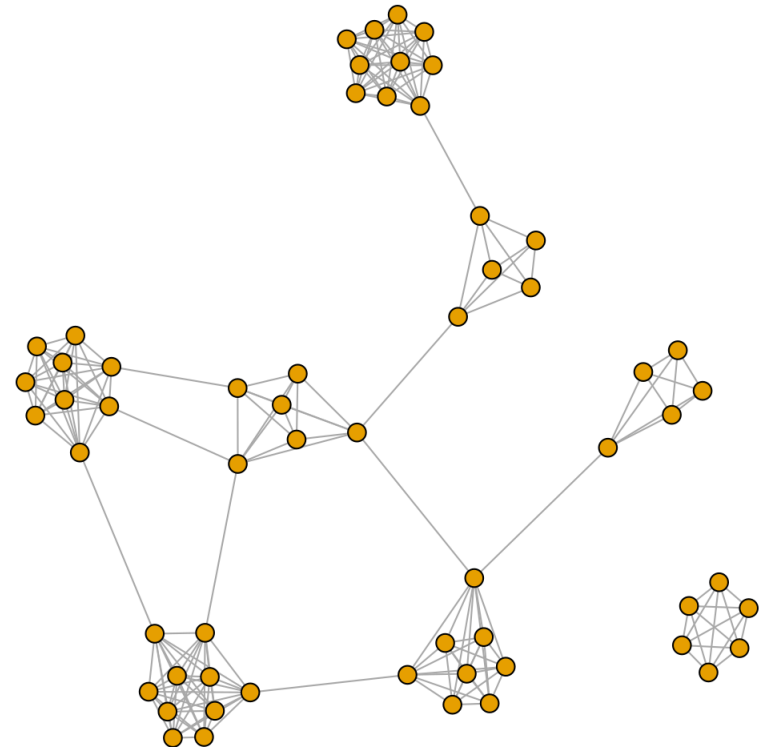
Web Advertising

Inhaltsverzeichnis

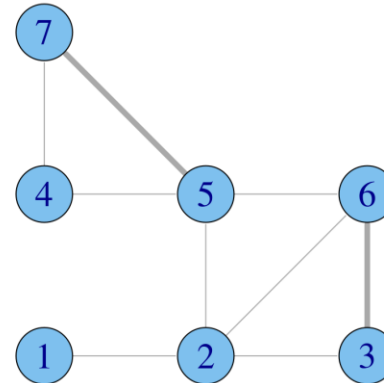
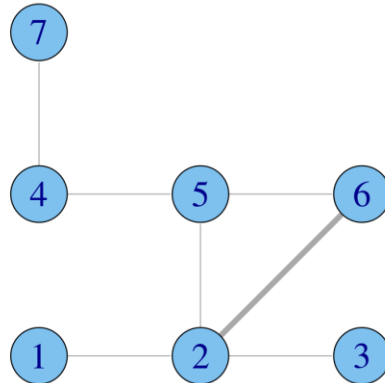
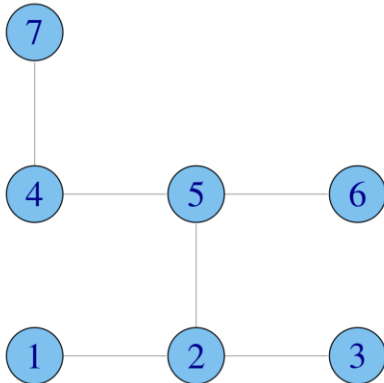
- **Einführung**
- **Dreiecke zählen**
- **Community Detection**
 - Girvan-Newman Algorithmus
 - Spectral Clustering

Graphen

- Graphen bestehen aus Knoten und Kanten
- Graphen sind entweder gerichtet oder ungerichtet
 - Gerichtet: Kanten haben Richtung (z.B. Follower-Netzwerk von Twitter)
 - Ungerichtet: Kanten ohne Richtung (z.B. Freundschaften auf Facebook)
- **Soziale Graphen**
 - Knoten repräsentieren Akteure (insb. Personen und Gruppen)
 - Kanten repräsentieren eine Beziehung zwischen Akteuren, z.B. Freundschaft, Kommunikation, Mitgliedschaft, ...
- Häufige Beobachtung:
Small-World-Phänomen
 - Geringe durchschnittliche Distanz **und**
 - Gruppierungen



Transitivität



- Wenn eine Kante zwischen **x** und **y** und eine Kante zwischen **y** und **z** existiert, dann ist die Existenz einer Kante zwischen **x** und **z** wahrscheinlicher als in einem *Zufallsgraph*
 - Sei **m** die Anzahl der Kanten und **n** die Anzahl der Knoten
 - Ungerichteter Zufallsgraph: Wahrscheinlichkeit, dass eine bestimmte Kante existiert, ist $\frac{m}{\binom{n}{2}}$
- Erweiterung: Die Wahrscheinlichkeit der Existenz einer Kante zwischen x und z ist umso höher, je mehr gemeinsame Kontakte x und z haben

Inhaltsverzeichnis

- Einführung
- Dreiecke zählen
- Community Detection
 - Girvan-Newman Algorithmus
 - Spectral Clustering

Dreiecke zählen

- Dreieck: Vollständig verbundene 3-elementige Untermenge der Knoten
- Erwartete Anzahl der Dreiecke in Zufallsgraph: $\text{ca. } \frac{4}{3} \left(\frac{m}{n}\right)^3$
- Höhere Anzahl an Dreiecken ist Indikator für soziales Netzwerk
- Annahmen
 - Prüfung auf Existenz einer Kante in $O(1)$
 - Sei d_i die Anzahl der Kanten, die von einem Knoten i ausgehen (*Grad des Knotens*), dann können alle benachbarten Knoten dieses Knotens in $O(d_i)$ ermittelt werden
- 1. Ansatz: Prüfe alle 3-elementigen Mengen von Knoten auf die Existenz eines Dreiecks – $O(n^3)$
- 2. Ansatz: Prüfe alle Kanten e und alle Knoten u , ob zwischen den beiden Enden von e und u jeweils eine Kante besteht - $O(nm)$
- Effizientere Methode?

Heavy Hitters

- Heavy Hitter = Knoten mit mindestens \sqrt{m} Kanten (Grad $d_i \geq \sqrt{m}$)
- Es kann nicht mehr als $2\sqrt{m}$ Heavy Hitters in einem Graphen geben, da Summe aller Grade $\sum_i d_i = 2m$
- 1. Schritt:
 - Zähle alle Dreiecke, die in Mengen aus 3 Heavy Hitters vorkommen
 - $O((2\sqrt{m})^3) = O(m^{1.5})$
- 2. Schritt (Dreiecke mit maximal 2 Heavy Hitters)
 - Betrachte jede Kante mit den Knoten u und v wobei $d_v \geq d_u$
 - Ignoriere Kante, falls beide Enden Heavy Hitters
 - Sonst: Betrachte alle (max. $\sqrt{m} - 1$) Kanten von u und zähle, wie viele der anderen Enden mit v verbunden sind
 - $O(m\sqrt{m}) = O(m^{1.5})$
- $O(m^{1.5}) \leq O(n^3)$ und $O(m^{1.5}) \leq O(nm)$

Verteilte Berechnung in sehr großen Graphen

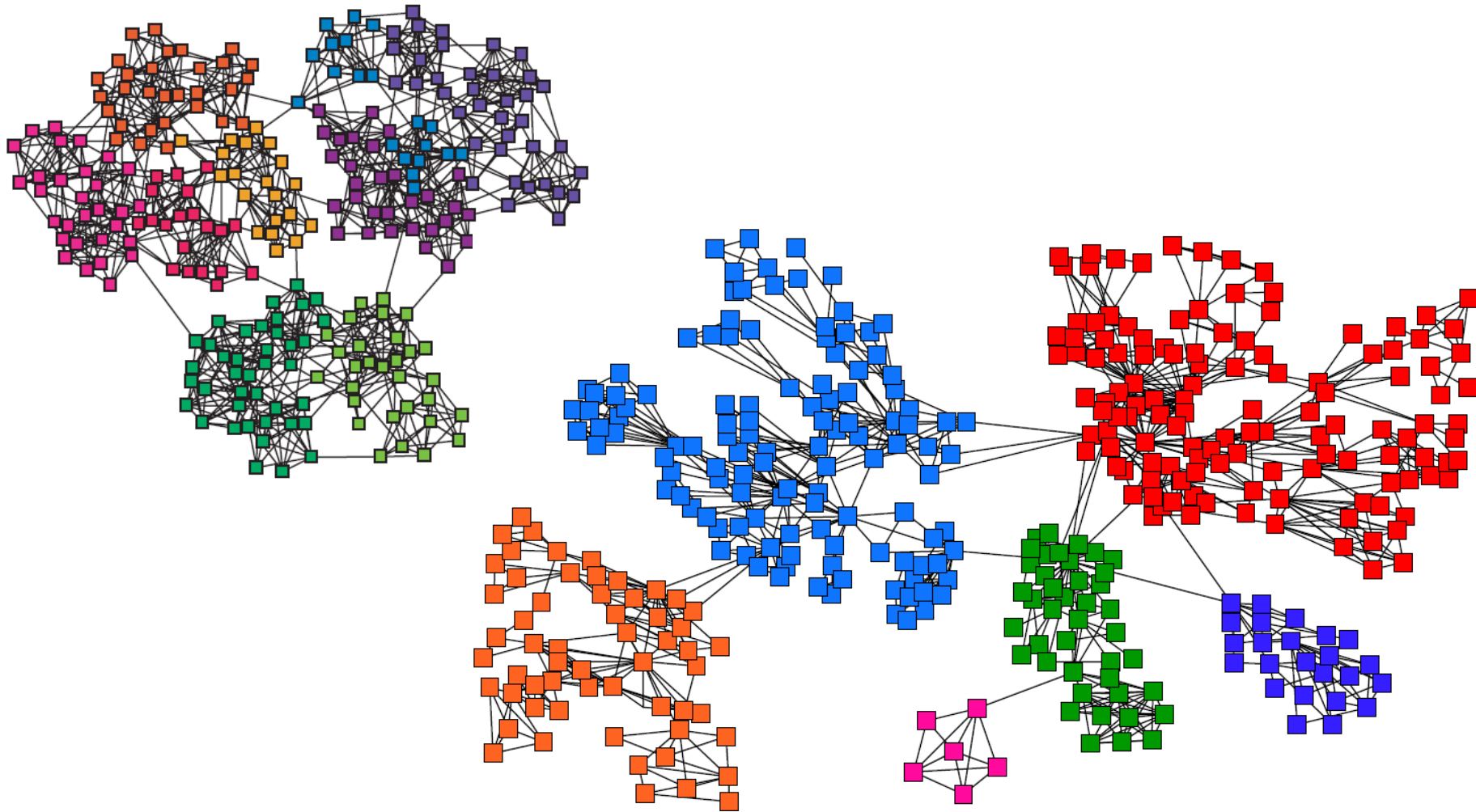
- Hash-Funktion auf Knoten $h:V \rightarrow \{1, \dots, b\}$
- Insgesamt $\binom{b}{3}$ Reducer, die jeweils für eine Menge $\{x, y, z\}$ zuständig sind ($x, y, z \in \{1, \dots, b\}$)
- Mapper: Für jede Kante (u, v) , Mapping auf Schlüssel $\{h(u), h(v), z\}$ für alle $z \in \{1, \dots, b\}$
- Replikationsrate $r = b$
- Jeder Reducer bekommt ca. $\frac{m \cdot b}{\binom{b}{3}} \approx \frac{6m}{b^2}$ Kanten
- Algorithmus mit Heavy Hitters: $O\left(\frac{m^{1.5}}{b^3}\right)$
- Nachteil: Berechnungszeit der Mapper und **Kommunikationskosten**

Inhaltsverzeichnis

- **Einführung**
- **Dreiecke zählen**
- **Community Detection**
 - Girvan-Newman Algorithmus
 - Spectral Clustering

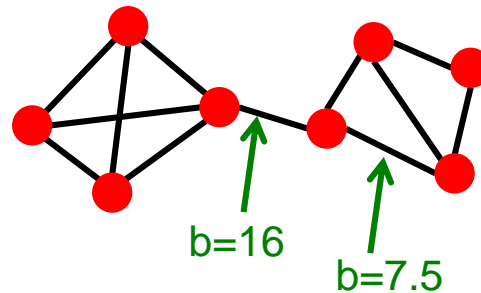
Community Detection

Suche nach Mengen von Knoten mit einer hohen Dichte an Kanten



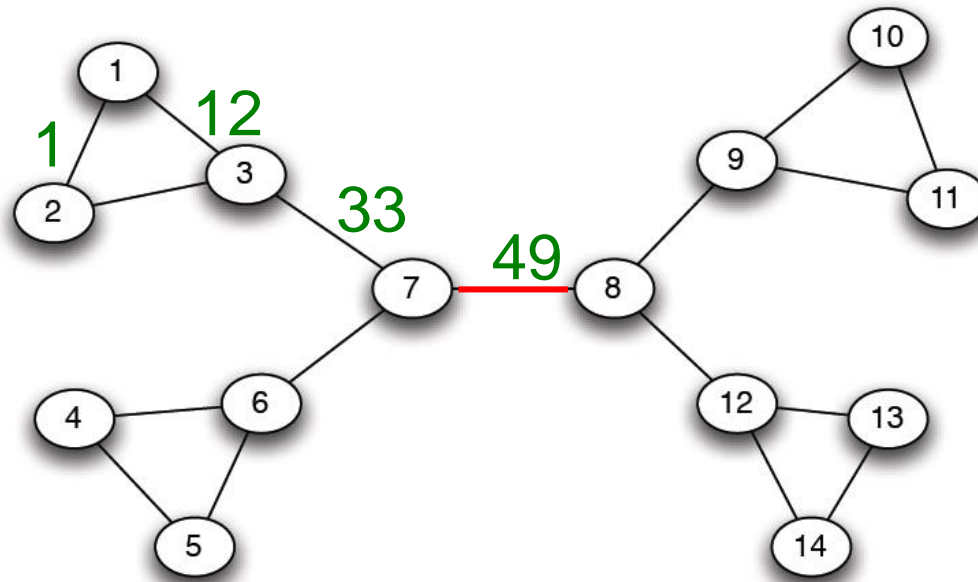
Girvan-Newman Algorithmus

- Edge-Betweenness eine Kante e :
 - Anzahl aller kürzesten Pfade des Graphen, welche e enthalten
 - Bei mehreren (m) kürzesten Pfaden zwischen 2 Knoten wird der Anteil $1/m$ addiert



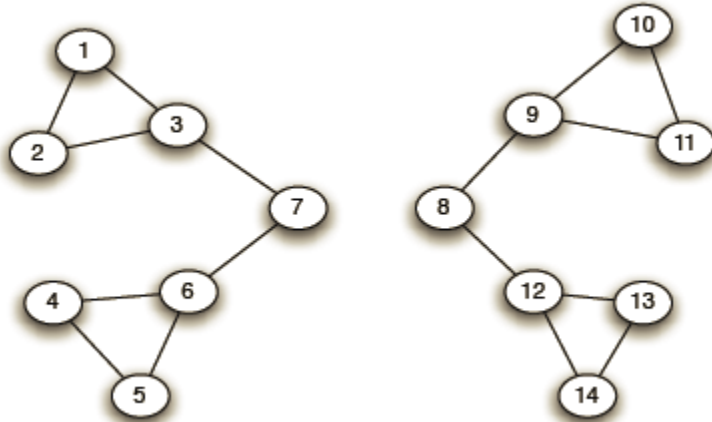
- Girvan-Newman-Algorithmus (ungerichteter Graph):
 - Wiederhole bis gewünschte Anzahl an Cluster erreicht
 - Berechne Edge-Betweenness aller Kanten
 - Entferne Kante mit maximaler Edge-Betweenness
 - Verbundene Komponenten sind Cluster
 - Hierarchische Zerlegung der Graph

Girvan-Newman: Beispiel

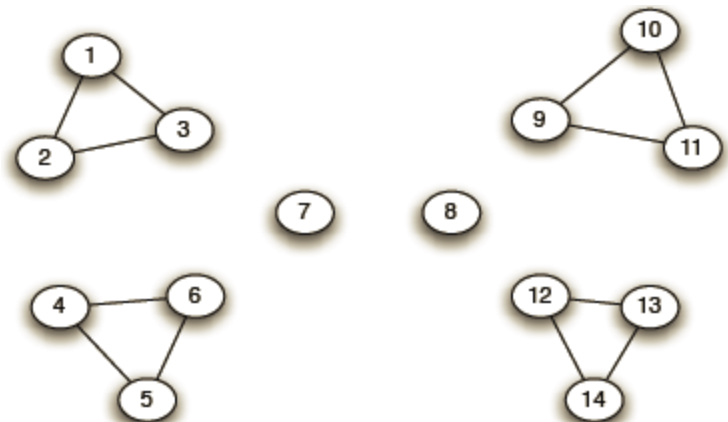


Girvan-Newman: Beispiel

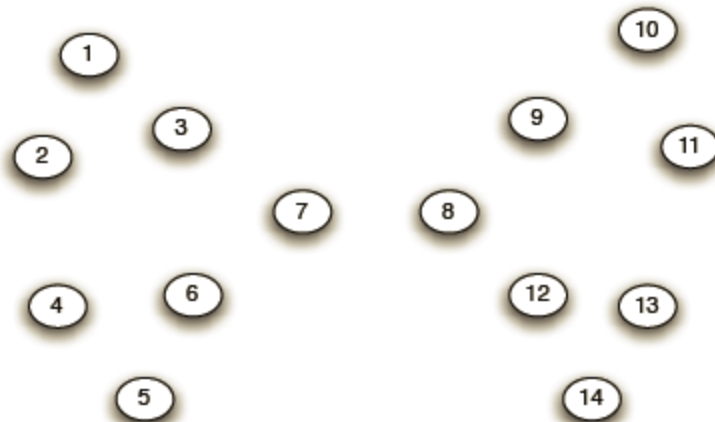
Schritt 1:



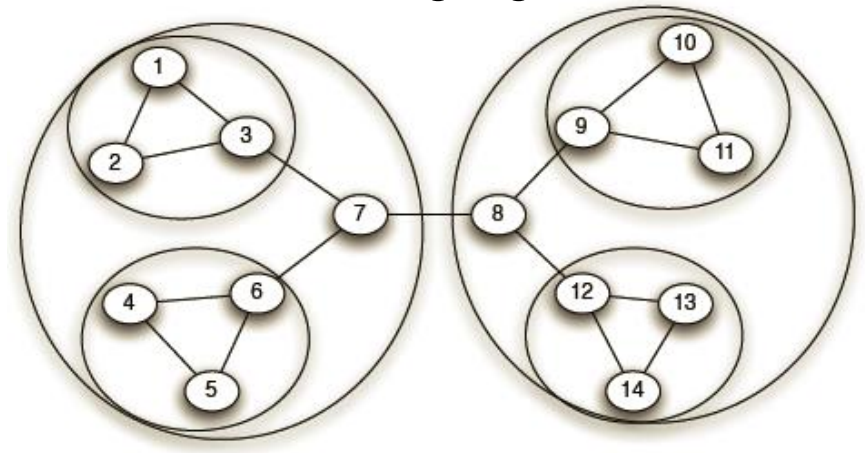
Schritt 2:



Schritt 3:

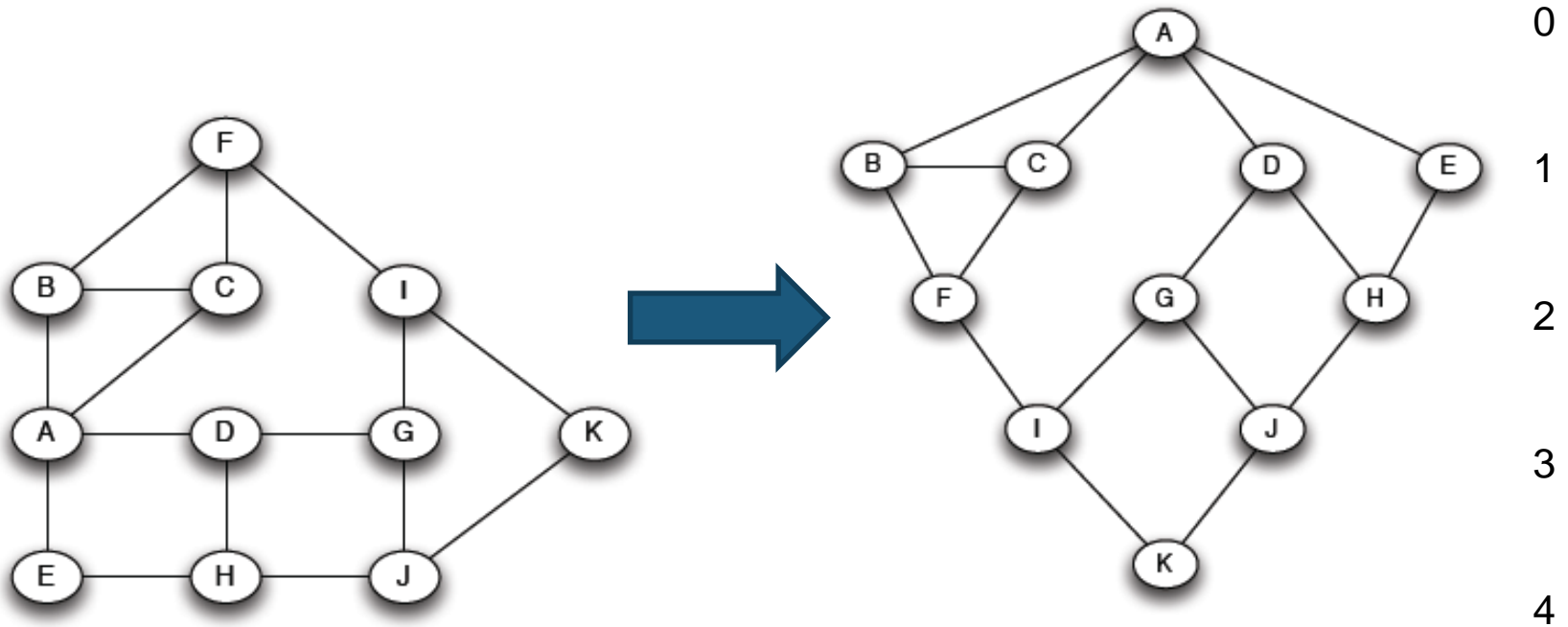


Hierarchische Zerlegung:



Berechnung der Edge-Betweenness

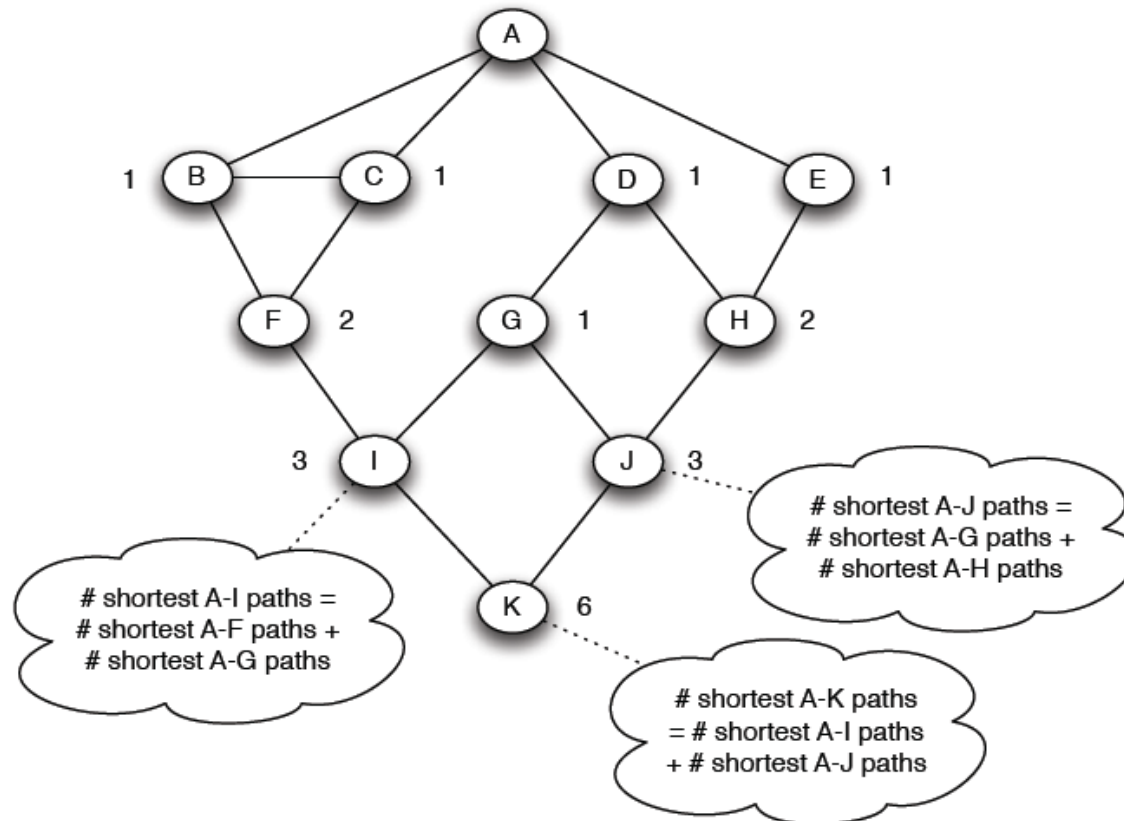
1. Sortierung der Knoten über Breitensuche (Breath-first Search), ausgehend von einem Knoten A



Berechnung der Edge-Betweenness

2. Berechnung der Anzahl der kürzesten Pfade mit Knoten A als Anfang:

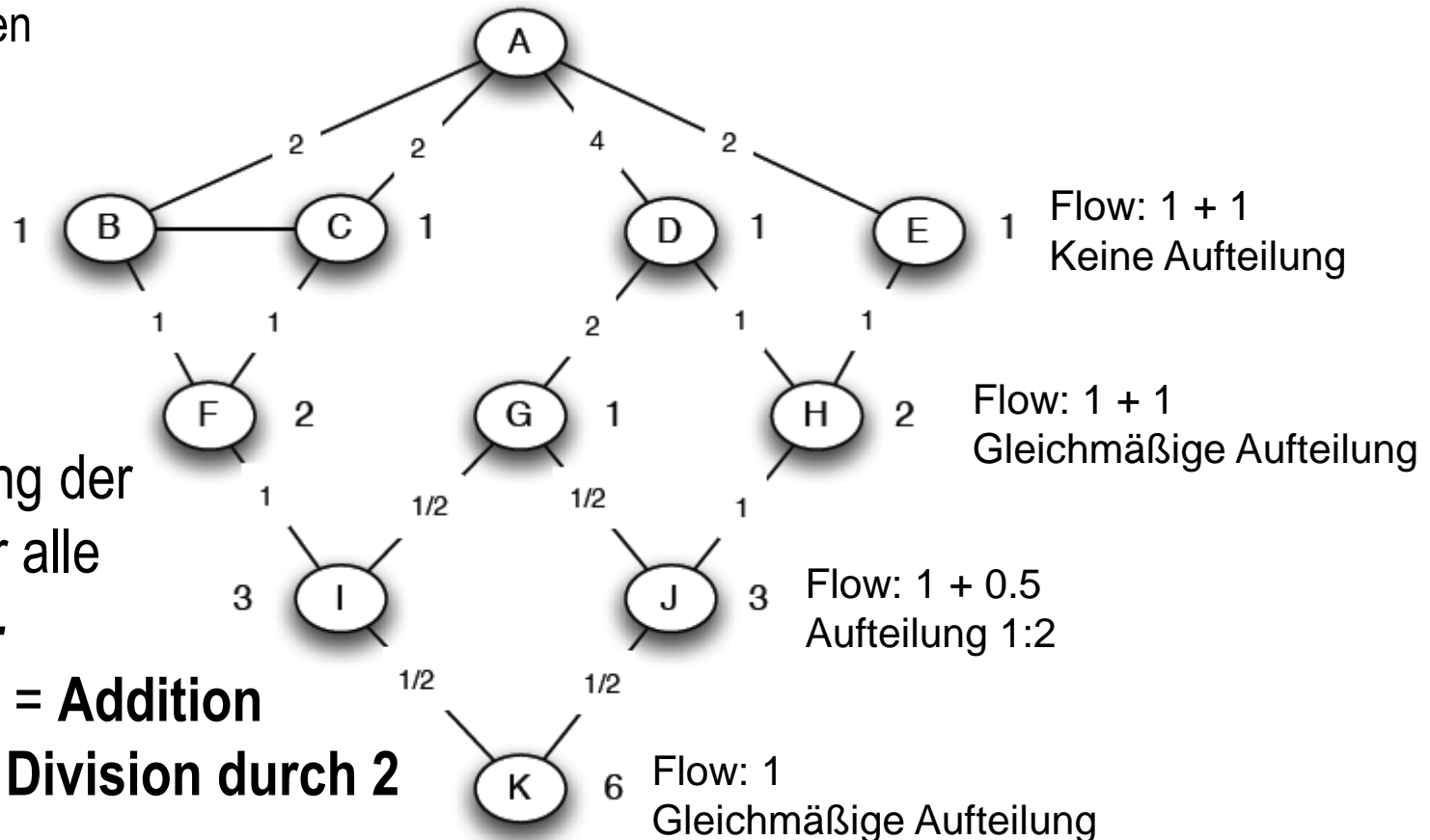
- Wurzelknoten bekommt die Zahl 1 als Beschriftung
- Anzahl der kürzesten Pfade zu einem Knoten = Summe der Beschriftungen der direkten Vorgängerknoten



Berechnung der Edge-Betweenness

3. Berechnung des **Flow** aller Knoten und Kanten

- Beginne auf untersten Stufe (Knoten K)
- Flow eines Knotens: 1 + (Summe des Flow aller nachfolgenden Kanten)
- Aufteilung des Flow über alle vorherigen Kanten, anteilig nach den Beschriftungen der Knoten



4. Wiederholung der Schritte 1-3 für alle Knoten; **Edge-**

Betweenness = Addition des Flow und Division durch 2

Optimale Anzahl an Cluster

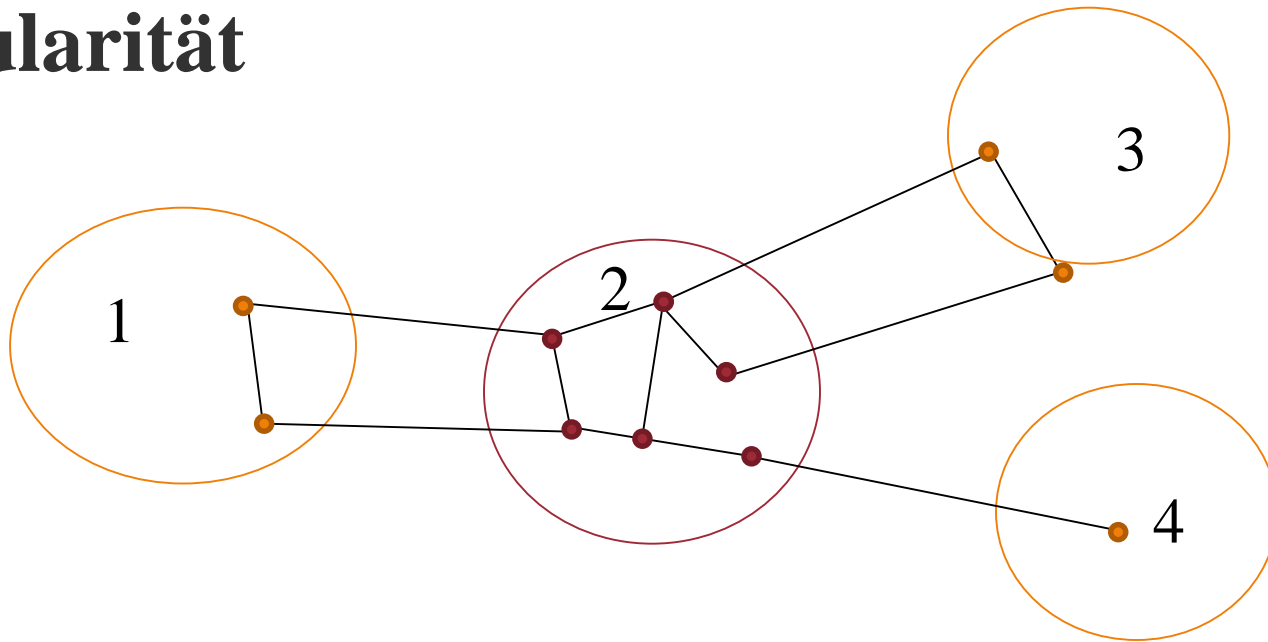
- Cluster = Menge von Knoten mit einer hohen Dichte an Kanten
- Definition: Modularität Q (ungerichteter Graph)

- Maß für die Güte einer Partitionierung eines Graphen in Cluster
- Gegeben einer Partitionierung S :

$$Q(S) = \sum_{\text{Cluster } i \in S} \left[\frac{o_{ii}}{2} - e_i^2 \right]$$

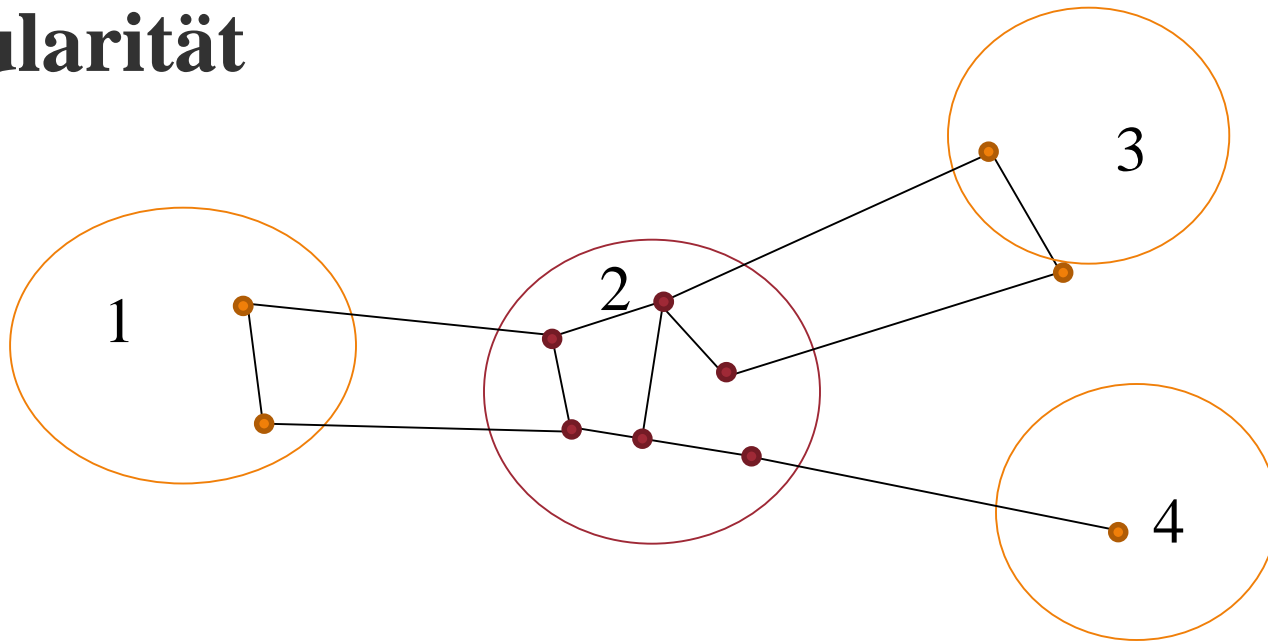
- $a_{vw} = 1$, falls eine Kante zwischen v und w existiert und $a_{vw} = 0$, falls keine Kante existiert; d.h. $a_{vw} = a_{wv}$
- $o_{ij} = \frac{1}{m} \sum_{v \in i, w \in j} a_{vw}$
 - Mit $i \neq j$: o_{ij} ist Anteil der Kanten mit einem Ende in i und dem anderen Ende in j
 - Außerdem: $\frac{o_{ii}}{2}$ ist die Anteil der Kanten innerhalb i
- $e_i = \frac{1}{2} \sum_j o_{ij}$ bezeichnet den Anteil der Kantenenden mit Ursprung in i
- d.h. e_i^2 ist der (in einem Zufallsnetzwerk) erwartete Anteil der Kanten innerhalb i

Modularität



- $o_{12} = o_{21} = \frac{2}{13}, o_{23} = o_{32} = \frac{2}{13}, o_{24} = o_{42} = \frac{1}{13}$
- $o_{11} = \frac{2}{13}, o_{22} = \frac{12}{13}, o_{33} = \frac{2}{13}, o_{44} = 0$
- $e_1 = \frac{4}{26}, e_2 = \frac{17}{26}, e_3 = \frac{4}{26}, e_4 = \frac{1}{26}$
- $Q = \frac{1}{13} - \frac{4}{169} + \frac{6}{13} - \frac{289}{676} + \frac{1}{13} - \frac{4}{169} + 0 - \frac{1}{676} = 0.14$

Modularität



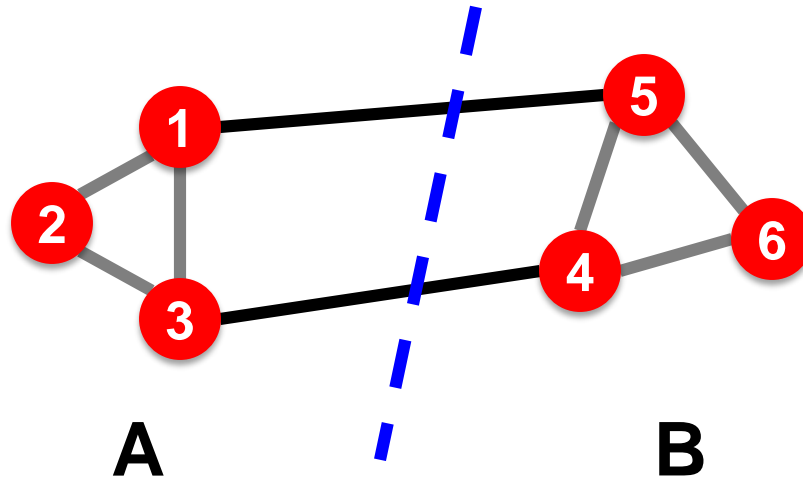
- Q liegt im Intervall $[-1, 1]$
 - $Q > 0$ falls Anzahl der tatsächlichen Kanten innerhalb der Cluster die erwartete Anzahl in einem Zufallsgraph überschreitet
 - Falls $Q > 0.3$ spricht man von einer *Clusterstruktur*
- Bei Girvan-Newman: Verwendung der Modularität um optimale Partitionierung in Cluster zu wählen
- Alternative: Verwendung der Modularität um Partitionierung zu finden (z.B. Louvain Methode)

Inhaltsverzeichnis

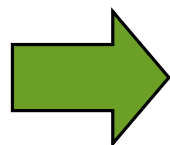
- **Einführung**
- **Dreiecke zählen**
- **Community Detection**
 - Girvan-Newman Algorithmus
 - Spectral Clustering

Graphpartitionierung

- Finden einer Partitionierung mit hoher Modularität
 - Maximiere die Anzahl der Kanten innerhalb der Gruppen
 - Minimiere die Anzahl der Kanten zwischen den Gruppen



- Einfacheres Maß für die Güte einer Partitionierung bei nur **zwei Gruppen**: Anzahl der Kanten zwischen den Gruppen (*Graph Cut*)



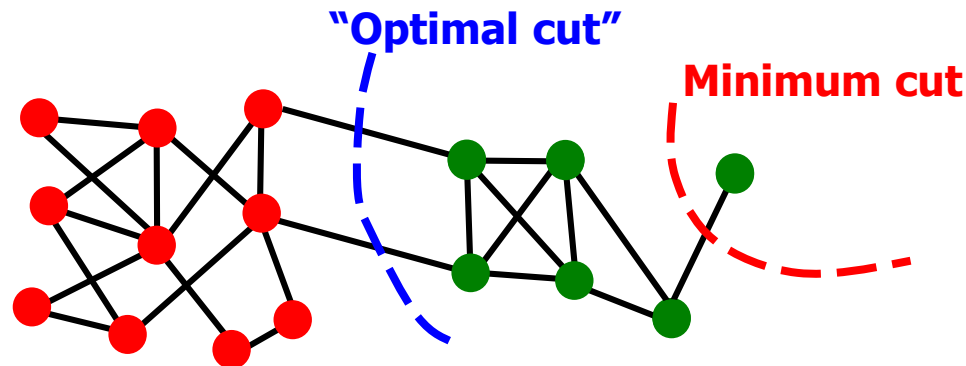
$$cut(A,B) = 2$$

Graph Cut

- Kriterium: *Minimaler Cut*

$$\arg \min_{A,B} \text{cut}(A,B)$$

- **Problem:** Keine Berücksichtigung der Kantendichte innerhalb der Cluster bzw. der Größe der Cluster



- **Alternative: Normalisierter Cut**
$$ncut(A,B) = \frac{\text{cut}(A,B)}{\text{vol}(A)} + \frac{\text{cut}(A,B)}{\text{vol}(B)}$$
 - Verbundenheit zwischen den Clustern relativ zu der Dichte der einzelnen Cluster
 - $\text{vol}(A) = \sum_{i \in A} d_i$ (Summe der Grade aller Knoten des Clusters)

Spectral Graph Partitioning

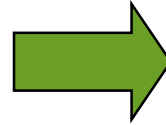
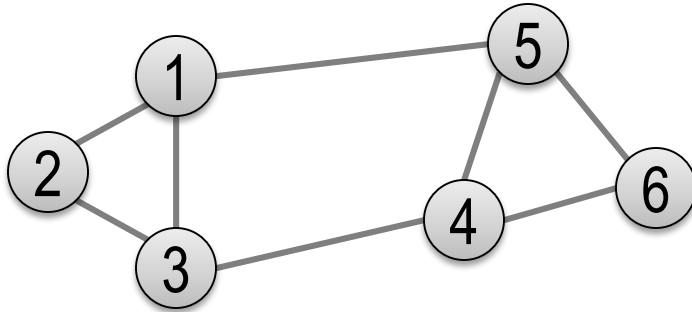
- Wie kann eine Aufteilung mit minimalem normalisierten Cut effizient identifiziert werden?
- Matrixrepräsentation $\mathbf{M}(n \times n)$ eines Graphen, wobei n die Anzahl der Knoten im Graphen
- *Eigenvektoren* $\mathbf{x} = (x_1, \dots, x_n)^T$:

$$\mathbf{M} \cdot \mathbf{x} = \lambda \cdot \mathbf{x}$$

- **Spektrum:** zu den Eigenvektoren \mathbf{x}_i gehörige Eigenwerte λ_i :
 $\Lambda = \{\lambda_1, \dots, \lambda_n\}$ mit $\lambda_1 \leq \dots \leq \lambda_n$
- **Spectral Graph Theory:** Analyse des “Spektrums” der Matrixrepräsentation eines Graphen

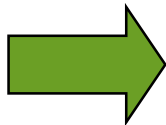
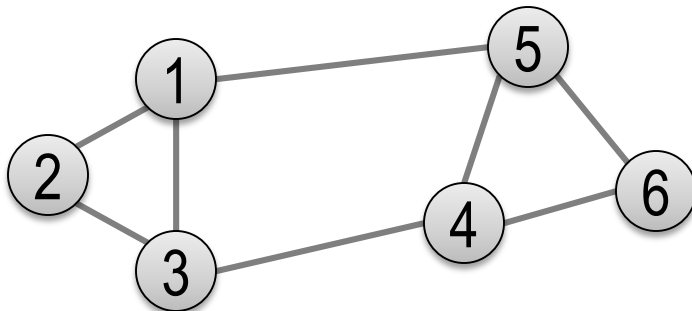
Matrixrepräsentation eines Graphen

- Adjazenzmatrix A



	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

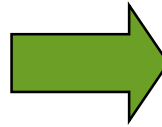
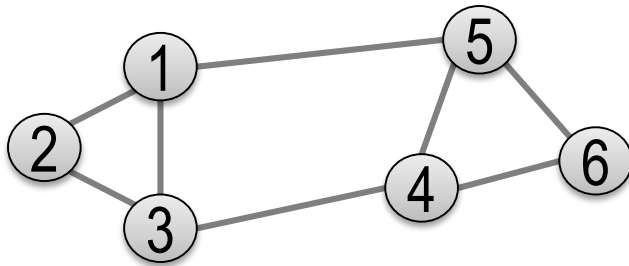
- Gradmatrix D



	1	2	3	4	5	6
1	3	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	3	0	0	0
4	0	0	0	3	0	0
5	0	0	0	0	3	0
6	0	0	0	0	0	2

Matrixrepräsentation eines Graphen

- Laplace-Matrix $L = D - A$



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

- Eigenschaft: Laplace-Matrix ist **positive semi-definite**:

$$x^T L x = x^T D x - x^T A x = \sum_{i=1}^n d_i x_i^2 - \sum_{\{i,j\} \in E} 2x_i x_j = \sum_{\{i,j\} \in E} (x_i - x_j)^2 \geq 0$$

- E ist Menge der Kanten
- D.h. Eigenwerte sind nicht-negative reelle Zahlen

Der zweitkleinste Eigenwert λ_2

- Erster Eigenvektor von L (mit kleinstem Eigenwert): $\mathbf{x} = (\mathbf{1}, \dots, \mathbf{1})^T$:
 $L \cdot \mathbf{x} = \mathbf{0}$ und $\lambda = \lambda_1 = 0$

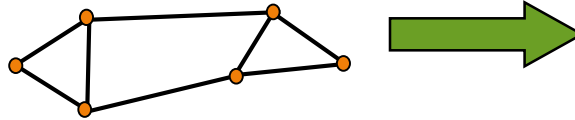
- **Satz:** Unter der Bedingung $\mathbf{x}^T \mathbf{x} = 1$ und \mathbf{x} orthogonal zum ersten Eigenvektor gilt für den zweitkleinsten Eigenwert von L

$$\lambda_2 = \min_{\mathbf{x}} \mathbf{x}^T L \mathbf{x} = \min_{\mathbf{x}} \sum_{\{i,j\} \in E} (x_i - x_j)^2$$

- Der Vektor \mathbf{x} ist der zugehörige Eigenvektor
- Intuition:
 - Minimiere $\mathbf{x}^T L \mathbf{x}$: Wähle x_i und x_j nahe beieinander, falls $\{i, j\} \in E$
 - Da \mathbf{x} orthogonal zu erstem Eigenvektor $(1, \dots, 1)$, muss $\sum_{i=1}^n x_i = 0$ gelten
 - Es müssen also sowohl positive als auch negative Werte in \mathbf{x} vorkommen
- **Idee:** Einteilung der Knoten mit $x_i > 0$ in Cluster X und alle Knoten mit $x_i < 0$ in Cluster Y

Spectral Partitioning

1. Vorbehandlung: Erstellung der Laplace-Matrix



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

2. Aufspaltung

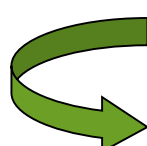
- Berechne Eigenwerte und Eigenvektoren
- Verwende zweiten Eigenvektor um die Knoten zwei Clustern zuzuordnen
 - Einfach: Aufteilung an der 0
 - Komplex: Aufteilung an Schwellenwert, so dass normalisierter Cut minimal

$\lambda =$

0.0
1.0
3.0
3.0
4.0
5.0

$X =$

0.4	0.3	-0.5	-0.2	-0.4	-0.5
0.4	0.6	0.4	-0.4	0.4	0.0
0.4	0.3	0.1	0.6	-0.4	0.5
0.4	-0.3	0.1	0.6	0.4	-0.5
0.4	-0.3	-0.5	-0.2	0.4	0.5
0.4	-0.6	0.4	-0.4	-0.4	0.0

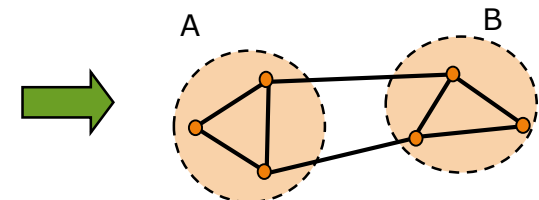


1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6

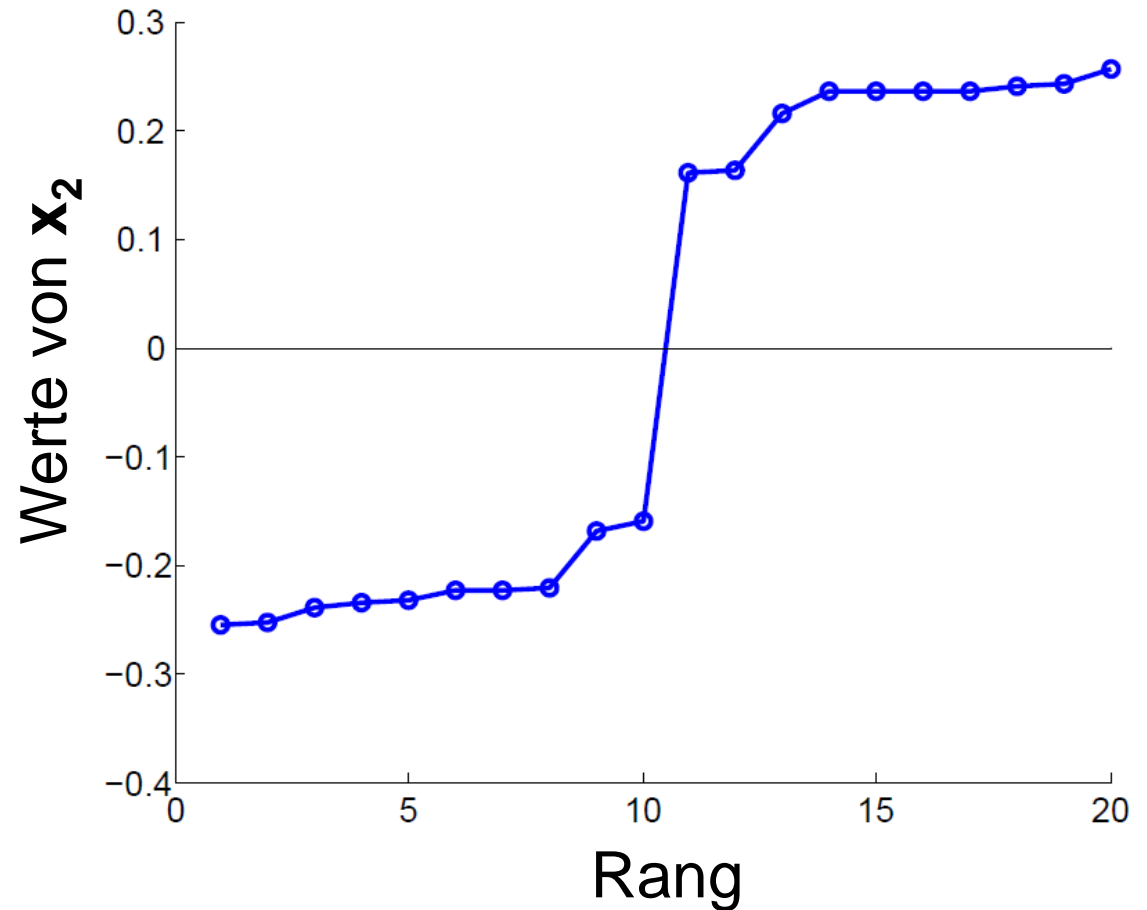
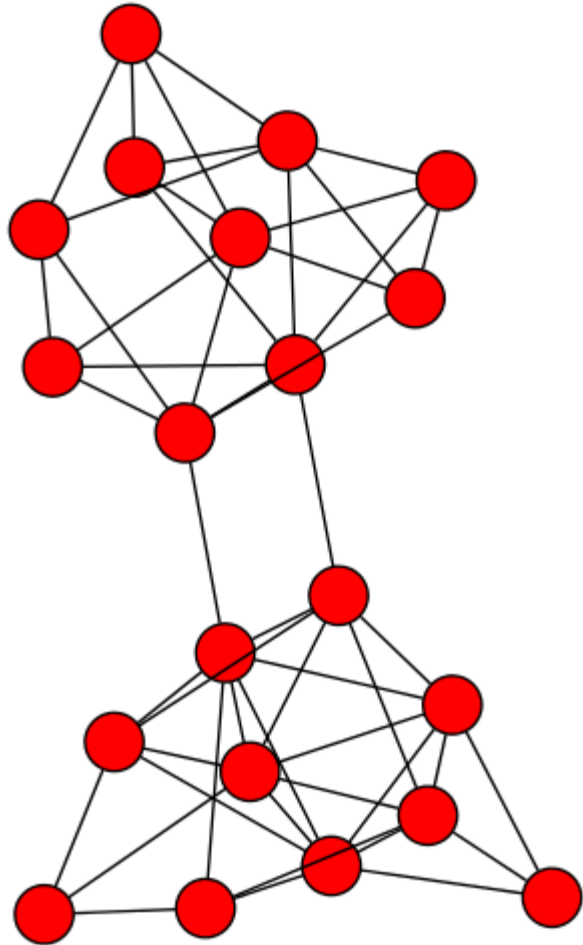
Aufteilung an der 0

1	0.3
2	0.6
3	0.3

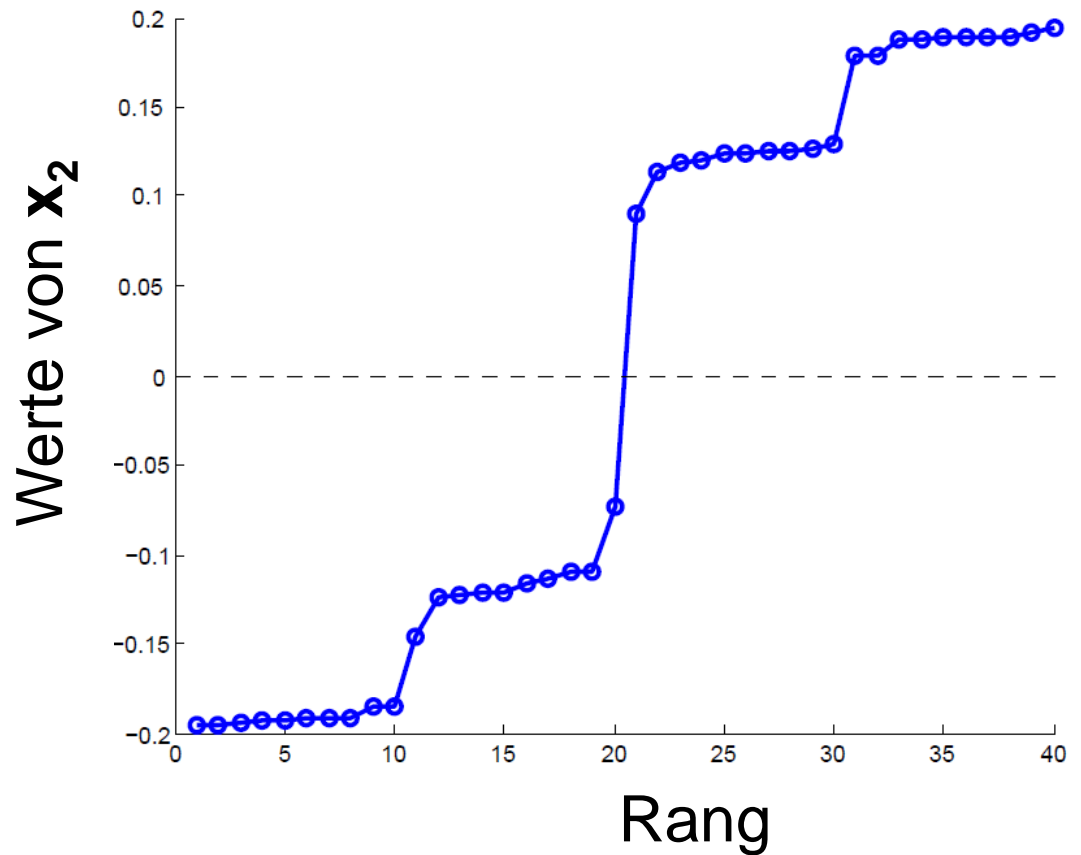
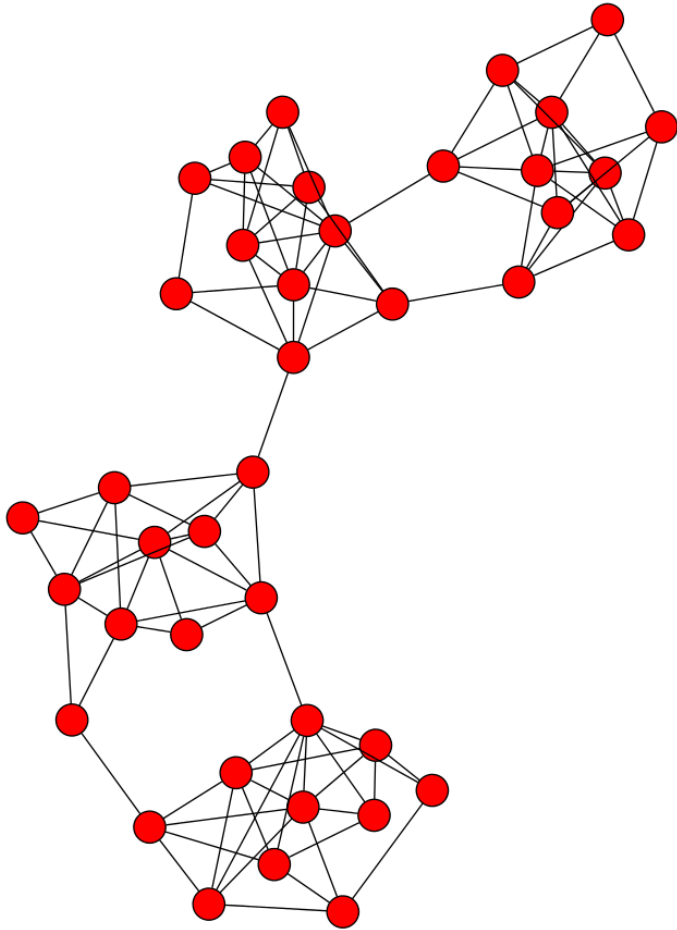
4	-0.3
5	-0.3
6	-0.6



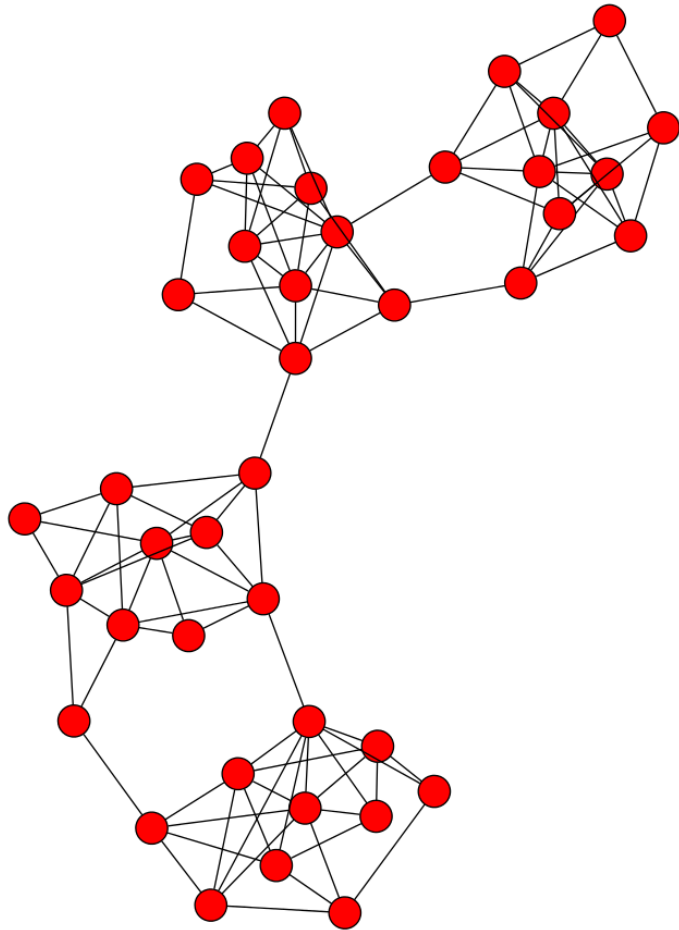
Beispiel: Spectral Partitioning



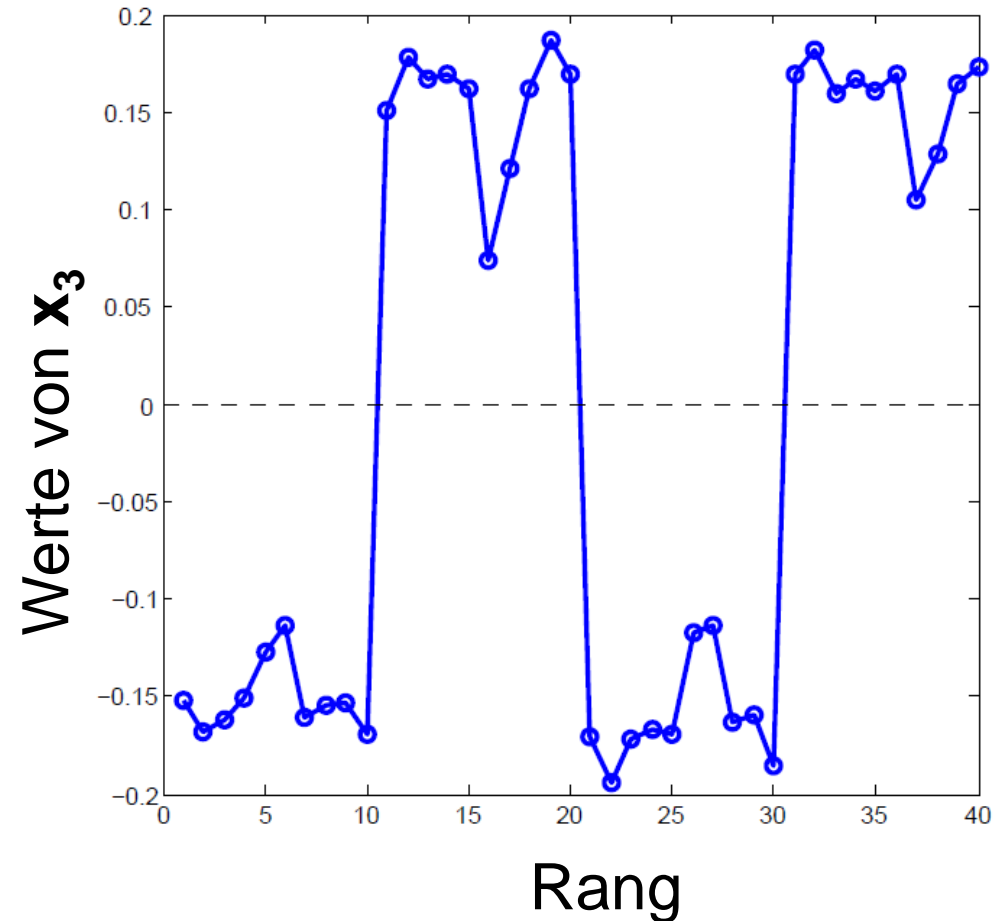
Beispiel: Spectral Partitioning



Beispiel: Spectral Partitioning



Zusätzliche Verwendung von x_3



Kombination mehrerer Eigenvektoren um Graph in mehr als zwei Cluster zu partitionieren bzw. den normalisierten Cut zu minimieren