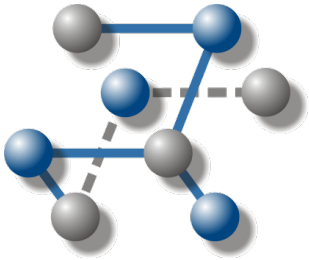
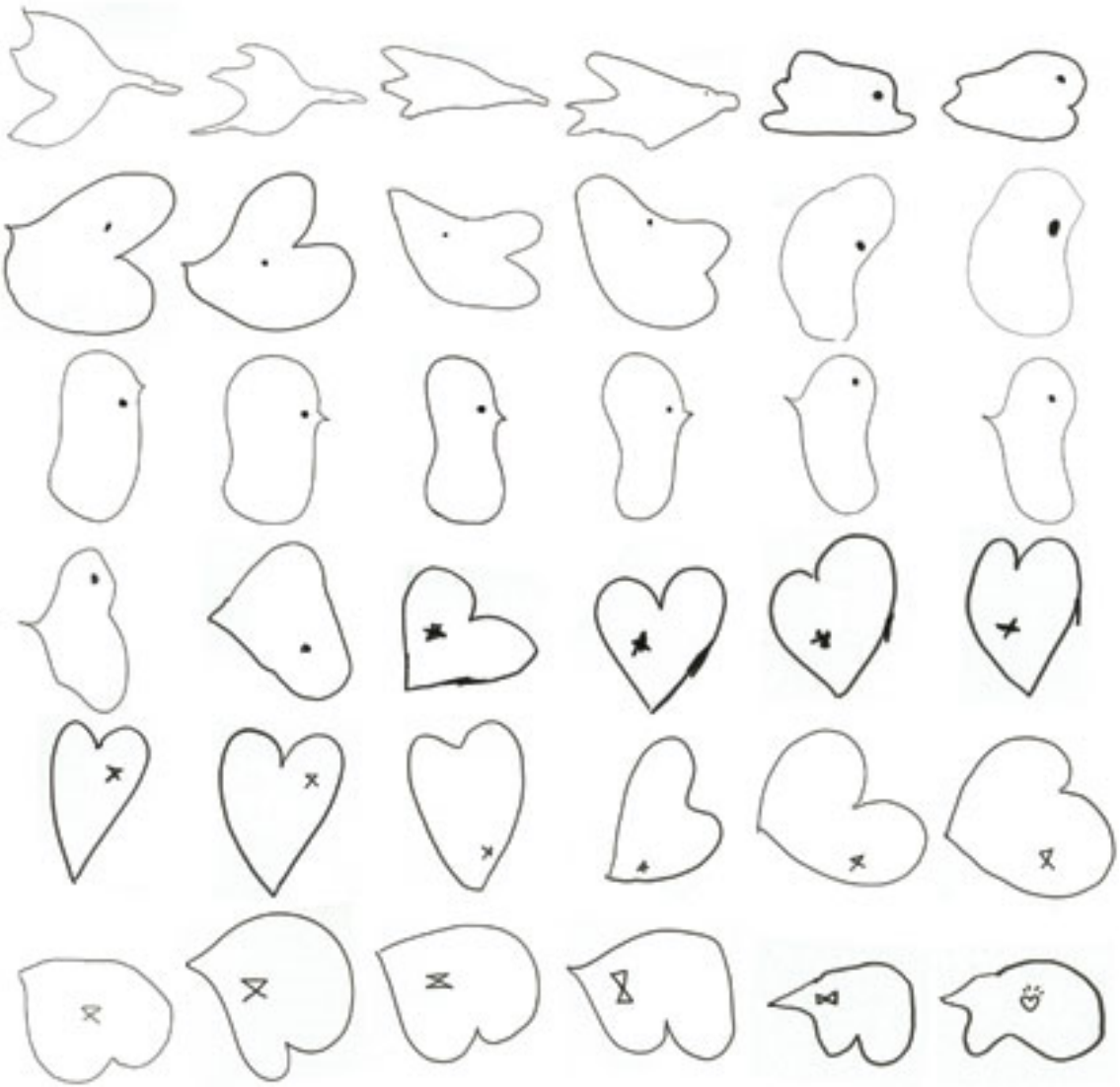
Chinese Whispers



Chinese Whispers



Gliederung

- Grundidee
- Algorithmus
- Parallelisierung
- CW = optimiertes MCL?
- Analyse verschiedener Graphentypen
- Probleme

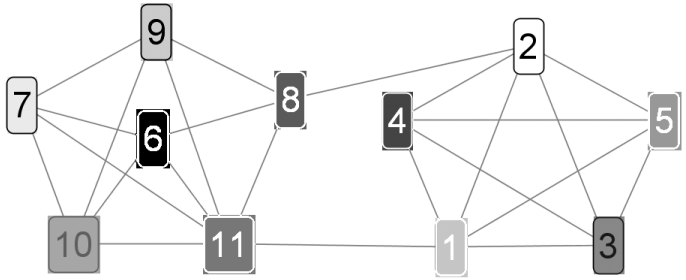
Grundidee – Abstraktion ist das Zauberwort

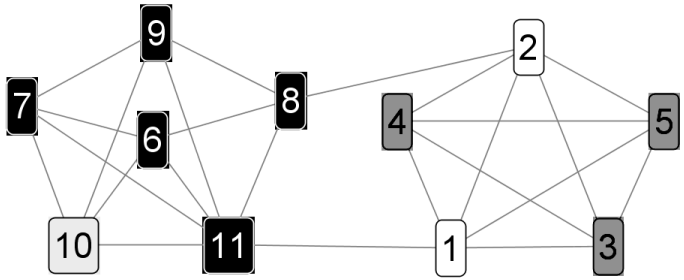
- Wir haben eine Menge von Menschen (Knoten) und alle haben eine eigene Meinung (Klasse)
- Jeder Mensch hat eine Menge von anderen Menschen, denen er vertraut und mit denen er sich austauscht (Kanten)
- Die Meinung eines Menschen verändert sich, abhängig von der vorherrschenden Meinung in seiner Umgebung
- Ergebnis: Irgendwann haben wir Cluster, weil wir Gruppen mit in sich gefestigten Meinungen haben, die aber voneinander abgeschottet sind

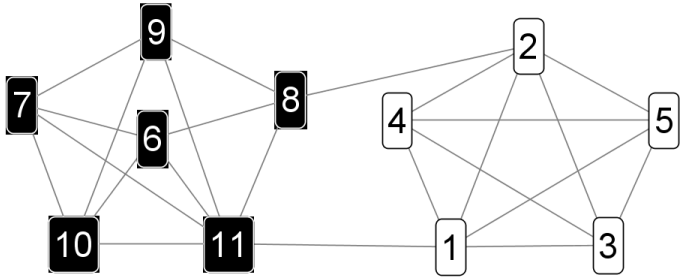
Ein fast-Algorithmus

```
initialize:
 forall vi in V: class(vi)=i;
while changes:
 forall v in V, randomized order:
  class(v)=highest ranked class
             in neighborhood of v;
```

Beispiel



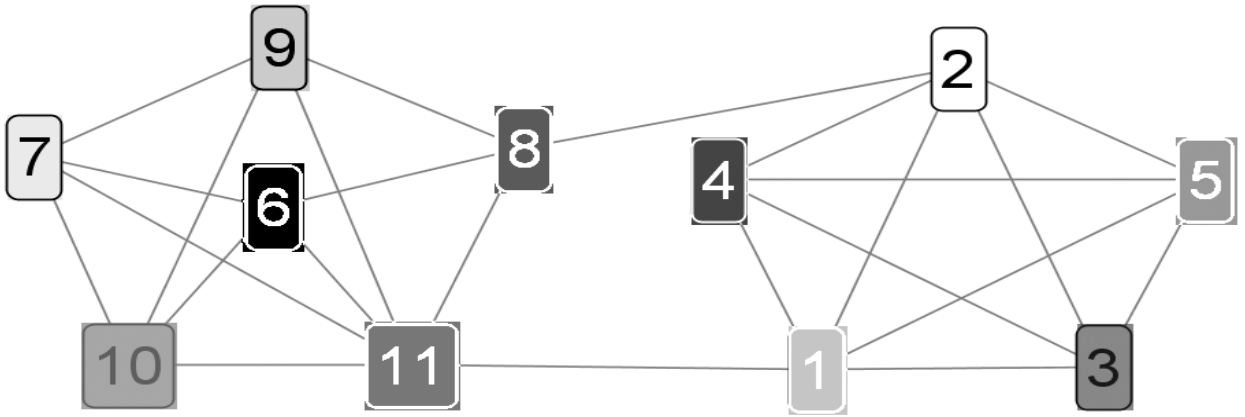




Algorithmus II

```
initialize:
  forall vi in V: class(vi)=i;
```

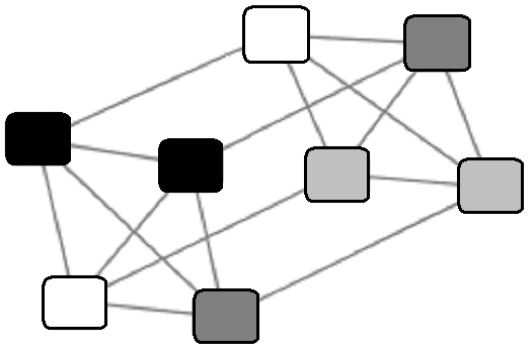
- Jeder Knoten bekommt eine Klasse zugewiesen.
- Komplexität O(n)

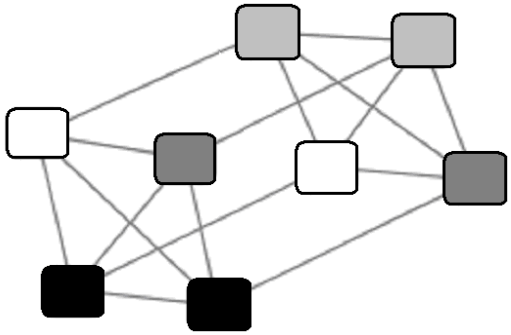


Algorithmus III

```
while changes:
```

- Im Optimalfall konvergiert das Clustering
- für den Normalfall benötigt man aber eine Abbruchbedingung bzw. feste Anzahl an Läufen
- Komplexität O(?)





Algorithmus IV

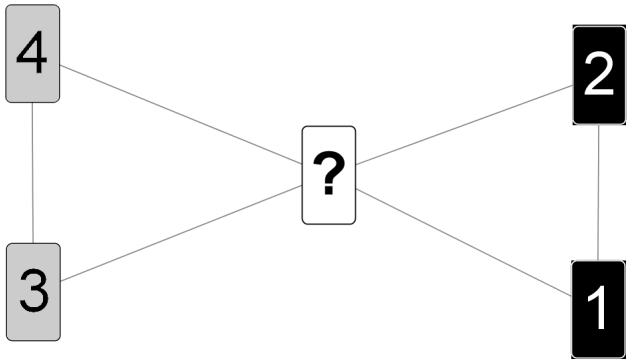
```
forall v in V, randomized order:
```

- Durch die zufällige Reihenfolge der Knoten, erhalten wir einen Nicht-Deterministischen Algorithmus
- Komplexität O(n)

Algorithmus V

```
class(v)=highest ranked class
                in neighborhood of v;
```

- Sollte es mehrere stärkste Klassen geben, wird zufällig eine ausgewählt
  - Nicht-Deterministisch II
- Komplexität O(n)



Algorithmus VI

- Theoretische Zeitkomplexität: $O(\infty)$
- Praktische Zeitkomplexität: $O(|E|)$
- Die Zeitkomplexität des Algorithmus wird als linear zur Kantenanzahl angegeben
- Speicherkomplexität: $O(|E|+|V|) = O(n^2)$
- Der Algorithmus ist nicht deterministisch (zufällige Zuweisung der Klassen bei Gleichstand)
- Funktioniert auch bei gewichteten Graphen

Parallelisierbarkeit

- Parallelisierung der Klassenzuweisung

```
forall v in V, do parallel:
  class(v)=highest ranked class
                in neighborhood of v;
```

- durch die zu erwartende spätere Aktualisierung der Klassen, benötigt der Algorithmus mehr Iterationen um ein „stabiles“ Clustering zu erhalten
- Jeder Prozess benötigt globales Wissen
- Oszillationsproblem

CW Spezialfall von MCL

maxrow(D)= Setze zeilenweise den höchsten Wert der Zeile auf 1, den Rest auf 0

$D^0 = I_n$

for t=1 to iterations

$D^{t-1} = maxrow(D^{t-1})$ //Inflation

$D^t = D^{t-1}A_G$ //Expansion

Fehler im Expansionsschritt?

- D soll eigentlich die Stärke der Cluster in Bezug auf die einzelnen Knoten darstellen, daher: eine Zeile entspricht einem Knoten und die Werte der Verteilung der Cluster in der Umgebung
- Aber nach $\texttt{maxrow}(D^{t-1})A_G$ steht in d_{ij} der Wert der Kante von Knoten i nach Knoten x, mit x = das Cluster, welches von maxrow auf der i_{ten} Zeile ausgewählt wurde.

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

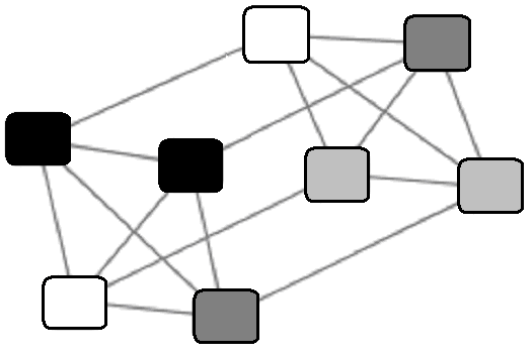
Fehler im Expansionsschritt? II

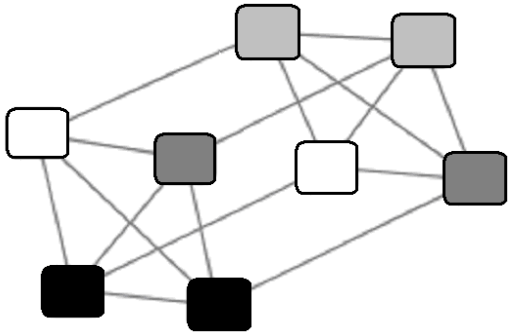
- Vertauscht man aber die Glieder, entspricht d_{ij} der Summe der Kantengewichte der Kanten von j nach i, mit j in Cluster i
- Alternative: maxrow wird zu maxcolumn

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 2 \\ 0 & 0 & 2 \\ 0 & 0 & 2 \end{pmatrix}$$

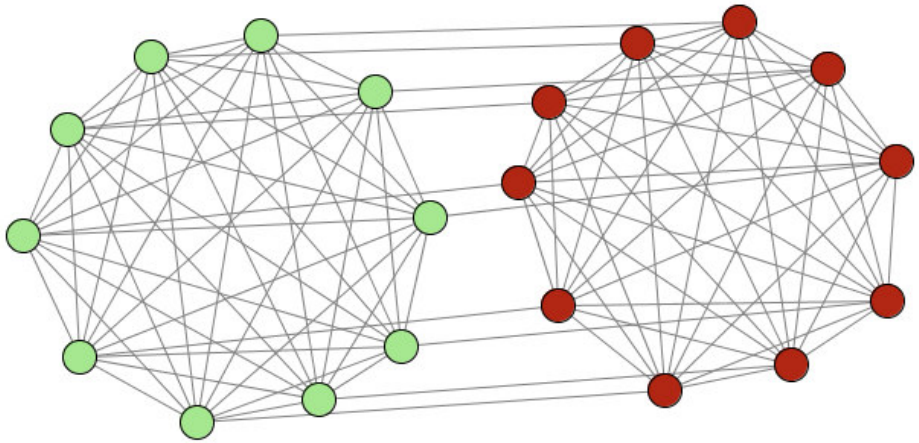
Osszilation

- Aufgrund der Natur von Matrizenmultiplikation werden die Klassen der Knoten nicht kontinuierlich (wie es der Algorithmus eigentlich beschreibt) sondern Iterationsschrittweise gesetzt
- Dies führt zu Oszillation
- Gegenmaßnahmen:
  - Zufällige Mutationen im maxrow Schritt
  - Oder beibehalten der Klasse von einem Schritt zum nächsten mit einer bestimmten Wahrscheinlichkeit.



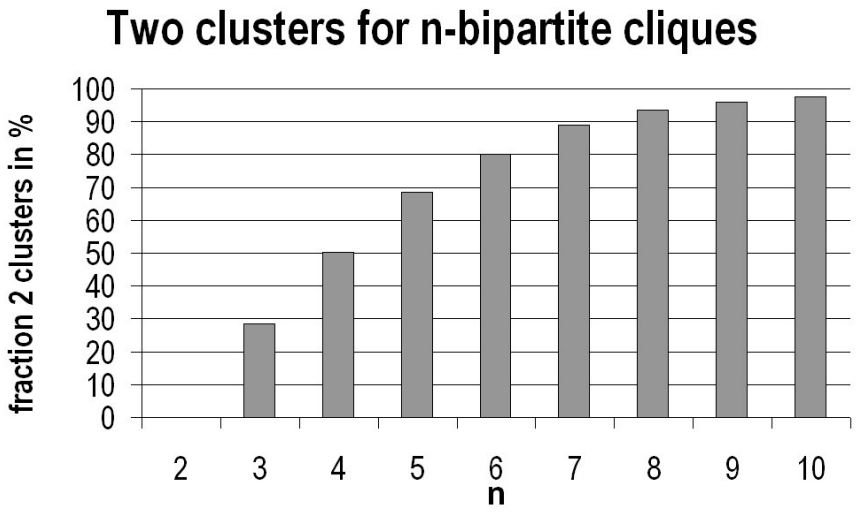


Bi-partite cliques



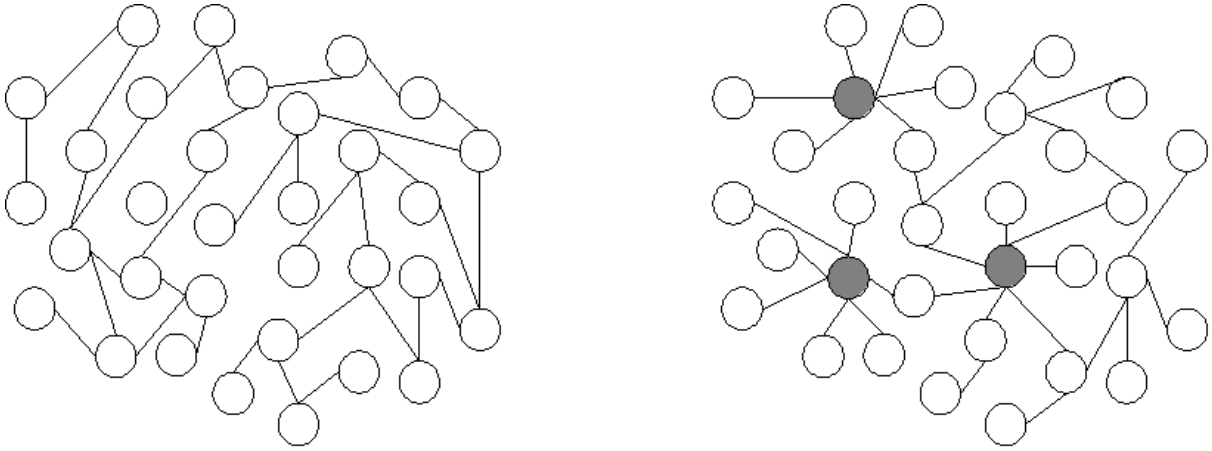
- 2 gleich große Teilgraphen
- Innerhalb der Teilgraphen existiert eine Kante von jedem Knoten zu jedem anderen Knoten (Clique)
- Jeder Knoten hat eine Kante, die in den anderen Teilgraphen führt

Bi-partite cliques II



- Problem: in einem der frühen Iterationsschritte, kann eine Klasse von einem der Teilgraphen „überspringen“ und der Algorithmus clustert diesen Graphen als einen großen Cluster
- Tritt vor allem bei kleinen Graphen auf

Scale Free Network



(a) Random network (b) Scale-free network

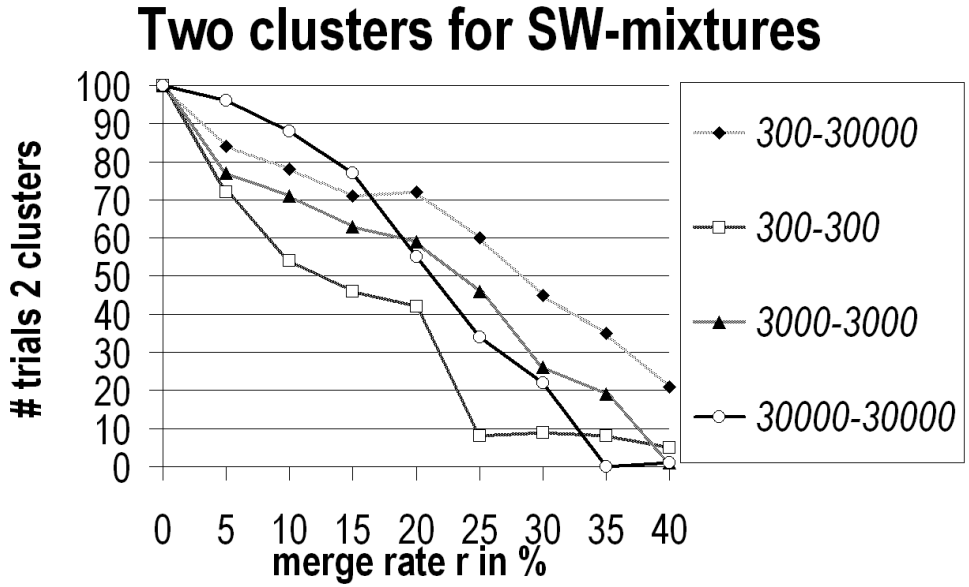
- Geringer Durchmesser (die Länge des Kantenzugs zweier beliebiger Knoten ist kurz)
- Hubs (siehe graue Knoten rechtes Bild)
- Beispiel: Small World Graph (im folgenden SW-Graph)

Scale Free Network II

- Man erzeuge 2 unterschiedliche Scale Free Graphs und füge Sie zu einem Graphen zusammen
- Dabei wird eine bestimmte Menge von Knoten (merge rate r) des kleineren Graphen, mit je einem Knoten des größeren Graphen zusammengeführt.
- Wenn man diesen neuen Graphen clustert, erwartet man, dass der Clustering-Algorithmus die Originalgraphen als einzelne Cluster heraushebt
- Dies wiederholt man oft und für unterschiedliche merge rates und bewertet das Clustering Ergebnis

Scale Free Network III

- Klar: größeres r => weniger korrekte Cluster
- Ansonsten scheint das Verhältnis der Graphengrößen keinen ausreichend großen Einfluss auf das Ergebnis zu haben
- Bis zu einer Merge Rate von 20% kann man über mehrere Läufe und dem Mitteln der Ergebnisse zu einem Erfolg kommen



•Mutual Information

- MI ist die Menge an Informationen die zwei Zufallsgrößen (X,Y) gemein haben
- $nMI = \frac{H(X)+H(Y)-H(X,Y)}{max(H(X),H(Y))}$
- I(X;Y) = H(X) - H(X|Y)
- H(X) ist die Entropie von X
  - $H(X) := -\sum_{x \in M} P(X = x) \log_b P(X = x)$
  - b meist 2 oder e
- H(X|Y) ist die Entropie von X unter Annahme von Y $H(X|Y) := \sum_{y \in L : P(Y=y) > 0} P(Y = y) H(X|Y = y)$
  -
- Übertragen auf Cluster: je höher der MI desto weniger Abhängig sind die beiden Cluster (desto weniger bedingen sich Änderungen)

Scale Free Network IV

- 1K = 1000 Knoten SW Graph, 10K = 10000 Knoten SW Graph
- 7ling co-occurence graph aus NLP (ASV), 22805 Knoten und 232875 gewichtete Kanten
- Wert ist der Mutual Information Wert in %
- Erkenntnis: bereits nach wenigen Iterationen, stabilisiert sich der MI-Wert

Iterations	1	2	3	5	10	20	30	40	49
1K	1	8	13	20	37	58	90	90	91
10K	6	27	46	64	79	90	93	95	96
7ling	29	66	90	97	99,5	99,5	99,5	99,5	99,5

Problem I - Merge

- Merge eines ungeclusterten Graphen in einen geclusterten Graphen:
  - Ungeclusterte Knoten gehen vor allem bei ungewichteten Kanten in bereits existierende Cluster auf
  - Wenn man einen bereits geclusterten Graphen hat, sollte man die anderen Graphen ebenfalls Clustern und dann erst Mergen

Problem II - Softpartitioning

- Kein Soft-Partitioning
  - Normalerweise gehört ein Knoten nur zu einem Cluster.
  - Für bestimmte Anwendungsbereiche (z.B. Language Separation) kann es aber sinnvoll sein, einem Knoten mehrere Cluster zuzuweisen
- In CW realisierbar, indem die Clusterklassen gewichtet verteilt werden (d.h. ein Knoten hat weiterhin nur einen Cluster, man merkt sich aber, wie stark andere Cluster in der Umgebung vertreten sind)

Problem III - Granularität

- Keinerlei Einfluss auf die Granularität
  - Granularität jedes Clusters zufällig
  - Innerhalb eines Graphen können nahezu äquivalente Teilgraphen zu unterschiedlichen Clusterbildern führen
- **Möglicher** Ausweg mehrere Läufe und paarweises Merken der Zugehörigkeit zu einem Cluster.
  - Setzen eines Schwellenwertes und Verwerfen aller Werte unter diesem.
  - Bilden von Clustern abhängig von den paarweisen Zusammenhängen
  - Je höher der Schwellenwert desto kleiner die Cluster

Problem IV - Hubs

- Knoten mit einer hohen Anzahl an Kanten (Hubs) beschleunigen zwar die Ausbreitung einer Klasse, sind oft aber auch das Einfallstor für ungewollte Klassenverteilungen
- 2 Ansätze
  - Löschen der Hubs
  - Abschwächen der Kantengewichte, indem man den Vernetzungsgrad der Knoten berücksichtigt
  - Nebenwirkung: wenn die Hubs weniger Bedeutung haben, werden die Cluster kleiner

Quellen

- http://wortschatz.uni-leipzig.de/~cbiemann/
  - Biemann, C. (2007): Unsupervised and Knowledge-Free Natural Language Processing in the Structure Discovery Paradigm.
  - Biemann, C. (2006): Chinese Whispers - an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. Proceedings of the HLT-NAACL-06 Workshop on Textgraphs-06, New York, USA
- http://www.scholarpedia.org/article/Mutual_information