A Fast Algorithm to Find Overlapping Communities in Networks

Steve Gregory

Proceedings of the 12th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2008)

Bianca Selzam, 14.7.2009

Motivation

- Netzwerke können zur Repräsentation komplexer Systeme verwendet werden
- Treten innerhalb einer Menge von Knoten mehr Kanten untereinander auf als zu anderen Knoten außerhalb, so formen diese Knoten eine Community
- Knoten innerhalb einer Community stehen oftmals miteinander in Beziehung
- Aber:
  - keine einheitliche Definition des Begriffs „Community“
  - keine alleingültigen Regeln zur Aufteilung eines Netzwerks in Communities
- Community soll als Subgraph verstanden werden, dessen interne Kanten größere Dichte haben als die externen Kanten

Motivation und Ziele

- Existierende Algorithmen partitionieren Netzwerke meistens in disjunkte Teilmengen (Cluster) von Knoten ohne tiefere Hierarchieebenen
- Aber:
  - Struktur der Communities nicht immer auf einer einzigen Ebene abbildbar
  - Communities sind nicht immer disjunkt
- Ziele:
  - Erkennen überlappender Communities
  - Verbesserung des bereits existierenden, aber langsamen CONGA Algorithmus

Der CONGA Algorithmus (2007)

- CONGA = **C**luster-**O**verlap **N**ewman **G**irvan **A**lgorithm
- Erweiterung des GN-Algorithmus von Girvan und Newman (2002)
- Idee: Kanten mit hoher Dichte verbinden Cluster, d. h. sie sind externe Kanten
- Eingabe-Netzwerk mit n Knoten wird in seiner Gesamtheit als Cluster betrachtet
- Cluster werden so lange in zwei Teile zerlegt, bis jedes Cluster nur noch einen Knoten enthält
- anschließende Rekonstruktion einer Partition mit der gewünschten Clustergröße

Betweenness

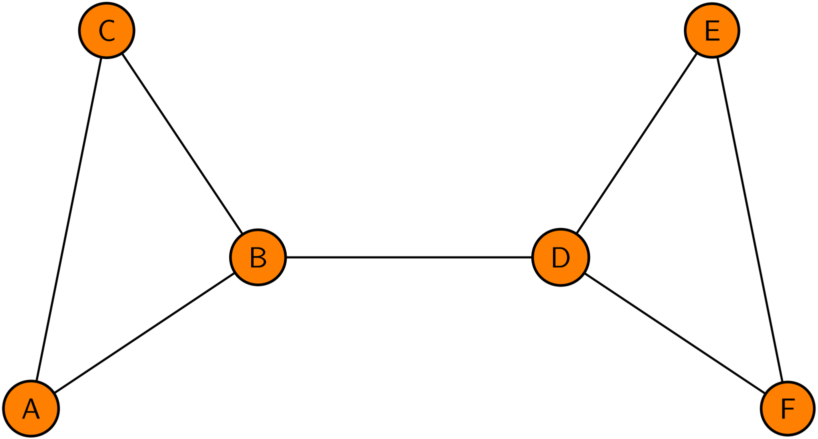
Definition (edge betweenness)

Die **edge betweenness** einer Kante e ist die Anzahl der kürzesten Pfade zwischen allen Knotenpaaren (v_1, v_2), die e benutzen.

Definition (split betweenness)

Die **split betweenness** eines Knoten v ist die Anzahl der kürzesten Pfade, die durch den fiktiven Pfad $\{v_1, v_2\}$ laufen würden, falls v in die beiden Teile v_1 und v_2 zerlegt werden würde.

Beispiel: Berechnung der edge betweenness



Beispiel: Berechnung der edge betweenness

Bestimmung der kürzesten Pfade zwischen allen Knoten:

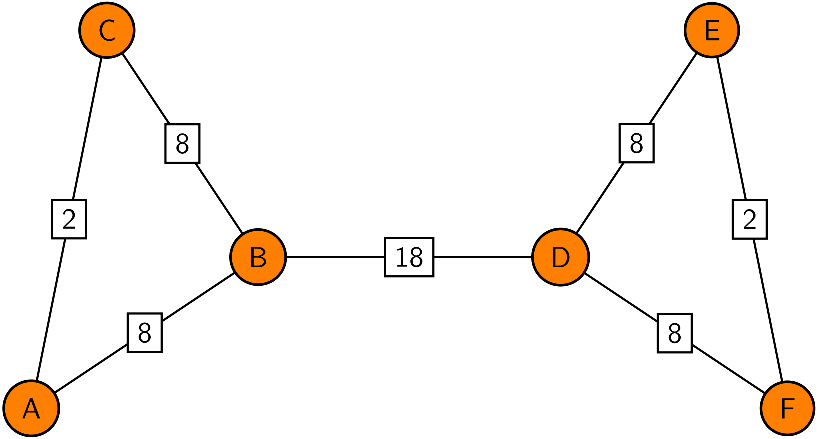
v_1	v_2	**kürzester Pfad**	v_1	v_2	**kürzester Pfad**
A	B	$\{A,B\}$	B	F	$\{B,D\}, \{D,F\}$
A	C	$\{A,C\}$	C	D	$\{C,B\}, \{B,D\}$
A	D	$\{A,B\}, \{B,D\}$	C	E	$\{C,B\}, \{B,D\}, \{D,E\}$
A	E	$\{A,B\}, \{B,D\}, \{D,E\}$	C	F	$\{C,B\}, \{B,D\}, \{D,F\}$
A	F	$\{A,B\}, \{B,D\}, \{D,F\}$	D	E	$\{D,E\}$
B	C	$\{B,C\}$	D	F	$\{D,F\}$
B	D	$\{B,D\}$	E	F	$\{E,F\}$
B	E	$\{B,D\}, \{D,E\}$			

Beispiel: Berechnung der edge betweenness

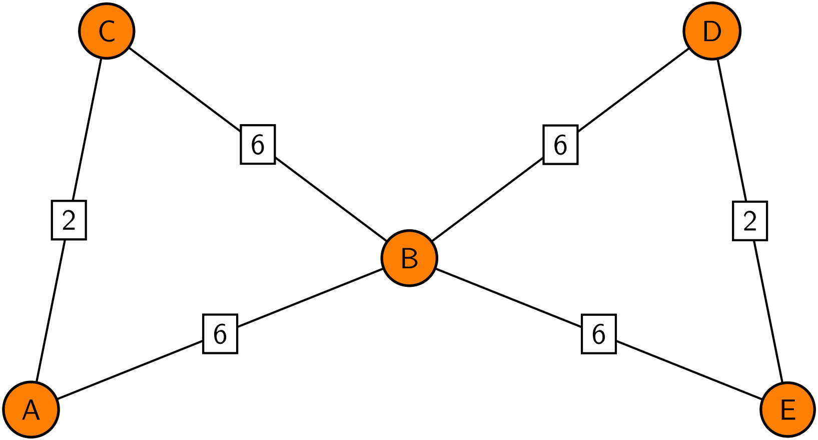
Bestimmung der edge betweenness für alle Kanten:

Kante	**Anzahl**
$\{A,B\}$ bzw. $\{B,A\}$	8
$\{A,C\}$ bzw. $\{C,A\}$	2
$\{B,C\}$ bzw. $\{C,B\}$	8
$\{B,D\}$ bzw. $\{D,B\}$	18
$\{D,E\}$ bzw. $\{E,D\}$	8
$\{D,F\}$ bzw. $\{F,D\}$	8
$\{E,F\}$ bzw. $\{F,E\}$	2

Beispiel: Berechnung der edge betweenness



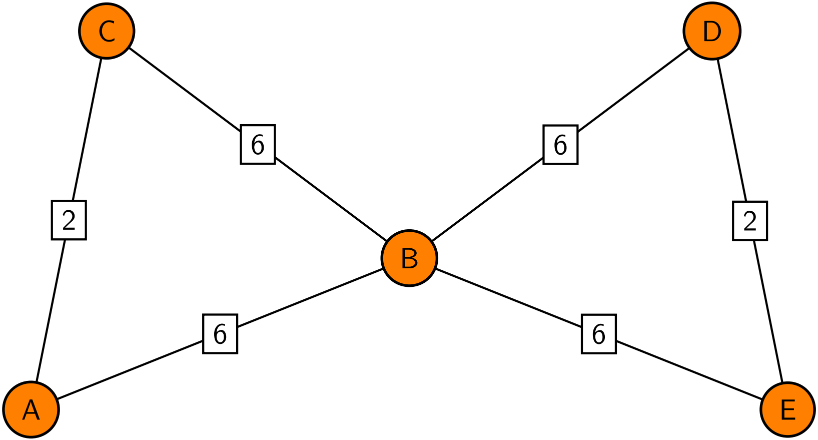
Beispiel: Berechnung der split betweenness



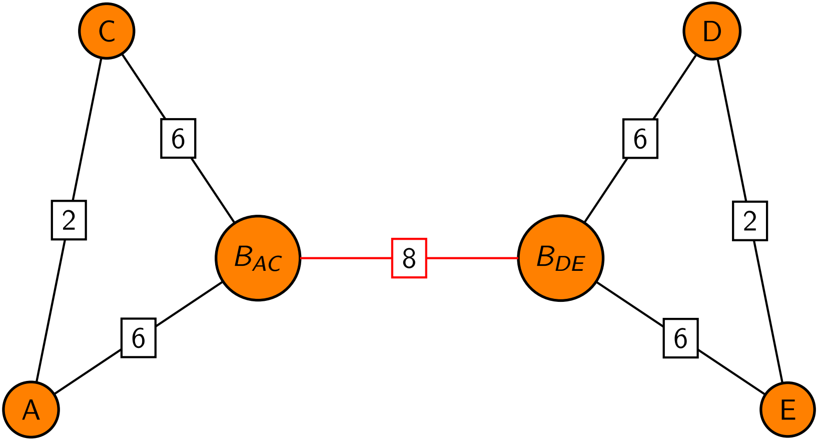
Beispiel: Berechnung der split betweenness

- Knoten B kann sowohl zum Cluster ABC als auch zum Cluster BDE gehören
- drei Möglichkeiten, Knoten B in zwei Teile zu splitten:
  - $\{B_{AC}, B_{DE}\}$ (vertikaler Schnitt)
  - $\{B_{CD}, B_{AE}\}$ (horizontaler Schnitt)
  - $\{B_{CE}, B_{AD}\}$ (diagonaler Schnitt)
- split betweenness ist Maximum der drei Werte

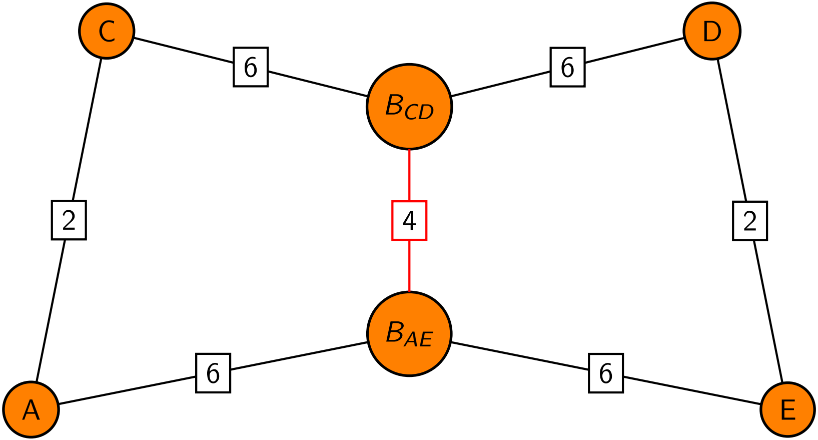
Beispiel: Berechnung der split betweenness



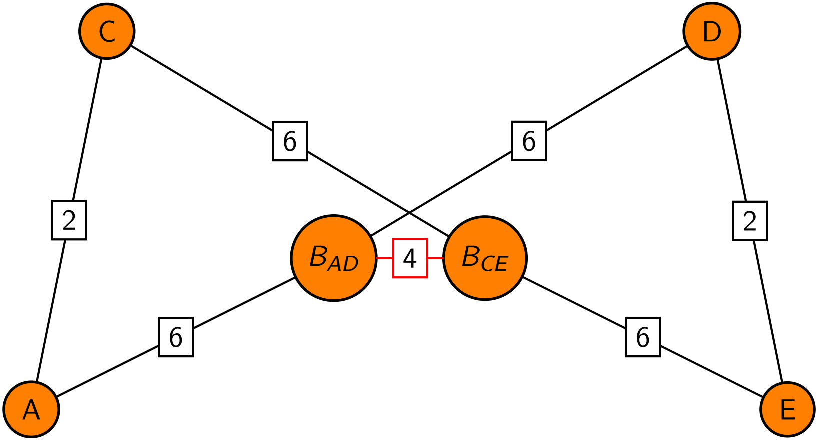
Beispiel: Vertikaler Schnitt



Beispiel: Horizontaler Schnitt



Beispiel: Diagonaler Schnitt



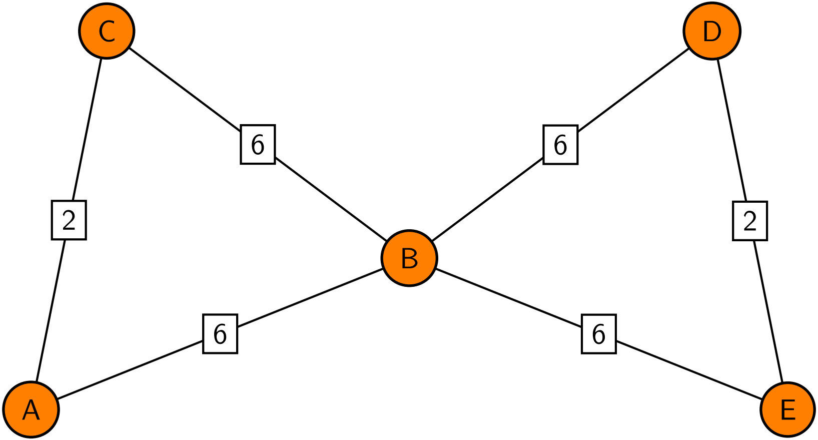
Pair betweenness

- Effiziente Berechnung der split betweenness mit Hilfe der pair betweenness

Definition (pair betweenness)

Die **pair betweenness** eines Knoten v für seine Nachbarknoten u und w ist die Anzahl der kürzesten Pfade durch die Kanten $\{u,v\}$ und $\{v,w\}$.

Beispiel: Berechnung der pair betweenness

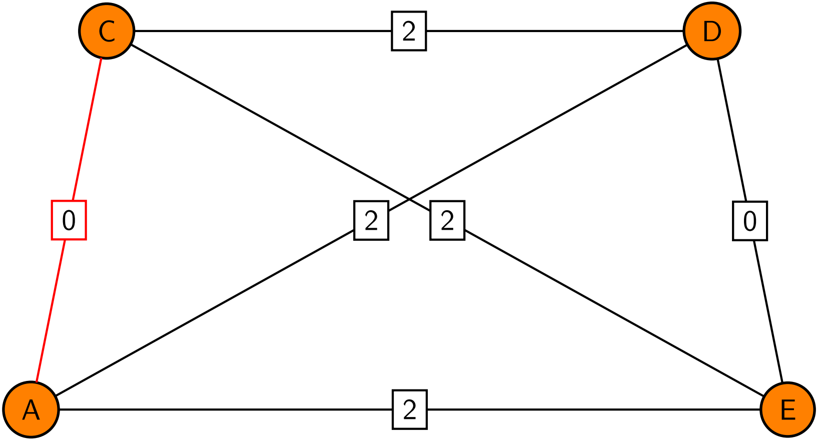


Beispiel: Berechnung der pair betweenness

Bestimmung der pair betweenness für Knoten B und seine Nachbarknoten:

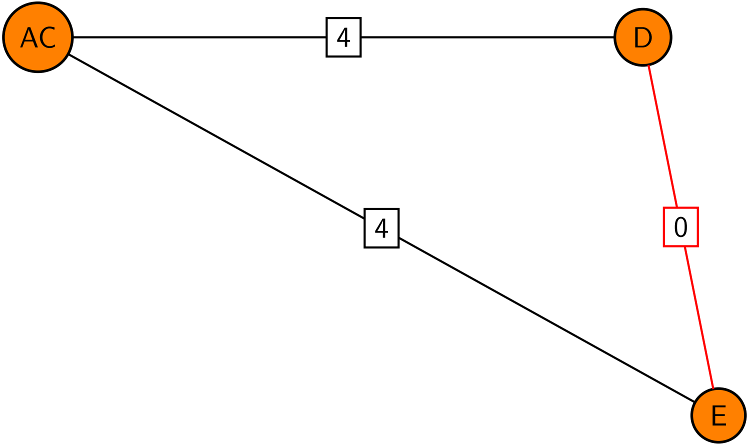
u	w	**pair betweenness**
A	C	0
A	D	2
A	E	2
C	D	2
C	E	2
D	E	0

Beispiel: Berechnung der pair betweenness



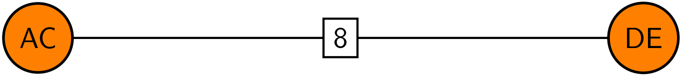
Beispiel: Berechnung der pair betweenness

Zusammenfassen der Knoten A und C:



Beispiel: Berechnung der pair betweenness

Zusammenfassen der Knoten D und E:



Der CONGA Algorithmus (2007)

while $E \neq \emptyset$:

1. for all $e \in E$: edgeBetweenness[e] = calculateEdgeBetweenness(e);
2. for all $v \in V$: splitBetweenness[v] = calculateSplitBetweenness(v);
3. Edge e_{max} := arg max(edgeBetweenness[e]);
4. Vertex v_{max} := arg max(splitBetweenness[v]);
5. if edgeBetweenness(e_{max}) $\geq$ splitBetweenness(v_{max}): $E = E \setminus \{e_{max}\}$;
   else: Vertex v_{split} = new Vertex(v_{max});

CONGA Algorithmus: Komplexität

- Für ein Netzwerk mit m Kanten und n Knoten gilt:
  - GN-Algorithmus: Worst case-Komplexität $\mathcal{O}(m^2 n)$
  - CONGA: $\mathcal{O}(m)$ Knoten nach dem Splitten
    $\Rightarrow$ Worst case-Komplexität von $\mathcal{O}(m^3)$
- Verbesserungsmöglichkeiten durch Beschränkung auf lokales Umfeld eines Knotens?

Local betweenness

Definition (h-betweenness)

Die h-**betweenness** einer Kante e ist die Anzahl der kürzesten Pfade zwischen allen Knotenpaaren (v_1, v_2), die e benutzen, mit höchstens der Länge h.

Definition (h-pair betweenness)

Die h-**pair betweenness** eines Knoten v für seine Nachbarknoten u und w ist die Anzahl der kürzesten Pfade durch die Kanten $\{u, v\}$ und $\{v, w\}$ mit höchstens der Länge h.

h-Region

Definition (h-Region einer Kante e)

Die h-**Region einer Kante** e ist der kleinste Subgraph, der alle kürzesten Wege mit höchstens Länge h enthält, die durch e laufen.

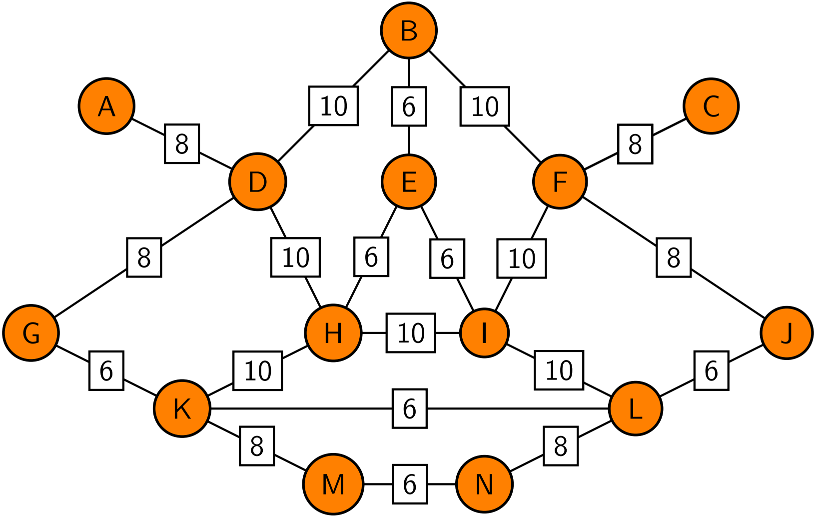
Definition (h-Region eines Knoten v)

Die h-**Region eines Knoten** v ist der kleinste Subgraph, der alle kürzesten Wege mit höchstens Länge h enthält, die v enthalten.

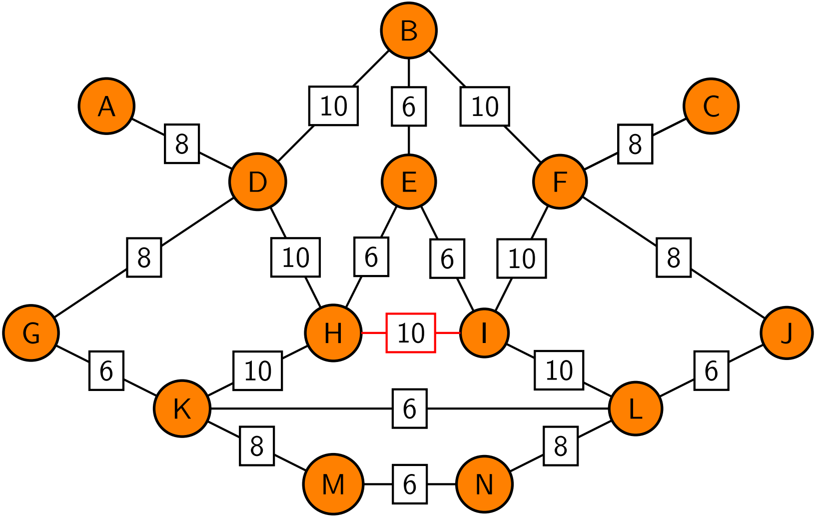
Der CONGO-Algorithmus (2008)

- CONGO = **CONG**A **O**ptimized
- Modifikation des CONGA-Algorithmus durch Beschränkung auf h-Region der entfernten Kante e_{max} bzw. des gesplitteten Knotens v_{max} bei der Neuberechnung der edge bzw. split betweenness
- Methode zur lokalen Neuberechnung:
  1. Finden aller Kanten der h-Region von e_{max} bzw. aller Knoten der h-Region von v_{max}
  2. Berechnen der h-betweenness für alle Kanten bzw. Knoten innerhalb der h-Region und Subtraktion von der aktuellen edge bzw. split betweenness
  3. Entfernen von e_{max} bzw. v_{max}
  4. Neuberechnung der edge bzw. split betweenness für h-Region

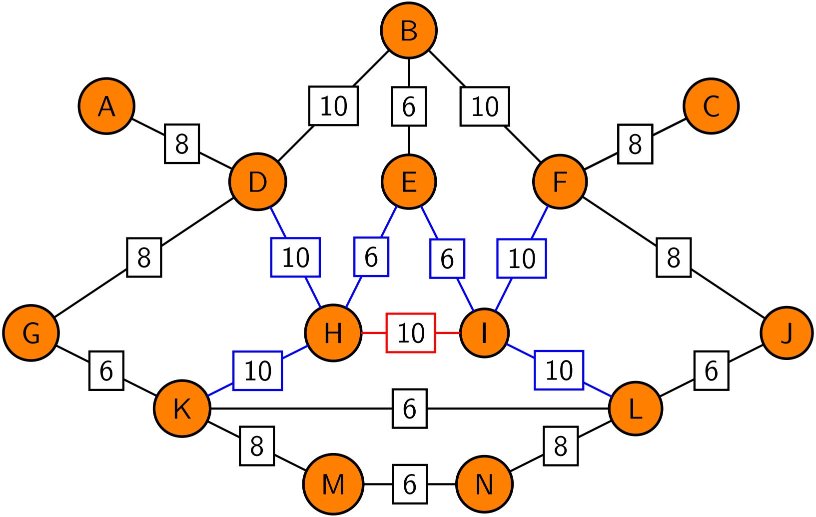
Beispiel: Neuberechnung der edge betweenness ($h = 2$)



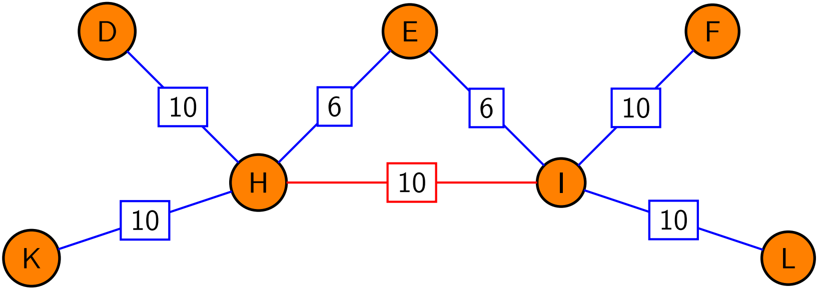
Beispiel: Neuberechnung der edge betweenness ($h = 2$)



Beispiel: Neuberechnung der edge betweenness ($h = 2$)



Beispiel: Neuberechnung der edge betweenness ($h = 2$)



Beispiel: Neuberechnung der edge betweenness ($h = 2$)

Bestimmung der kürzesten Pfade zwischen allen Knoten der 2-Region von $\{HI\}$ mit der maximalen Länge $h = 2$:

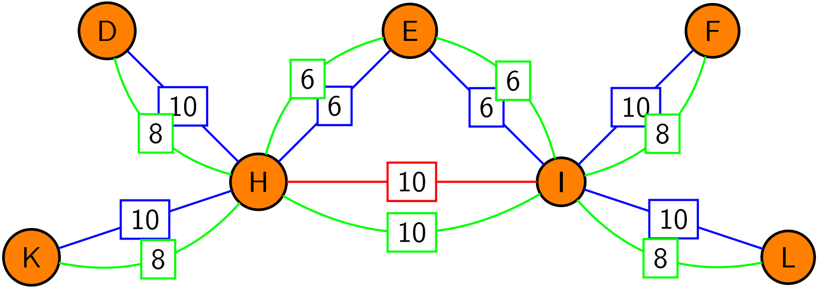
v_1	v_2	**kürzester Pfad**	v_1	v_2	**kürzester Pfad**
D	E	$\{D,H\}$, $\{H,E\}$	F	H	$\{F,I\}$, $\{I,H\}$
D	H	$\{D,H\}$	F	I	$\{F,I\}$
D	I	$\{D,H\}$, $\{H,I\}$	F	L	$\{F,I\}$, $\{I,L\}$
D	K	$\{D,H\}$, $\{H,K\}$	H	I	$\{H,I\}$
E	F	$\{E,I\}$, $\{I,F\}$	H	K	$\{H,K\}$
E	H	$\{E,H\}$	H	L	$\{H,I\}$, $\{I,L\}$
E	I	$\{E,I\}$	I	L	$\{I,L\}$
E	K	$\{E,H\}$, $\{H,K\}$	I	K	$\{I,H\}$, $\{H,K\}$
E	L	$\{E,I\}$, $\{I,L\}$			

Beispiel: Neuberechnung der edge betweenness ($h = 2$)

Bestimmung der edge betweenness für alle Kanten:

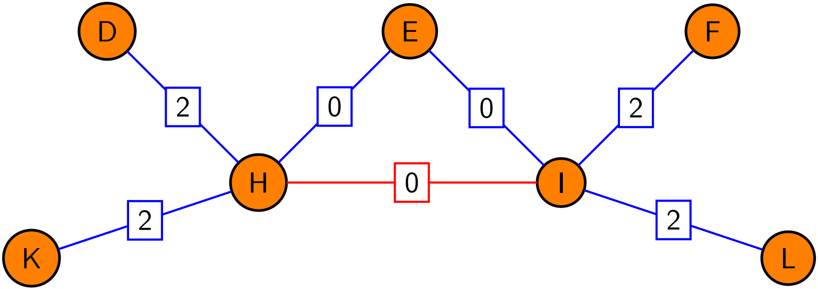
Kante	Anzahl
$\{D,H\}$ bzw. $\{H,D\}$	8
$\{E,H\}$ bzw. $\{H,E\}$	6
$\{E,I\}$ bzw. $\{I,E\}$	6
$\{F,I\}$ bzw. $\{I,F\}$	8
$\{H,I\}$ bzw. $\{I,H\}$	10
$\{H,K\}$ bzw. $\{K,H\}$	8
$\{I,L\}$ bzw. $\{L,I\}$	8

Beispiel: Neuberechnung der edge betweenness ($h = 2$)

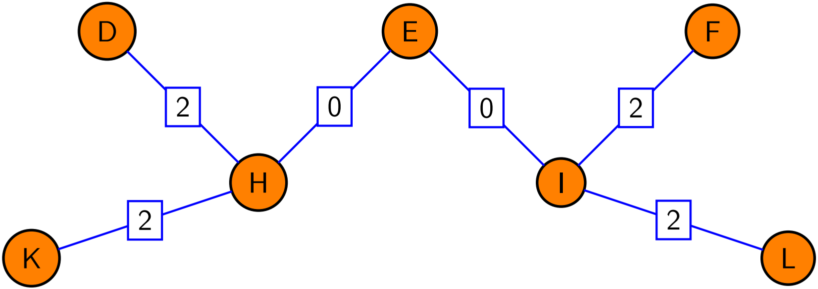


blau: alte edge betweenness
grün: neu berechnete h-betweenness mit $h = 2$

Beispiel: Neuberechnung der edge betweenness ($h = 2$)



Beispiel: Neuberechnung der edge betweenness ($h = 2$)



Beispiel: Neuberechnung der edge betweenness ($h = 2$)

Bestimmung der kürzesten Pfade maximal der Länge h zwischen allen Knoten des neuen Subgraphen:

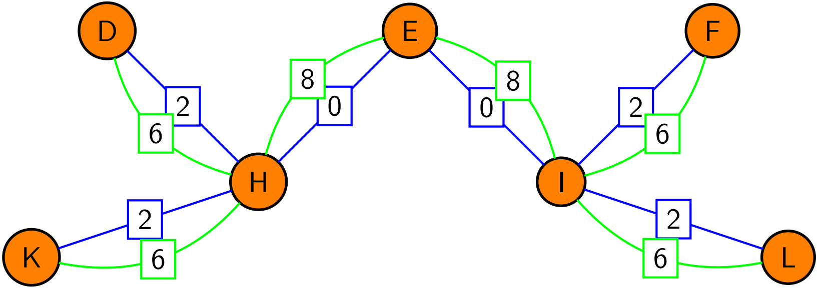
v_1	v_2	**kürzester Pfad**	v_1	v_2	**kürzester Pfad**
D	E	$\{D,H\}$, $\{H,E\}$	E	L	$\{E,I\}$, $\{I,L\}$
D	H	$\{D,H\}$	F	I	$\{F,I\}$
D	K	$\{D,H\}$, $\{H,K\}$	F	L	$\{F,I\}$, $\{I,L\}$
E	F	$\{E,I\}$, $\{I,F\}$	H	I	$\{H,E\}$, $\{E,I\}$
E	H	$\{E,H\}$	H	K	$\{H,K\}$
E	I	$\{E,I\}$	I	L	$\{I,L\}$
E	K	$\{E,H\}$, $\{H,K\}$			

Beispiel: Neuberechnung der edge betweenness ($h = 2$)

Bestimmung der edge betweenness für alle Kanten:

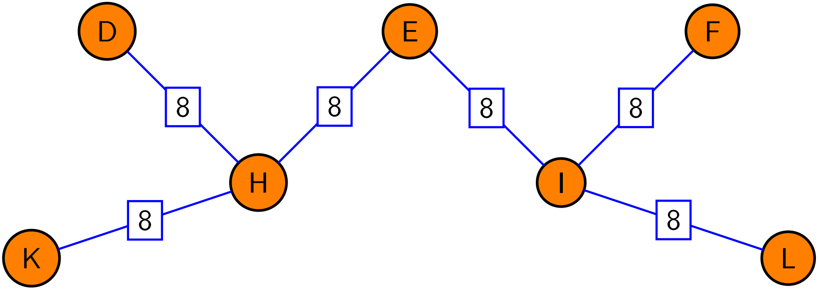
Kante	**Anzahl**
$\{D,H\}$ bzw. $\{H,D\}$	6
$\{E,H\}$ bzw. $\{H,E\}$	8
$\{E,I\}$ bzw. $\{I,E\}$	8
$\{F,I\}$ bzw. $\{I,F\}$	6
$\{H,K\}$ bzw. $\{K,H\}$	6
$\{I,L\}$ bzw. $\{L,I\}$	6

Beispiel: Neuberechnung der edge betweenness ($h = 2$)

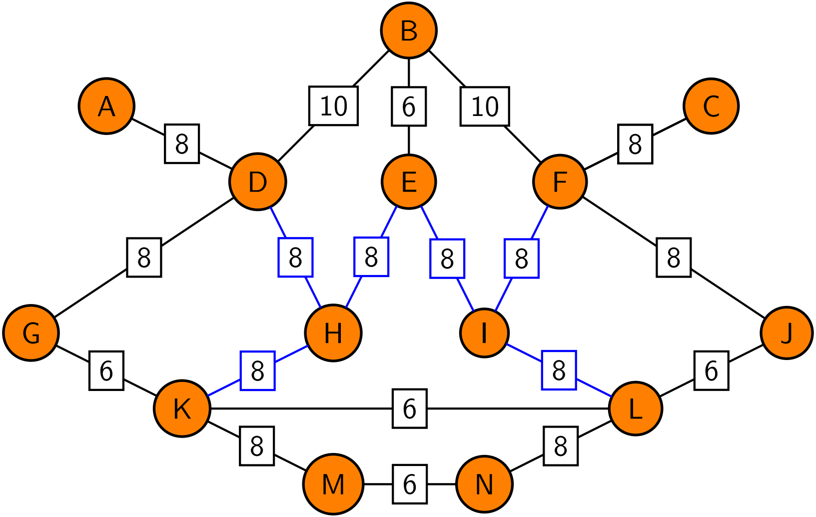


blau: alte edge betweenness
grün: neu berechnete h-betweenness mit $h = 2$

Beispiel: Neuberechnung der edge betweenness ($h = 2$)



Beispiel: Neuberechnung der edge betweenness ($h = 2$)



CONGO-Algorithmus: Komplexität

- Die Laufzeit des CONGO-Algorithmus hängt in der Praxis stark von der Struktur des Netzwerks ab
- Vereinfachende Annahme: Alle Knoten v haben den Grad $d(v) = \frac{2m}{n}$
- Worst case-Komplexität von CONGO: $\mathcal{O}(m \cdot log(m) + \frac{m^{2h+2}}{n^{2h+1}})$
- Laufzeit für kleine m: $\mathcal{O}(n \cdot log(n))$

F-Measure

- Künstliche Netzwerke zum Testen von Clustering-Algorithmen (Vergleich der tatsächlichen Communities mit den vom Algorithmus gefundenen Clustern)
- Verwendetes Gütemaß: F-Measure

Definition (F-Measure)

Das F-Measure ist das harmonische Mittel aus Precision und Recall:

$$F = \frac{2 \cdot p \cdot r}{p + r}$$

Precision und Recall

Definition (Precision)

Die Precision p gibt an, welcher Anteil der gefundenen Dokumente auch relevant sind.

$$p = \frac{|\{relevant\} \cap \{retrieved\}|}{|\{retrieved\}|}$$

Definition (Recall)

Der Recall r gibt an, welcher Anteil der relevanten Dokumente auch gefunden wurde.

$$r = \frac{|\{relevant\} \cap \{retrieved\}|}{|\{relevant\}|}$$

Precision und Recall

Definition (Precision)

Die Precision p gibt an, welcher Anteil der Knoten im selben Cluster auch zur selben Community gehört.

$$p = \frac{|\{community\} \cap \{cluster\}|}{|\{cluster\}|}$$

Definition (Recall)

Der Recall r gibt an, welcher Anteil der Knoten derselben Community auch zum selben Cluster zugeordnet wurde.

$$r = \frac{|\{community\} \cap \{cluster\}|}{|\{community\}|}$$

Modularity

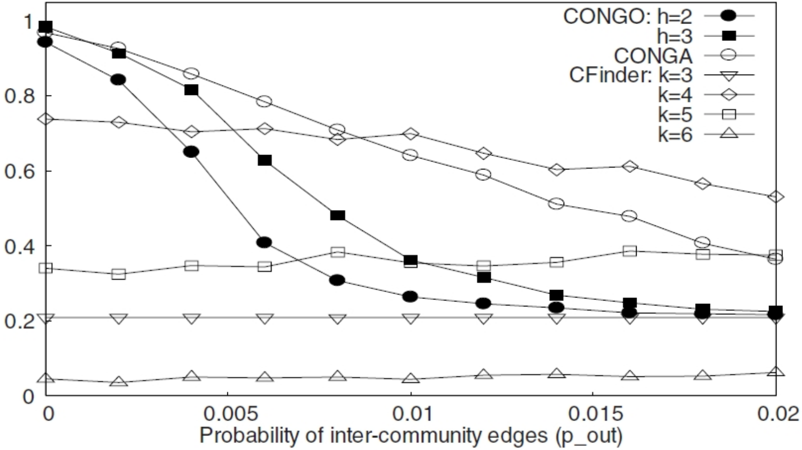
- Problem: Precision und Recall i. A. nur auf künstlichen Netzwerken berechenbar, da die tatsächlichen Communities nicht bekannt sind
- stattdessen z. B. Berechnung der Modularity:
  - $Q_{ov} = 0$: Alle Knoten gehören entweder zur selben Community, oder alle Knoten bilden einzelne Communities
  - je größer Q_{ov}, desto stärker die Community-Struktur
  - Jeder Knoten kann zu beliebig vielen Communities gehören $\rightarrow$ Modellierung durch *belonging coefficient*

Experimente mit künstlichen Netzwerken

- Netzwerk wird randomisiert erzeugt mit folgenden Parametern:
  - n: Anzahl der Knoten
  - c: Anzahl der Communities
  - r: Maß für Überlappung der Communities $(1 < r < c)$
  - $\frac{n \cdot r}{c}$: Anzahl der Knoten pro Community
  - $i(v_1, v_2)$: Anzahl der Communities, zu denen die Knoten v_1 und v_2 gleichzeitig gehören
  - $i(v_1, v_2) \cdot p_{in}$: Wahrscheinlichkeit für die Erzeugung der Kante $\{v_1, v_2\}$
  - p_{out}: Wahrscheinlichkeit für die Erzeugung einer Kante zwischen zwei Knoten aus verschiedenen Communities

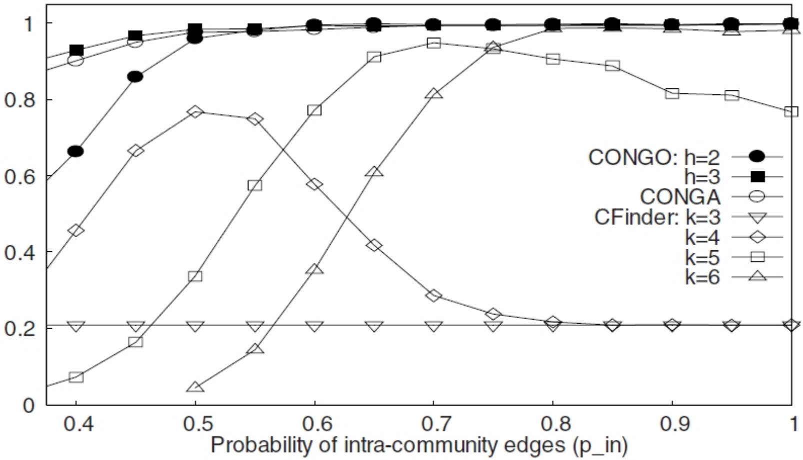
Experimente mit künstlichen Netzwerken

F-Measures für Netzwerke mit $n = 256$, $c = 32$, $r = 2$, $p_{in} = 0.5$:



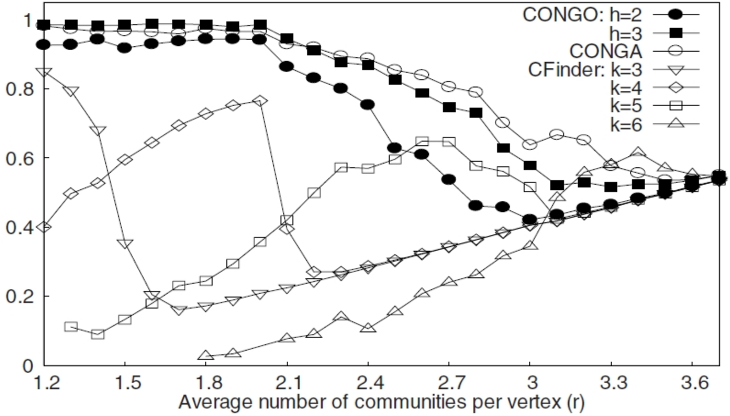
Experimente mit künstlichen Netzwerken

F-Measures für Netzwerke mit $n = 256$, $c = 32$, $r = 2$, $p_{out} = 0$:



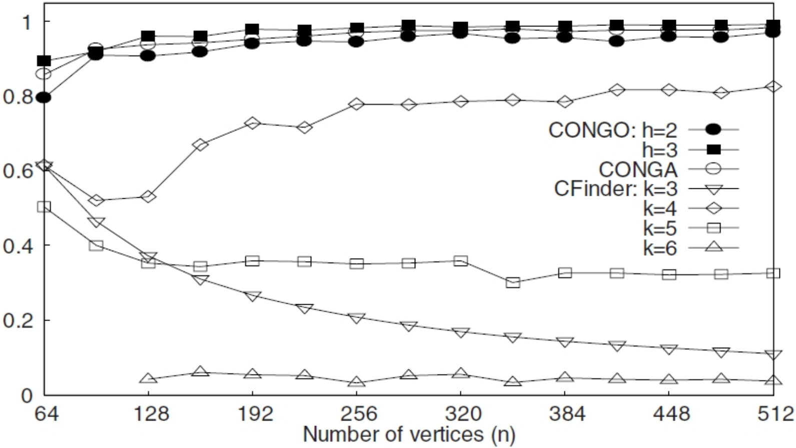
Experimente mit künstlichen Netzwerken

F-Measures für Netzwerke mit $n = 256$, $c = 32$, $p_{in} = 0.5$, $p_{out} = 0$:



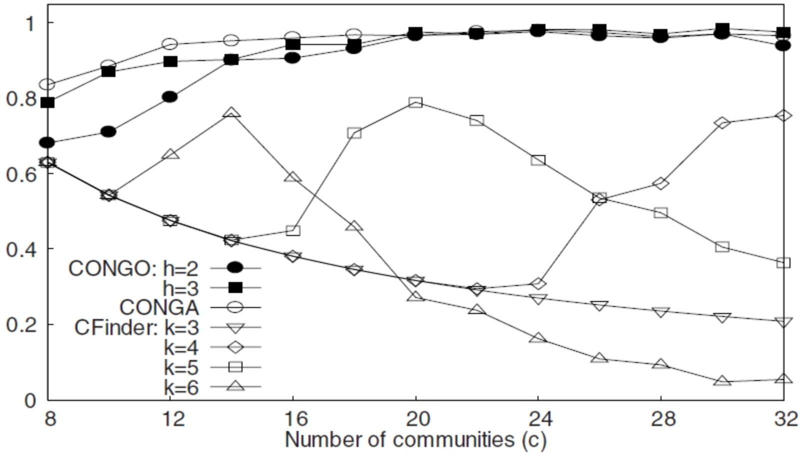
Experimente mit künstlichen Netzwerken

F-Measures für Netzwerke mit $c = \frac{n}{8}$, $r = 2$, $p_{in} = 0.5$, $p_{out} = 0$:



Experimente mit künstlichen Netzwerken

F-Measures für Netzwerke mit $n = 256$, $r = 2$, $p_{in} = 0.5$, $p_{out} = 0$:



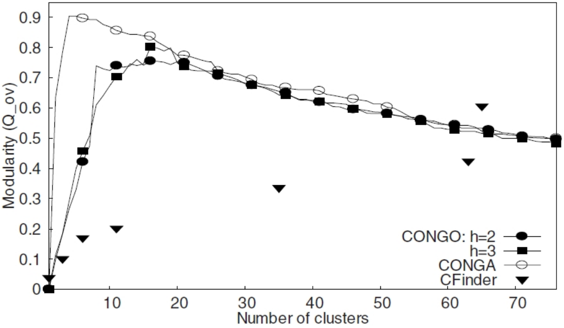
Experimente mit echten Netzwerken

Testen von CONGO und CFinder an sieben echten Netzwerken:

Name	**Knoten**	**Kanten**	**Rechenzeit in Sekunden**		
			CONGO		**CFinder**
			$h = 3$	$h = 2$	
netscience	379	914	1.4	1.3	0.3
cond-mat-2003	27 519	116 181	45 110	1 111	1 140
blogs	3 982	6 803	33.5	6.1	3.2
blogs2	30 557	82 301	11 702	286	405
PGP	10 680	24 316	636	82	35 022
word_association	7 205	31 784	12 026	172	97
protein-protein	2 445	6 265	94.5	8.2	2.9

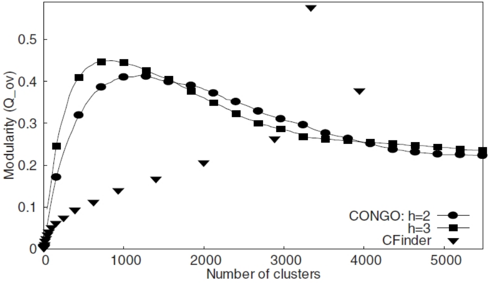
Netzwerk: netscience

Name	Knoten	Kanten	Rechenzeit in Sekunden		
			CONGO		CFinder
			$h = 3$	$h = 2$	
netscience	379	914	1.4	1.3	0.3



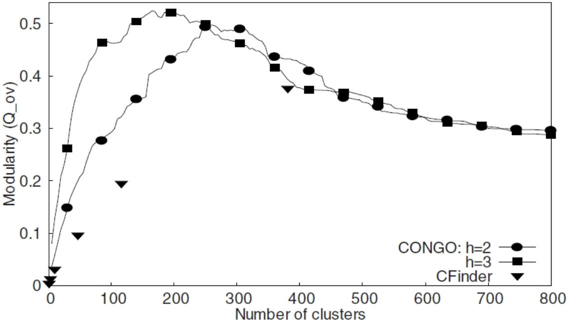
Netzwerk: cond-mat-2003

Name	Knoten	Kanten	Rechenzeit in Sekunden		
			CONGO		CFinder
			$h = 3$	$h = 2$	
cond-mat-2003	27 519	116 181	45 110	1 111	1 140



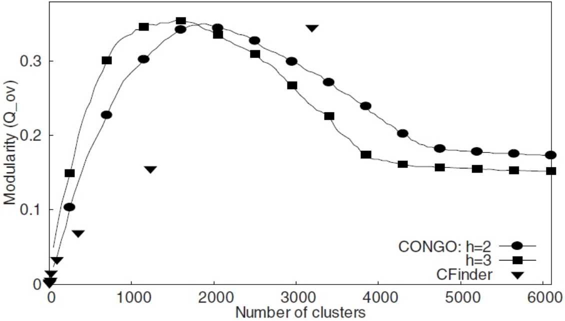
Netzwerk: blogs

Name	Knoten	Kanten	Rechenzeit in Sekunden		
			CONGO		CFinder
			$h = 3$	$h = 2$	
blogs	3 982	6 803	33.5	6.1	3.2



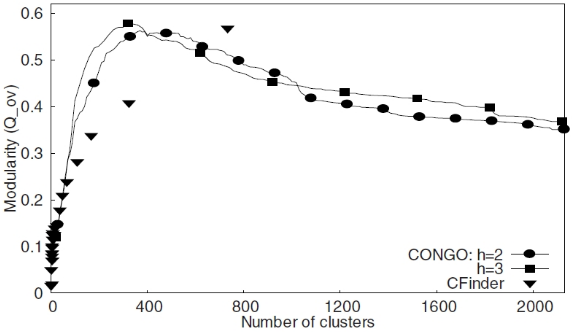
Netzwerk: blogs2

Name	Knoten	Kanten	Rechenzeit in Sekunden		
			CONGO		CFinder
			$h = 3$	$h = 2$	
blogs2	30 557	82 301	11 702	286	405



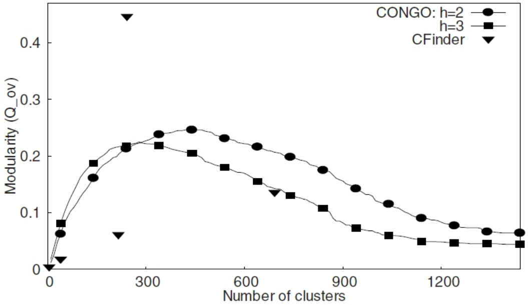
Netzwerk: PGP

Name	Knoten	Kanten	Rechenzeit in Sekunden		
			CONGO		CFinder
			$h = 3$	$h = 2$	
PGP	10 680	24 316	636	82	35 022



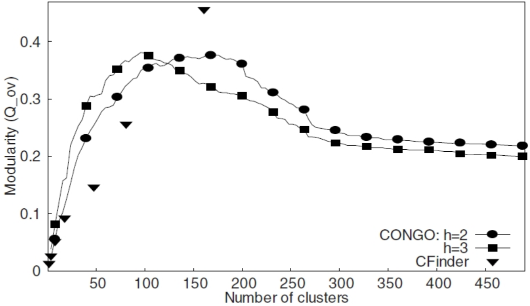
Netzwerk: word_association

Name	Knoten	Kanten	Rechenzeit in Sekunden		
			CONGO		CFinder
			$h = 3$	$h = 2$	
word_association	7 205	31 784	12 026	172	97



Netzwerk: protein-protein

Name	Knoten	Kanten	Rechenzeit in Sekunden		
			CONGO		CFinder
			$h = 3$	$h = 2$	
protein-protein	2 445	6 265	94.5	8.2	2.9



Zusammenfassung und Ausblick

Schlussfolgerungen des Autors:

- CONGO kann effektiv und schnell überlappende Communities in Netzwerken finden
- speziell für $h = 2$ ist CONGO schneller als CONGA
- CONGO arbeitet auf großen Netzwerken generell schneller als CFinder

Zukünftige Forschungsaufgaben:

- Verbesserung der Schnelligkeit durch dynamisches h?
- Einführung von belonging coefficients zur Gewichtung der Zugehörigkeit eines Knotens zu seinen Communities?

Literatur

Gregory, Steve (2008): **A fast algorithm to find overlapping communities in networks**. In: *Proceedings of the 12th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2008)*. ISBN 978-3-540-87478-2, pp. 408–423.

Gregory, Steve (2007): **An algorithm to find overlapping community structure in networks**. In: *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2007)*. ISBN 978-3-540-74975-2, pp. 91–102.