Restricted Neighbourhood Search Clustering

Michael Petrifke

Universität Leipzig
Institut für Informatik

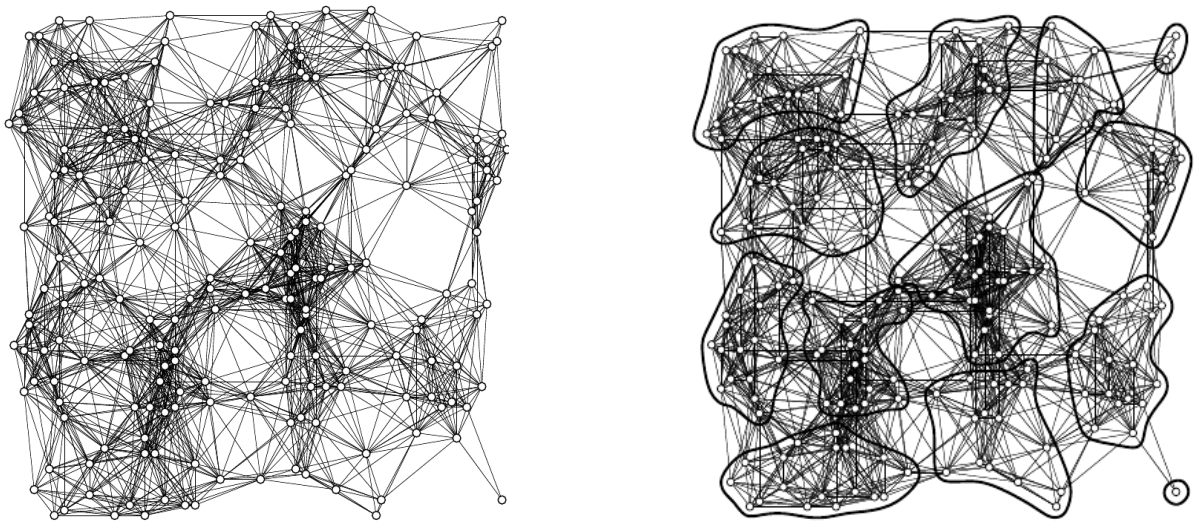
13. Mai 2009

1. Einführung
   - Graph Clustering
   - Begriffe
2. Algorithmus
   - Grobe Beschreibung
   - Kostenfunktionen
   - Initialzustand
   - Erster Teil des Algorithmus
   - Zweiter Teil des Algorithmus

3. Bemerkungen
   - Tabu-Liste
   - Diversification
   - Parameter
   - Komplexität
   - Randomized Local Search
4. Evaluation
   - Allgemein
   - Vergleiche auf Zufälligen Graphen
   - Vergleiche mit PPI
     - Bewertung
     - Parameterevaluation
     - Vergleiche

Was ist Graph Clustering?

- Graph G=(V,E)
- Zerlegung von V in k Partitionen $C_1, C_2, \ldots, C_k$
- Partitionen induzieren Subgraphen



(a) Ohne Clustering (b) Mit Clustering

Figure: Clustering von $G_G(200, 100, 100, 20)$, Bild: [1]

Bemerkungen

- Was ist ein gutes Clustering?
  - Viele Kanten innerhalb eines Clusters
  - Wenige Kanten zwischen den Clustern
- Große Zahl an möglichen Clusterings
- Bell-Zahlen geben Anzahl an Clustering an, wachsen superexponentiell, 9 Knoten: 21147 Möglichkeiten, 10 Knoten: 115975, 11 Knoten: 678570

Begriffe

- Graph $G = (V, E)$ mit Knotenmenge V, Kantenmenge E
- $|V| = n$, $|E| = m$
- $v \in V : N(v)$ sind alle Knoten, die mit v verbunden sind
- C_v ist Cluster, welches v enthält

Aufbau

- Beginn mit initialem Clustering
- Schrittweise einen Knoten in anderes Cluster bewegen
- Auswahl des Knotens anhand Kostenfunktion
- Schritt soll Gesamtkosten des Clustering verringern

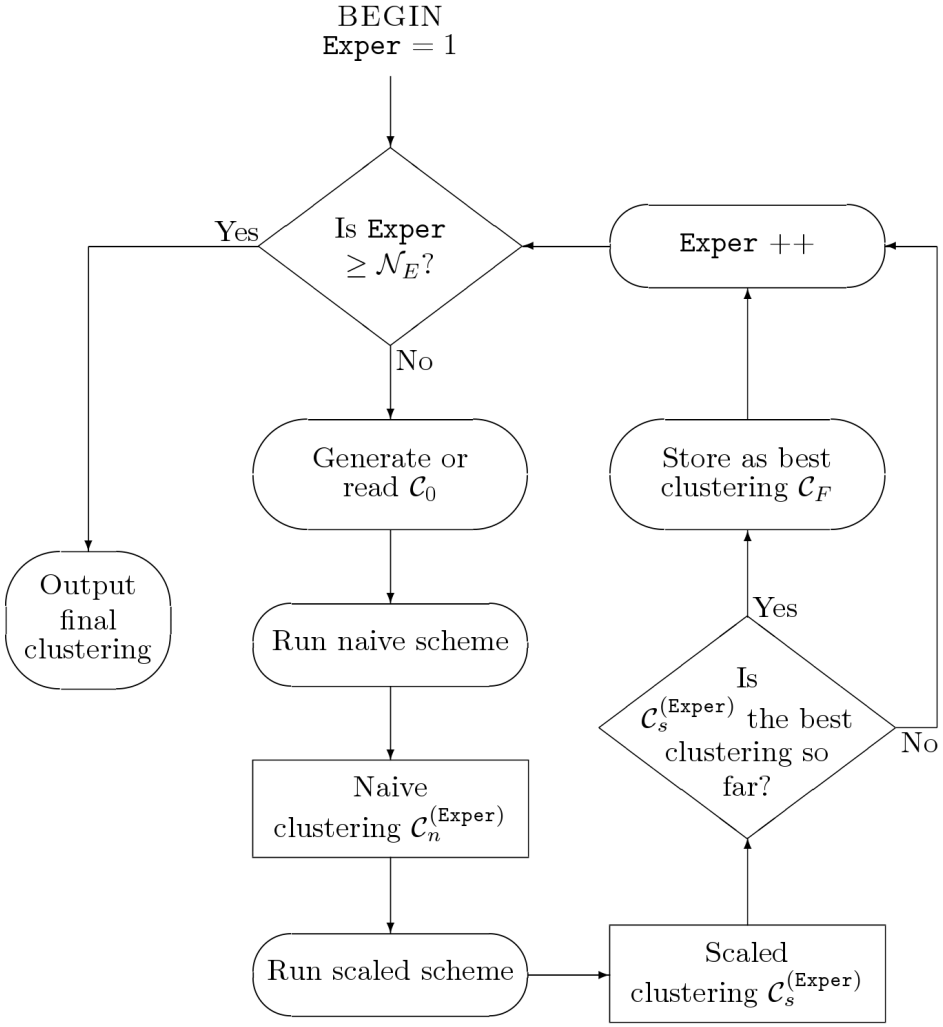


Figure: Bild: [1]

Kostenfunktionen

- Nutzt zwei Kostenfunktionen zur Clustering-Bewertung:
  - Naive Cost Function
  - Scaled Cost Function
- Erster Schritt mit Naive Cost Function bis keine Verbesserung mehr eintritt
- Anschließend Verfeinerung mit Scaled Cost Function
- Bewegung in beiden Kostenfunktionen ändern Kosten um maximal n-1

Naive Cost Function

- $C(G, C) = \frac{1}{2} \sum_{v \in V} \alpha_v$
- α_n Anzahl schlechter Kanten bzgl. v
- gibt gutes Clustering in kurzer Zeit

Scaled Cost Function

- $C(G, C) = \frac{n-1}{3} \sum_{v \in V} \frac{\alpha_v}{\beta_v}$
- $\beta_v = |N(v) \cup C_v|$
- bezieht mehr Umstände ein, verfeinert Suche

Initialzustand

- Initiales Clustering C_0 vorgegeben (Anwender) oder zufallsgeneriert
- Algorithmus nimmt feste Zahl an Cluster an (leere Cluster erlaubt)

Erster Teil

- Starte mit C_0
- Bewerte Clustering anhand Naive Cost Function
- Merke bestes Clustering C_n sowie beste Kosten
- Abbruch falls sich bestes Clustering innerhalb von T_n Zügen nicht ändert
- Wähle nächste mögliche Bewegung mit geringsten Kosten (des entstehenden Clusters), optimale Wahl
- Mehrere Möglichkeiten: Zufällige Wahl mit Gleichverteilung

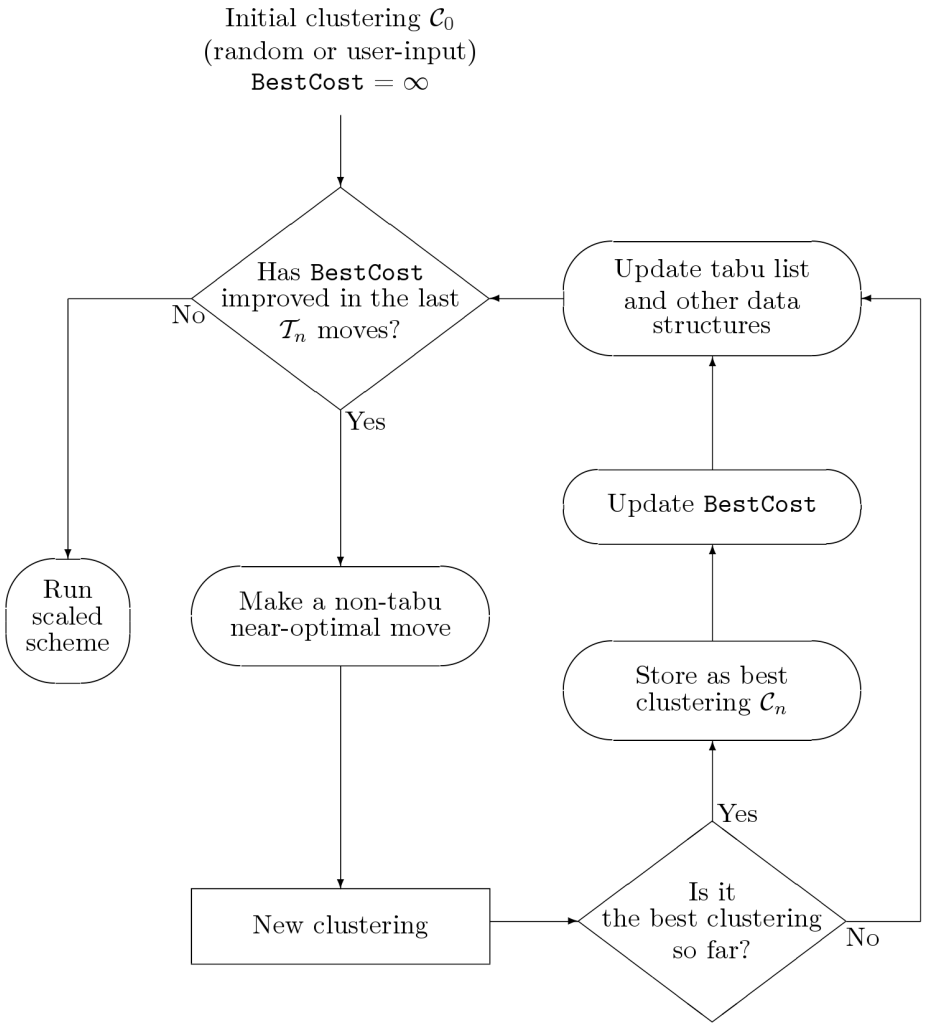


Figure: Bild: [1]

Zweiter Teil

- Starte mit C_n
- Bewerte Clustering anhand Scaled Cost Function
- Merke bestes Clustering C_S sowie beste Kosten
- Abbruch falls sich bestes Clustering innerhalb von T_s Zügen nicht ändert
- Wähle nächste mögliche Bewegung mit geringsten Kosten (des entstehenden Clusters)
- Kosten auf Ganzzahl aufrunden, Wahl near-optimal
- Mehrere Möglichkeiten: Zufällige Wahl mit Gleichverteilung
- In bestimmter Frequenz zufällige Bewegungen durch Diversification Move

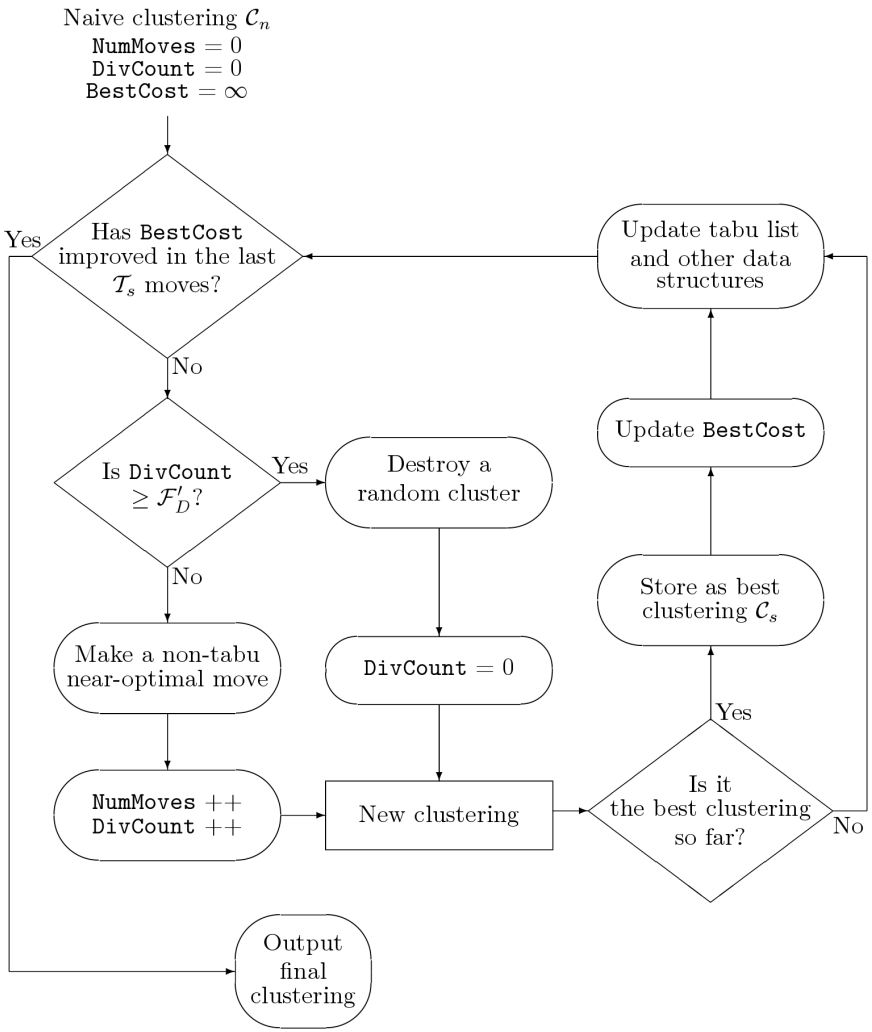


Figure: Bild: [1]

Eigenschaften

- Erlaubt keine Mehrfachzuweisung
- Nur für ungewichtete Graphen
- Knoten müssen zugewiesen werden

Tabu-Liste

- Erlaube nicht alle Clusterings
- Vermeidet Zyklen
- Ermöglicht Ausbrechen aus lokalem Minimum
- Weiterhin: Verbiete Bewegung von 1-Element Cluster zu leerem Cluster

Möglichkeiten

- Komplette Clusterings verbieten
  - Viele Möglichkeiten, benötigt viel Speicher
  - Aufwendige Suche
- Bewegungen verbieten
  - Ebenso aufwendig
- Einfacher:
  - Merke zuletzt benutze Knoten, diese dürfen nicht bewegt werden
  - FIFO fester Größe

Diversification

- Führe zufällige Bewegungen aus
- Ermöglicht Untersuchung weiterer lokaler Minima
- Zwei Methoden:
- Shuffling Diversification
  - Nach fester Zahl an Zügen, führe feste Zahl an zufälligen Bewegungen aus
  - Problem: Bewegungen unabhängig voneinander, keine Strukturänderung
- Destructive Diversification
  - Löse feste Zahl an Clustern auf, verteile Knoten an andere Cluster
  - Vorteil: Struktur ändert sich, effektiver

Parameter

- Maximale Anzahl an Clusterings $\mathcal{N}_C$
- Anzahl der Experimente $\mathcal{N}_E$
- Länge der Tabu-Liste $\mathcal{L}_T$
- Tabu Toleranz $\mathcal{T}_T$
- Suchlängen $\mathcal{T}_n, \mathcal{T}_s$
- Parameter für Diversification: Shuffling Frequency $\mathcal{F}_D$, Diversification Length $\mathcal{L}_D$, Destructive Frequency $\mathcal{F}'_D$

Komplexität

- Schritt mit Naive Cost Function in $O(n)$
  - Idee: Schritt muss Kosten aller möglichen $(\mathcal{N}_C - 1) \cdot n$ Bewegungen aktualisieren, Kostendifferenz für Bewegung auf völlig anderen Clustern bleibt gleich, sonst Kostenänderung bekannt (abhängig von Startcluster, Zielcluster, bewegter Knoten benachbart zu vorher bewegtem Knoten), damit Änderung pro Bewegung in konstanter Zeit
- Schritt mit Scaled Cost Function in $O(n^2)$
  - Idee: wie oben $(\mathcal{N}_C - 1) \cdot n$ Bewegungen aktualisieren, aber Berechnung jeweils nicht in konstanter Zeit (für alle Knoten aus Start- sowie Zielcluster notwendig)
- Speicherverbrauch $O(n^2)$

Randomized Local Search

- Nicht deterministisch
- Unterschiedliche Ergebnisse aber auch Vorteil:
- Mehrere Durchläufe ergeben verschiedene Clusterings, wähle bestes
- Findet nur lokales Minimum:
  - Tabu-Liste um an anderen Stellen zu suchen
  - Diversification um lokales Minimum zu verlassen

Welches Clustering ist gut?

- Oft subjektiv
- Hängt von Kontext ab, unterschiedliche Anwendungen haben unterschiedliche Anforderungen
- Keine Einheitliche Benchmark-Funktion

Vergleichsmöglichkeiten

- Zufällige Graphen
  - Erdös-Renyi Graph $G_E(n, p)$, verbinde zwei Knoten mit Wahrscheinlichkeit p
  - Scale-Free Graph $G_S(n, k)$, wähle k Knoten als Startgraph $G^{(k)}$, füge sukzessiv fehlende $n - k$ Knoten hinzu, für jeden Knoten zufällig k Kanten, Knoten mit höherem Grad wahrscheinlicher
  - Geometrischer Graph $G_G(n, l, w, d)$, verteile n Knoten auf Grid Größe $l \times w$, Kante falls Distanz zwischen Knoten kleiner d
- Direkte Anwendung auf reale Daten
  - Protein-Protein-Interaktion

Vergleich mit MCL auf Zufälligen Graphen (200 Knoten)

Graph	Time	Time per scaled move	# of clusters	Scaled cost	Scaled cost of MCL output
$G_E^{200,s}$	0.150s	1.76×10^{-3}	109	3741	5400
$G_E^{200,m}$	0.198s	1.09×10^{-3}	61	7931	9225
$G_E^{200,d}$	0.273s	1.08×10^{-3}	29	10491	12404
$G_G^{200,s}$	0.129s	2.40×10^{-3}	90	2132	5143
$G_G^{200,m}$	0.155s	2.04×10^{-3}	47	4158	5527
$G_G^{200,d}$	0.233s	3.01×10^{-3}	14	5341	6523
$G_S^{200,s}$	0.127s	2.43×10^{-3}	123	4631	4745
$G_S^{200,m}$	0.193s	1.12×10^{-3}	76	7760	10139
$G_S^{200,d}$	0.328s	1.54×10^{-3}	33	10296	11900

Figure: Bild: [1]

Vergleich mit MCL auf Zufälligen Graphen (500 Knoten)

Graph	Time	Time per scaled move	# of clusters	Scaled cost	Scaled cost of MCL output
$G_E^{500,s}$	1.464	4.65×10^{-3}	188	46917	54750
$G_E^{500,m}$	1.819	4.28×10^{-3}	112	65724	76474
$G_E^{500,d}$	7.187	8.54×10^{-3}	26	72219	81094
$G_G^{500,s}$	1.040	8.52×10^{-3}	116	25594	34122
$G_G^{500,m}$	1.690	1.01×10^{-2}	47	34580	41918
$G_G^{500,d}$	2.651	2.61×10^{-2}	16	38712	45064
$G_S^{500,s}$	1.272	5.64×10^{-3}	198	45327	62878
$G_S^{500,m}$	1.825	4.02×10^{-3}	134	64517	76013
$G_S^{500,d}$	16.14	3.05×10^{-3}	44	70935	74800

Figure: Bild: [1]

Vergleich mit MCL auf Protein-Protein-Interaktionsgraphen

Graph	Method	Time	Time per scaled move	# of clusters	Scaled cost
Y_{2k}	RNSC	4.03s	.0032s	391	8.56×10^4
	MCL	0.38s	n/a	427	1.44×10^5
Y_{11k}	RNSC	68.3s	.193s	972	7.67×10^5
	MCL	3.42s	n/a	850	1.02×10^6
Y_{45k}	RNSC	916s	.956s	1815	4.07×10^6
	MCL	61.5s	n/a	1199	4.93×10^6
Y_{78k}	RNSC	1071s	.983s	1811	5.26×10^6
	MCL	102s	n/a	1179	6.49×10^6

Figure: Bild: [1]

Vergleich mit MCL, MCODE, SPC

- Bekannte PPI-Cluster als Graph dargestellt
- Weitere Graphen $A_{n,k}$ durch zufälliges Hinzufügen von n Prozent Kanten, Entfernen von k Prozent Kanten
- Abgrenzung von zufälliger richtiger Clusterzuweisung: Zufallsgraphen (Knotengrad beibehalten), Vertauschen der Knoten in resultierenden Clustern (Clustergröße beibehalten)

Bewertung

- Betrachte Matrix $T_{n \times m}$, $T_{i,j}$: gemeinsame Proteine von Complex i und Cluster j
- $F_{\text{row}_{i,j}} = \frac{T_{i,j}}{\sum_{j=1}^{m} T_{i,j}}$
- $F_{\text{col}_{i,j}} = \frac{T_{i,j}}{\sum_{i=1}^{n} T_{i,j}}$
- Separation: $Sep_{i,j} = F_{\text{row}_{i,j}} \cdot F_{\text{col}_{i,j}}$
- Complex-wise Seperation: $Sep_{co_i} = \sum_{j=1}^{m} Sep_{i,j}$, Mittelwert: Sep_{co}
- Cluster-wise Seperation: $Sep_{cl_j} = \sum_{i=1}^{n} Sep_{i,j}$, Mittelwert: Sep_{cl}
- Geometrical Seperation: $Sep = \sqrt{Sep_{co} \cdot Sep_{cl}}$

Parameterevaluation für RNSC

- Alle Parameter der Algorithmen evaluiert für maximale Bewertung

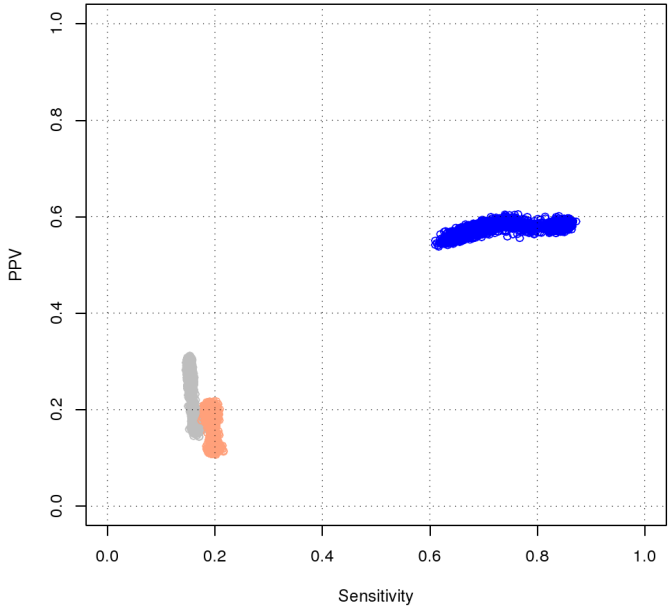


Figure: blau: $A_{100,40}$, orange: $R_{100,40}$, grau: permutierte Cluster, Bild: [2]

Hinzufügen von Kanten zu Test Graph

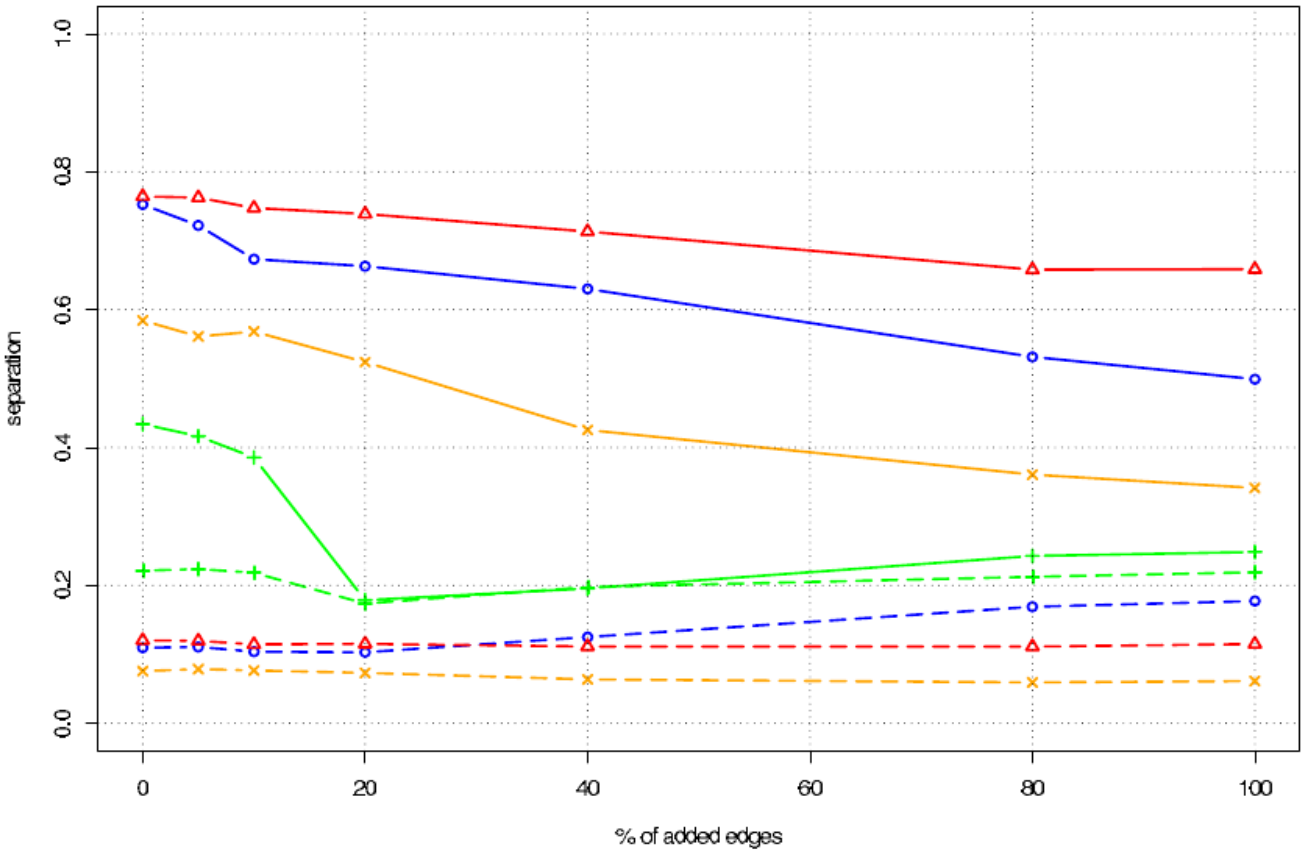


Figure: blau: MCL, rot: RNSC, orange: MCODE, grün: SPC, Bild: [2]

Entfernen von Kanten des Test Graph

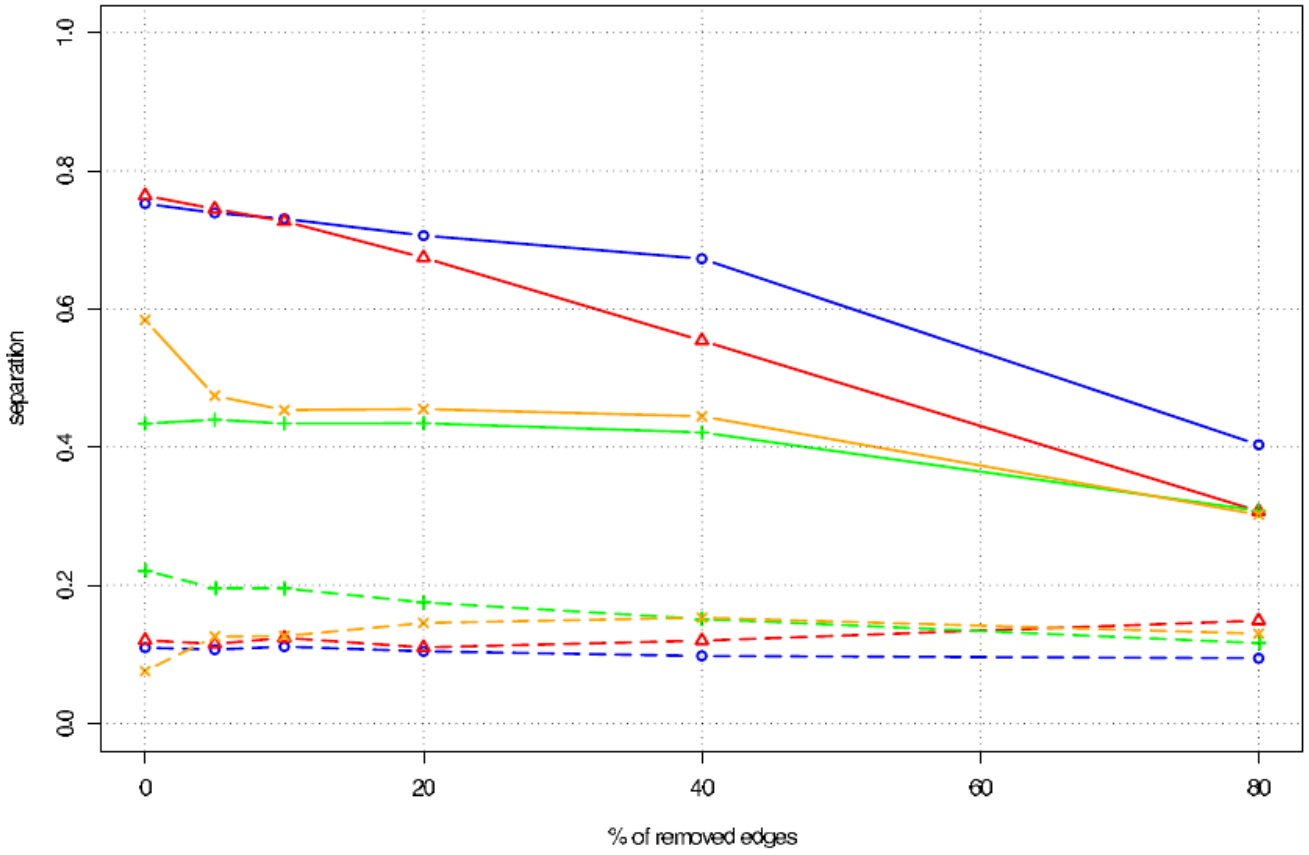


Figure: blau: MCL, rot: RNSC, orange: MCODE, grün: SPC, Bild: [2]

Hinzufügen von Kanten zu $A_{k,40}$

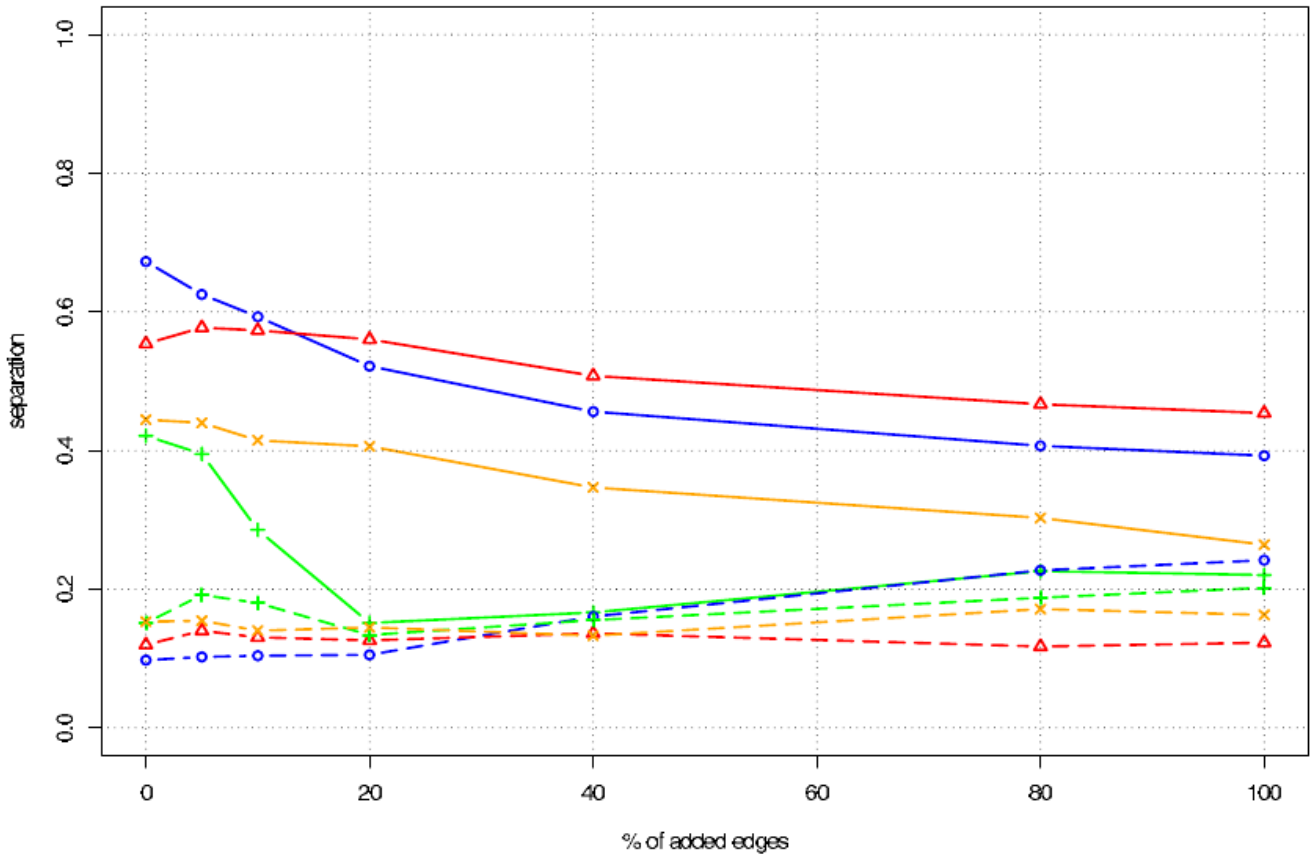


Figure: blau: MCL, rot: RNSC, orange: MCODE, grün: SPC, Bild: [2]

Entfernen von Kanten des $A_{100,l}$

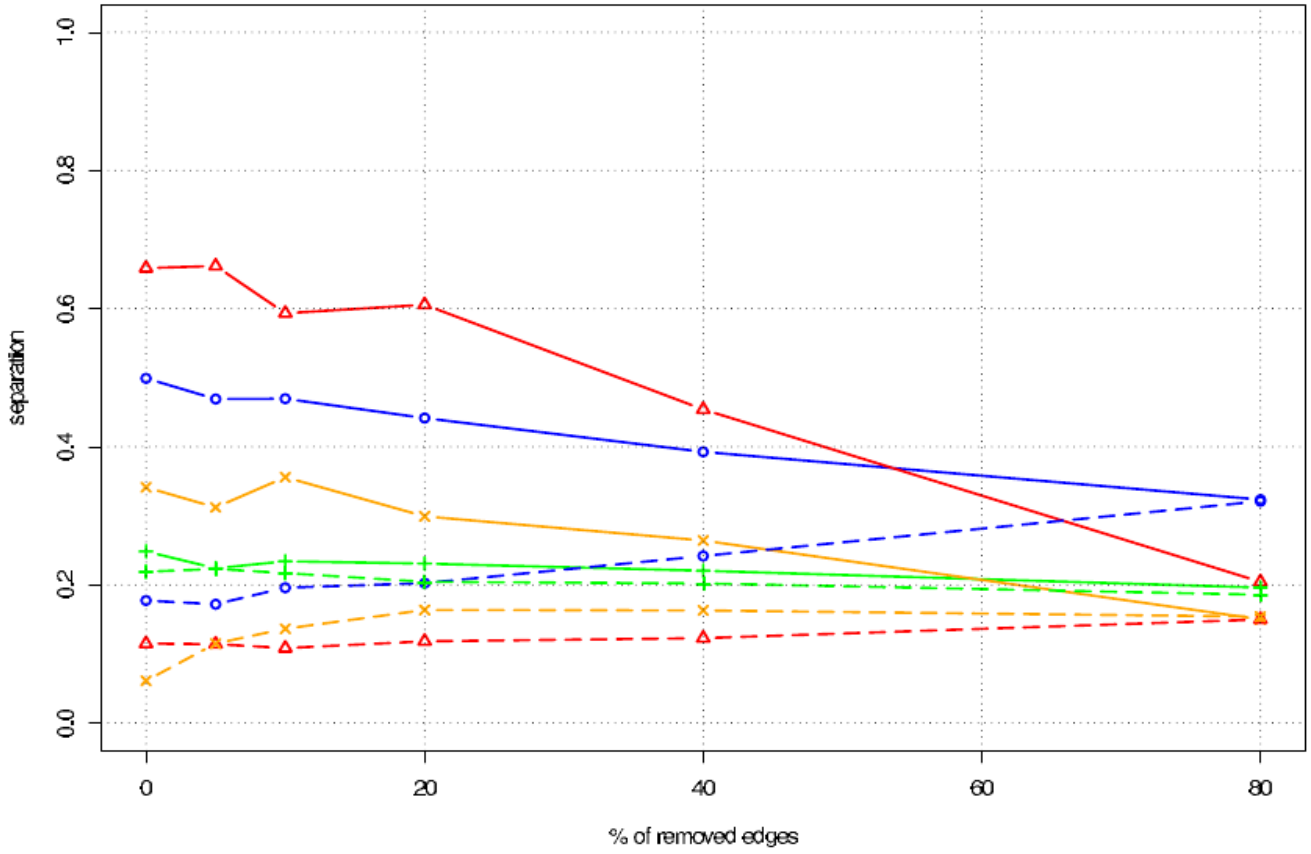


Figure: blau: MCL, rot: RNSC, orange: MCODE, grün: SPC, Bild: [2]

High-Throughput Daten

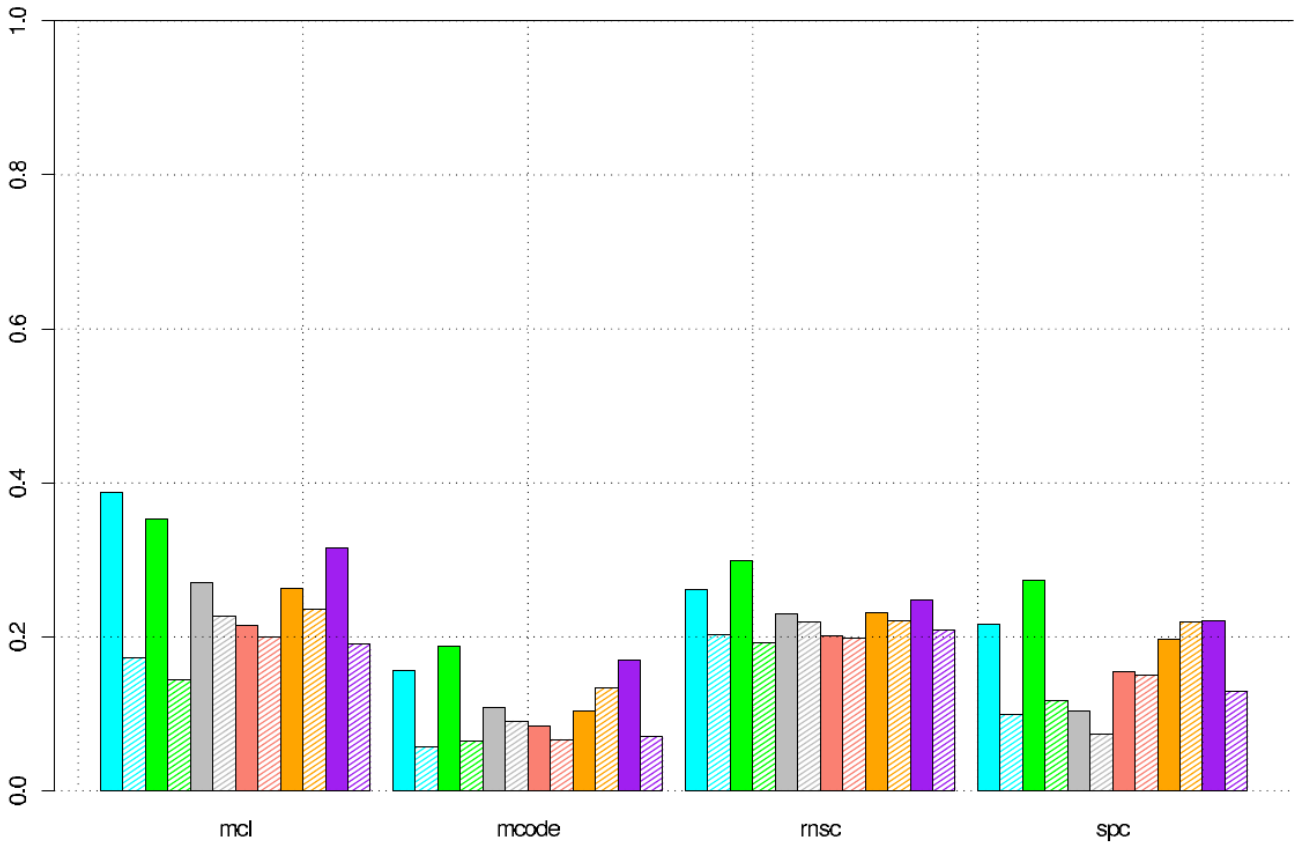


Figure: Bild: [2]

Anderer Ansatz

- Vorherige Vergleiche haben Cluster von RNSC direkt übernommen
- RNSC ist allgemeiner Cluster-Algorithmus, nicht konkret an Problemstellung von PPI angepasst
- Ansatz: Filtere resultierende Cluster nach Kriterien die auf Proteinkomplexe zutreffen
  - Clustergröße
  - Clusterdichte
  - Funktionelle Gleichheit der Proteine

Conclusion

- Leicht implementierbarer Algorithmus
- Erzeugt gute Ergebnisse bezüglich Kostenfunktion
- Berechnungsaufwand ist hoch

Literatur I

[1] Andrew Douglas King.
Graph clustering with restricted neighbourhood search.
Master's thesis, University of Toronto, 2004.

[2] Sylvain Brohée and Jacques van Helden.
Evaluation of clustering algorithms for protein-protein interaction networks.
BMC Bioinformatics, 2006.

Literatur II

[3] A. D. King, N. Przulj, and I. Jurisica.
Supplementary information for protein complex prediction via cost-based clustering: The restricted neighbourhood search clustering algorithm, April 2004.

[4] A. D. King, N. Przulj, and I. Jurisica.
Protein complex prediction via cost-based clustering, April 2004.