

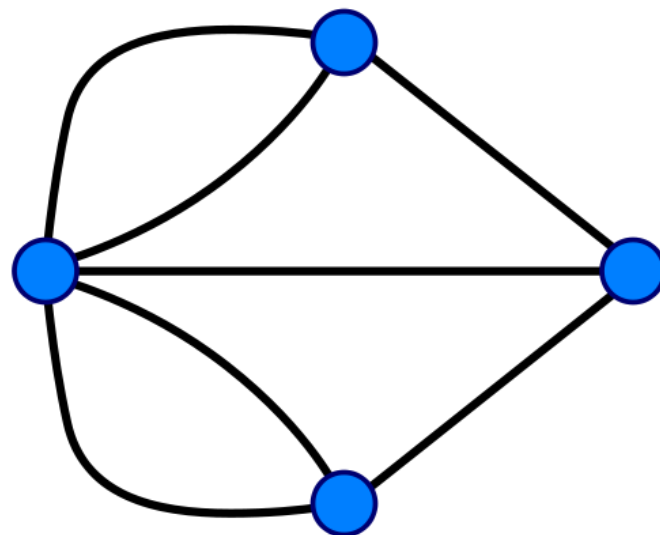
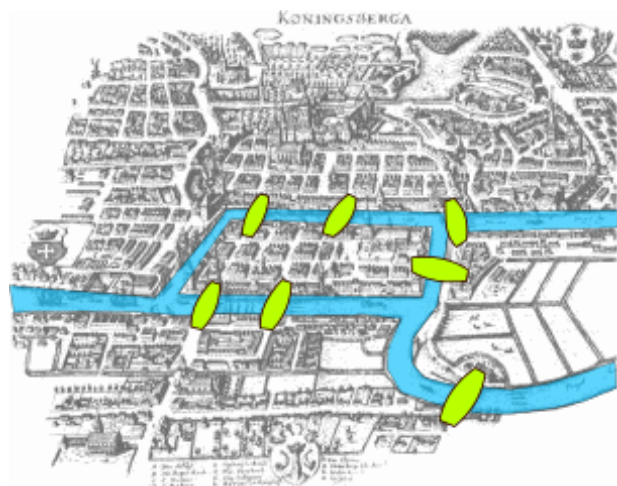


Community Detection in Graphs

Srinivasan Parthasarathy



Graphs from the Real World

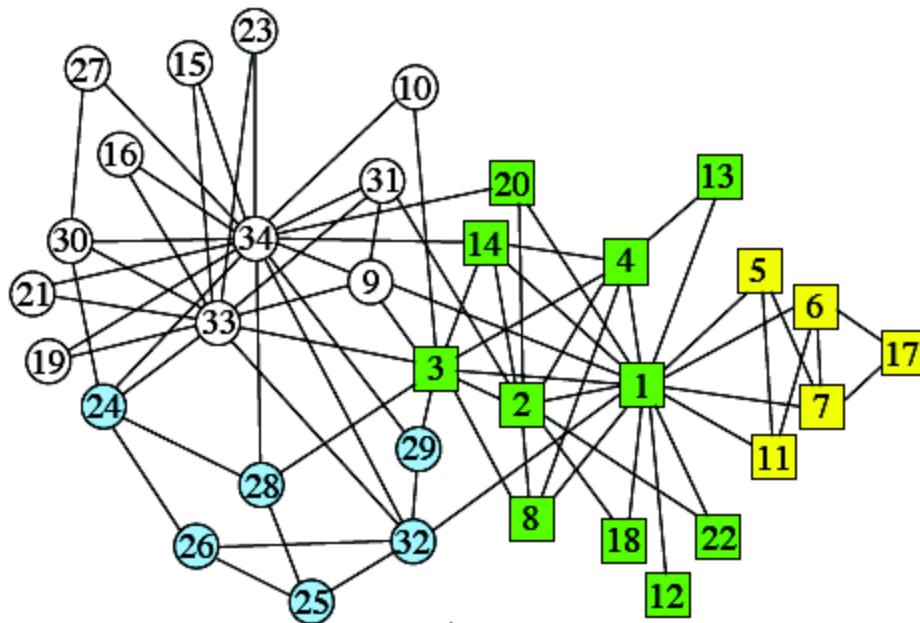


Königsberg's Bridges

Ref: http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

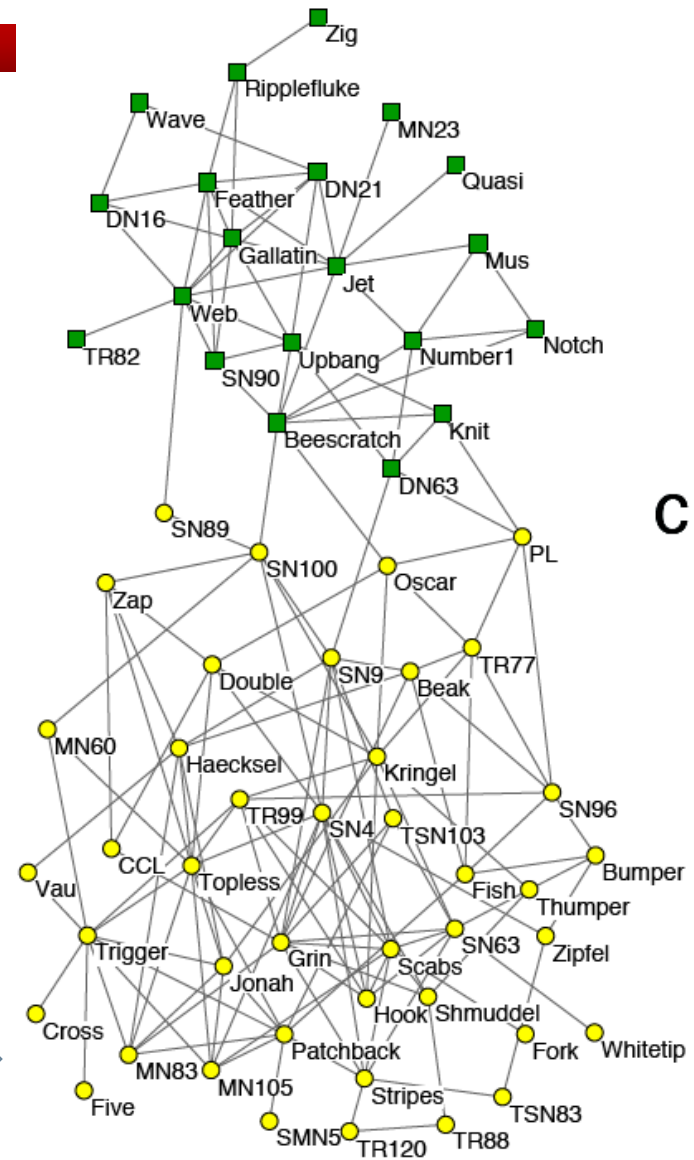


Graphs from the Real World



Zachary's Karate Club

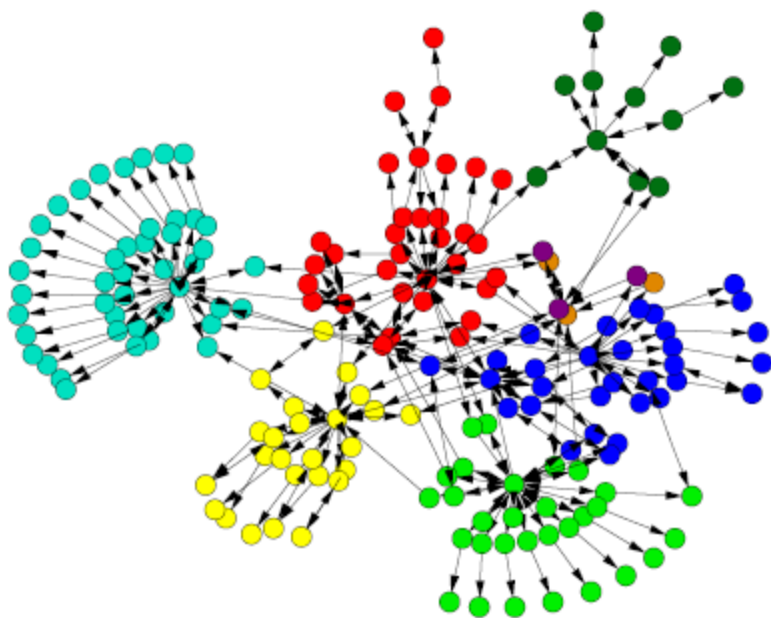
Lusseau's network of bottlenose dolphins



C



Graphs from the Real World

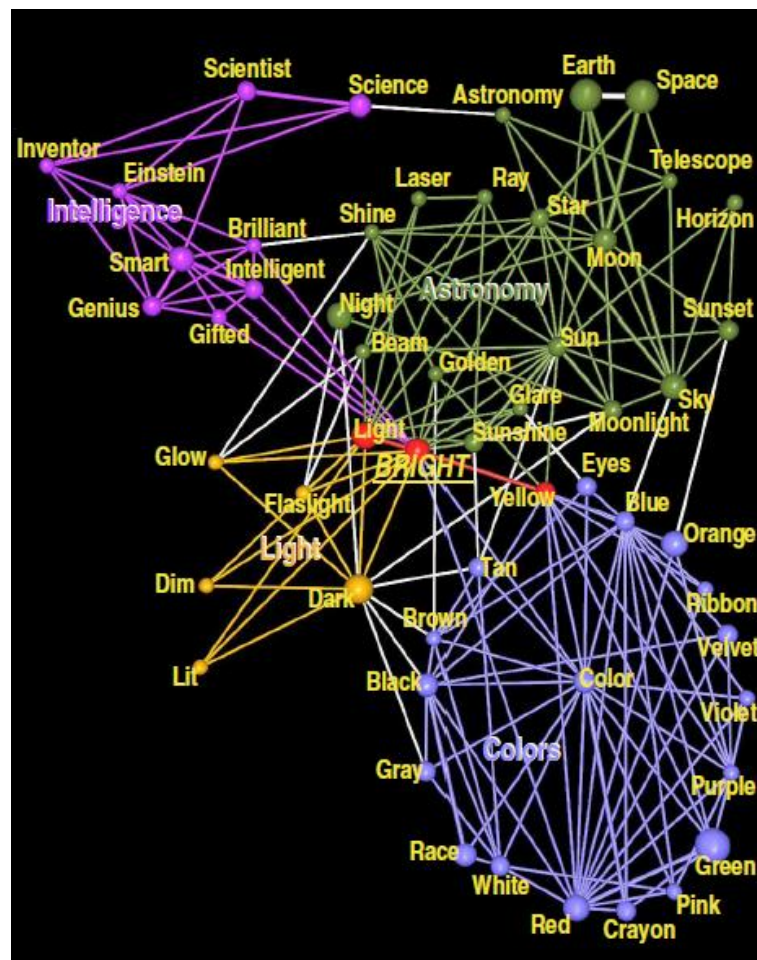


Webpage Hyperlink Graph

Directed Communities

Network of Word Associations

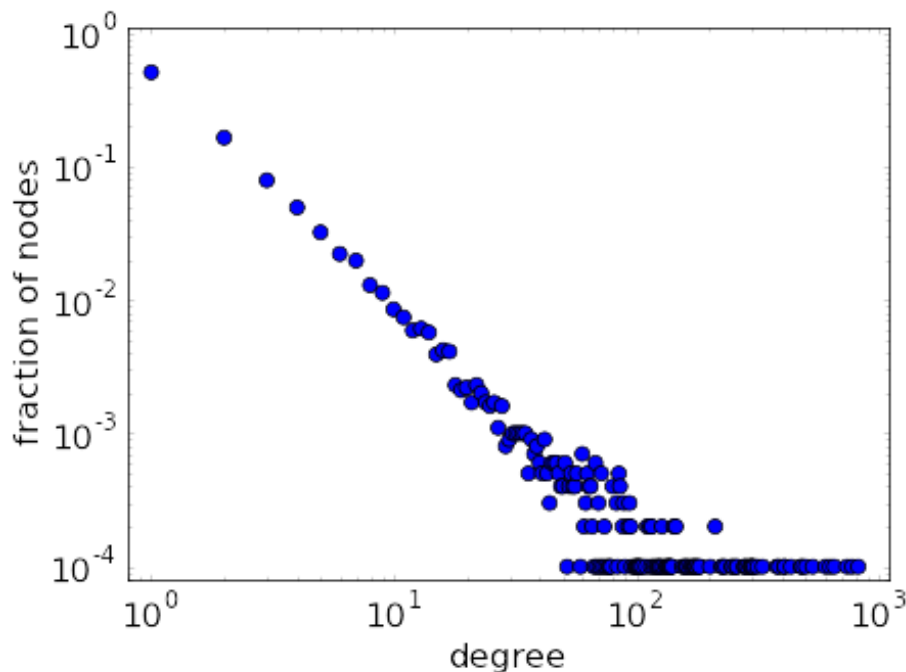
Overlapping Communities





Real Networks Are Not Random

- ❑ Degree distribution is broad, and often has a tail following power-law distribution



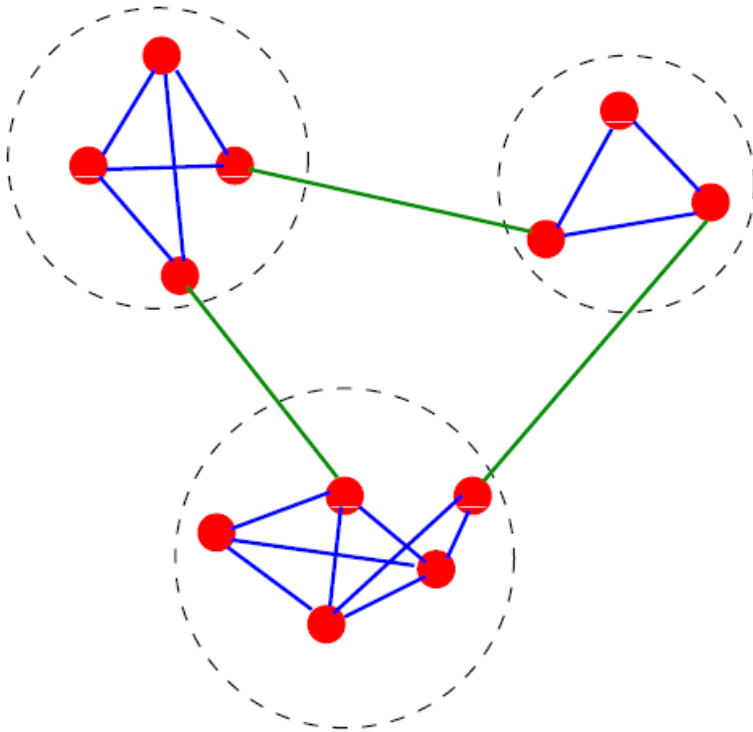
$$P(d) = cd^a$$
$$\Rightarrow \log(P(d)) = \log(c) + a \log(d)$$

Ref: "Plot of power-law degree distribution on log-log scale." From Math Insight.
http://mathinsight.org/image/power_law_degree_distribution_scatter



Real Networks Are Not Random

- ❑ Edge distribution is locally inhomogeneous



Community Structure!



Applications of Community Detection

- ❑ Website mirror server assignment
- ❑ Recommendation system
- ❑ Social network role detection
- ❑ Functional module in biological networks
- ❑ Graph coarsening and summarization
- ❑ Network hierarchy inference



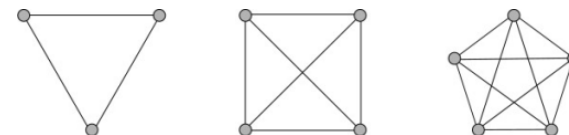
General Challenges

- ❑ Structural clusters can only be identified if graphs are sparse (i.e. $m = O(n)$)
 - ❑ Motivation for graph sampling/sparsification
- ❑ Many clustering problems are **NP**-hard. Even polynomial time approaches may be too expensive
 - ❑ Call for scalable solutions
- ❑ Concepts of “cluster”, “community” are not quantitatively well defined
 - ❑ Discussed in more details below

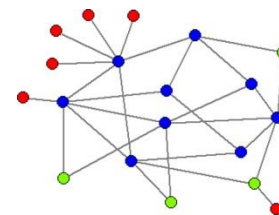


Defining Motifs (Micro communities)

- ❑ Local definitions: focus on the subgraph only
 - ❑ Clique: Vertices are all adjacent to each other
 - ❑ Strict definition, NP-complete problem
 - ❑ *n-clique, n-clan, n-club, k-plex*



- ❑ k-core: Maximal subgraph that each vertex is adjacent to at least k other vertices in the subgraph



- ❑ More advanced motifs – graphlets, k-truss etc.



Evaluating Community Quality

- ❑ So we can compare the “goodness” of extracted communities, whether extracted by different algorithms or the same.
 - ❑ We know about Modularity but there are others!
- ❑ Define $a(A, B) = \sum_{i \in A} \sum_{j \in B} a_{ij}$
- ❑ Normalized cut (n-cut): $ncut_C = \frac{a(C, \bar{C})}{a(C, V)}$
- ❑ Conductance: $\phi_C = \frac{a(C, \bar{C})}{\min\{a(C, V), a(\bar{C}, V)\}}$



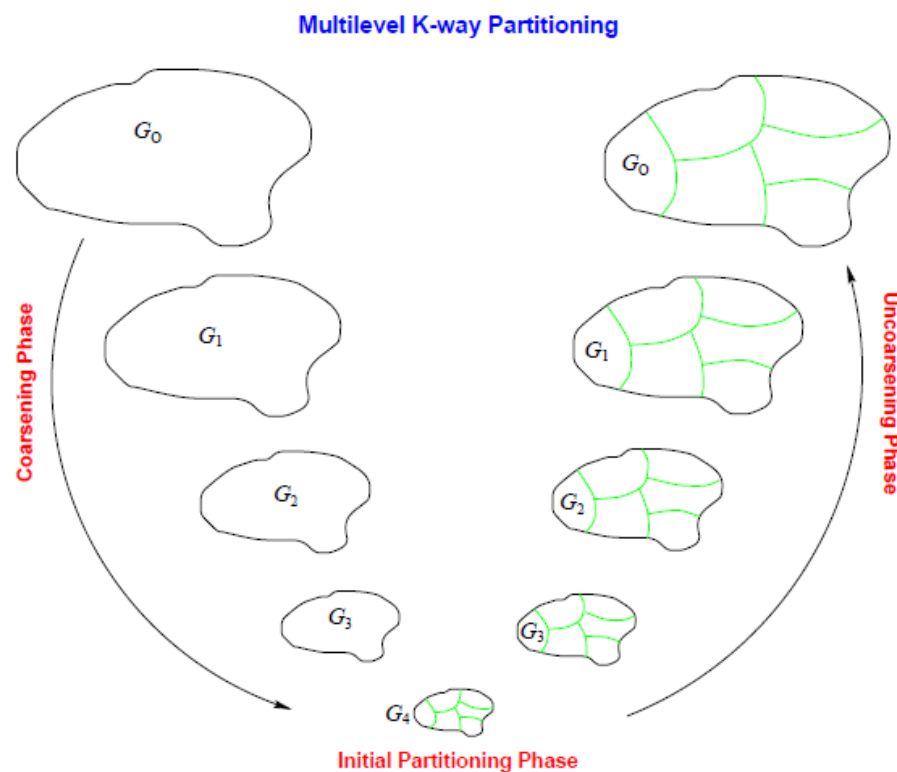
Traditional Methods

- ❑ Graph Partitioning
 - ❑ Dividing vertices into *groups of predefined size*
- ❑ Kernighan-Lin algorithm
 - ❑ Create initial bisection
 - ❑ Iteratively swap subsets containing equal number of vertices
 - ❑ Select the partition that maximize (number of edges insider modules – cut size)



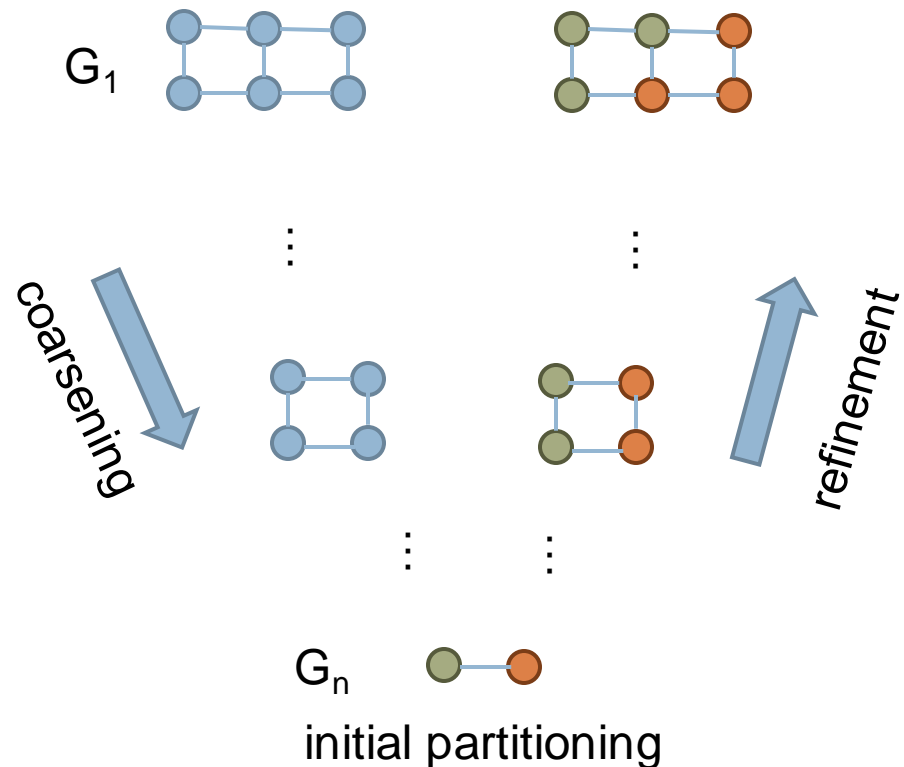
Traditional Methods (Sec. 4)

- ❑ Graph Partitioning
 - ❑ METIS (Karypis and Kumar)
 - ❑ Multi-level approach
 - ❑ Coarsen the graph into skeleton
 - ❑ Perform K-L and other heuristics on the skeleton
 - ❑ Project back with local refinement





- ❑ Multilevel
 - ❑ Use short range and long range structure
- ❑ 3 major phases
 - ❑ coarsening
 - ❑ initial partitioning
 - ❑ refinement

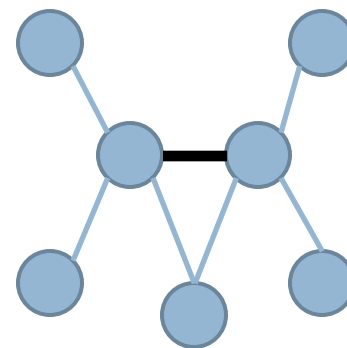
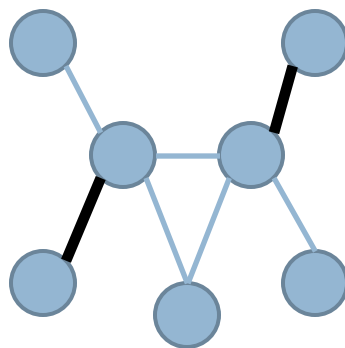
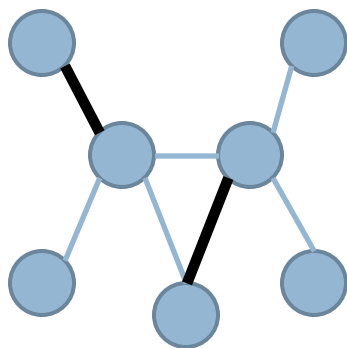




Coarsening

14

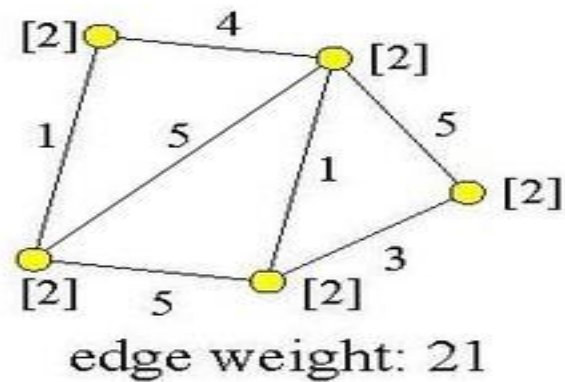
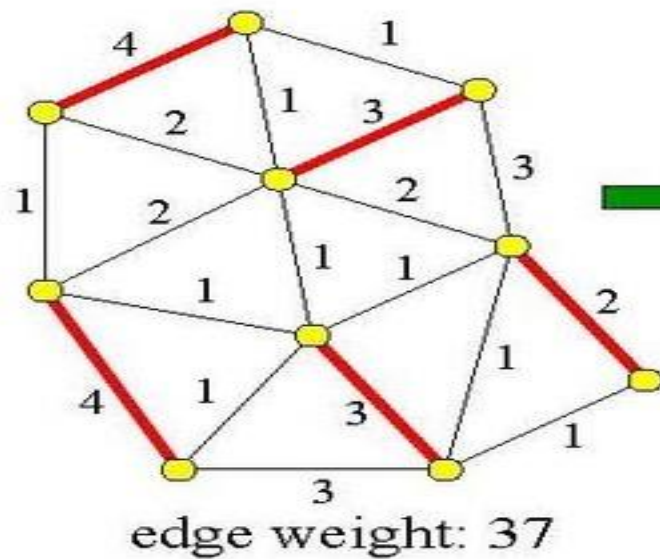
- ❑ Find matching
 - ❑ related problems:
 - ❑ maximum (weighted) matching ($O(V^{1/2}E)$)
 - ❑ minimum maximal matching (NP-hard), i.e., matching with smallest #edges
 - ❑ polynomial 2-approximations





HEM: Example

Heavy-edge Matching

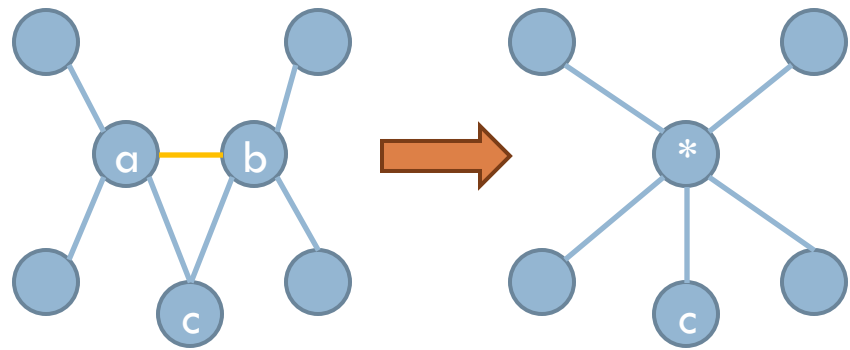




Coarsening

16

□ Edge contract

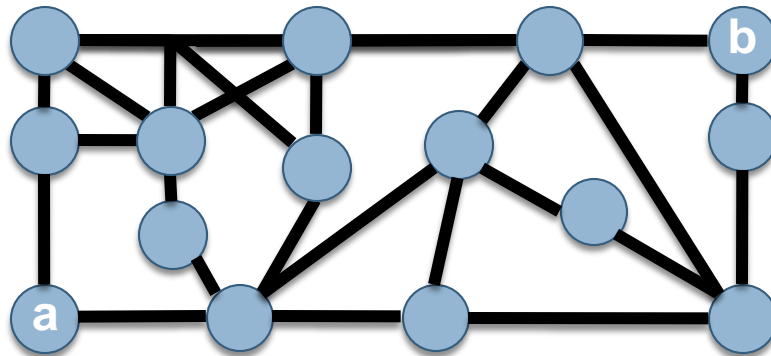




Initial Partitioning

17

- ❑ Breadth-first traversal
 - ❑ select k random nodes



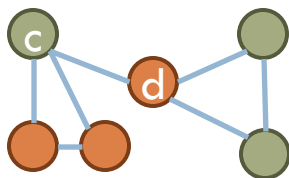


Initial Partitioning

18

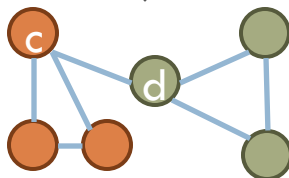
□ Kernighan-Lin

- improve partitioning by greedy swaps



$$D_c = E_c - I_c = 3 - 0 = 3$$

$$D_d = E_d - I_d = 3 - 0 = 3$$



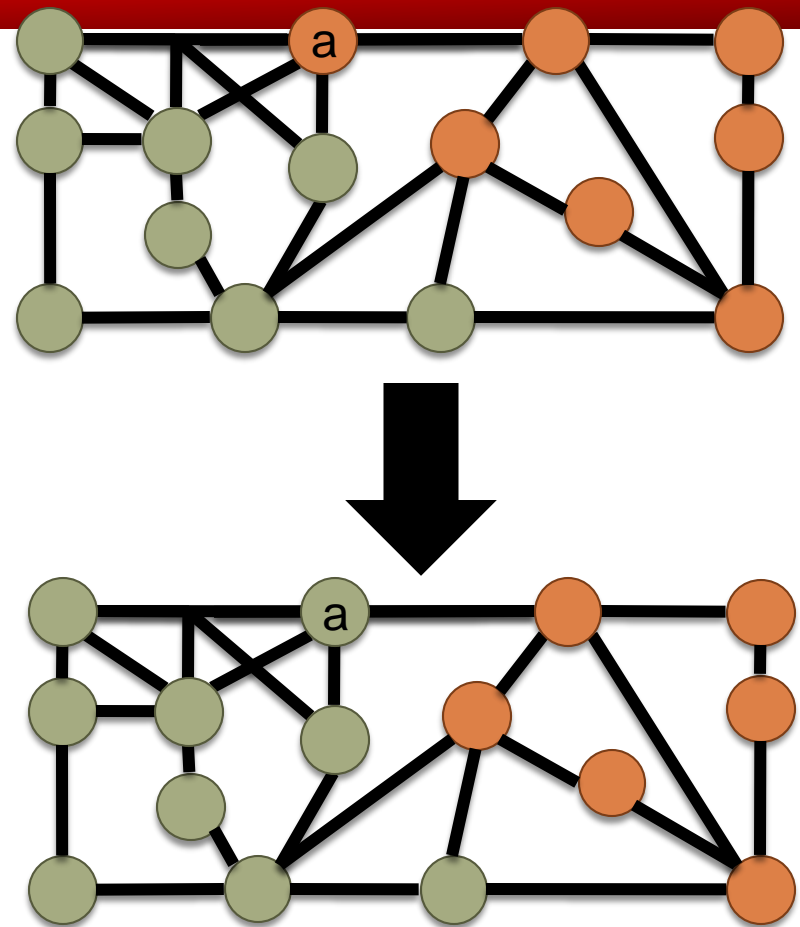
$$\text{Benefit}(\text{swap}(c, d)) = D_c + D_d - 2A_{cd} = 3 + 3 - 2 = 4$$



Refinement

19

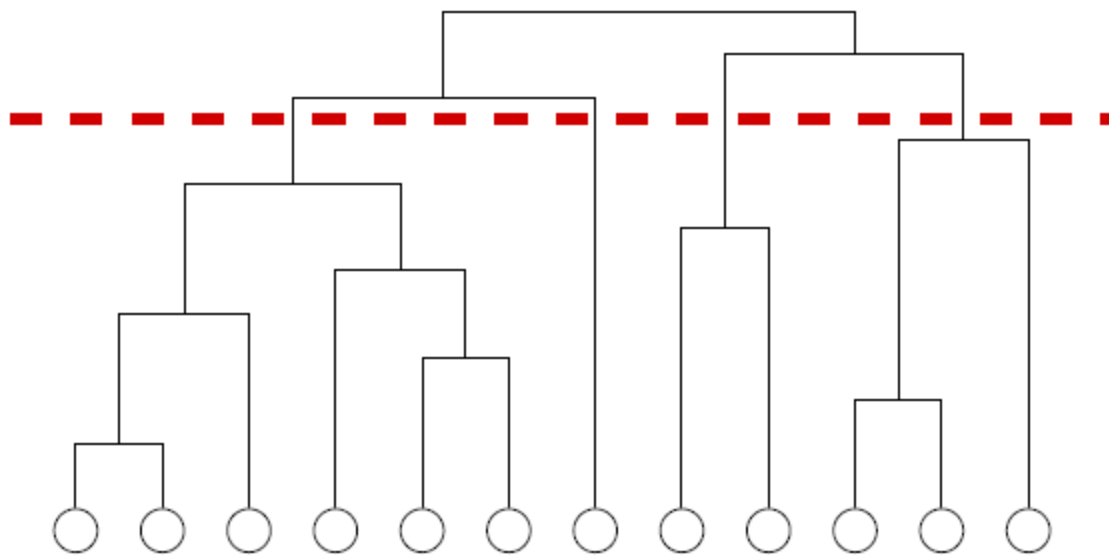
- ❑ Random K-way refinement
 - ❑ Randomly pick boundary node
 - ❑ Find new partition which reduces graph cut and maintains balance
 - ❑ Repeat until all boundary nodes have been visited





Hierarchical Clustering

- ❑ Hierarchical Clustering
 - ❑ Graphs may have hierarchical structure
 - ❑ *Embed vertices in a metric space and then cluster*





Hierarchical Clustering

- ❑ Hierarchical Clustering
 - ❑ Find clusters using a similarity matrix
 - ❑ Agglomerative: clusters are iteratively merged if their similarity is sufficiently high
 - ❑ Divisive: clusters are iteratively split by removing edges with low similarity
 - ❑ Define similarity between clusters
 - ❑ Single linkage (minimum element)
 - ❑ Complete linkage (maximum element)
 - ❑ Average linkage
- ❑ Drawback: dependent on similarity threshold



Other Methods

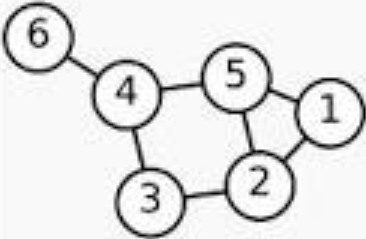
- ❑ *Partitional Clustering*
 - ❑ *Embed vertices in a metric space, and find clustering that optimizes the cost function*
 - ❑ *Minimum k -clustering*
 - ❑ *k -clustering sum*
 - ❑ *k -center*
 - ❑ *k -median*
 - ❑ *k -means*
 - ❑ *Fuzzy k -means*
 - ❑ *DBSCAN*



SPECTRAL CLUSTERING

- Spectral Clustering

- Un-normalized Laplacian: $L = D - A$

Labeled graph	Laplacian matrix
	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

- # of connected components = # of 0 eigenvalues

- Normalized variants:

- $L_{sym} = D^{-1/2} L D^{-1/2}$

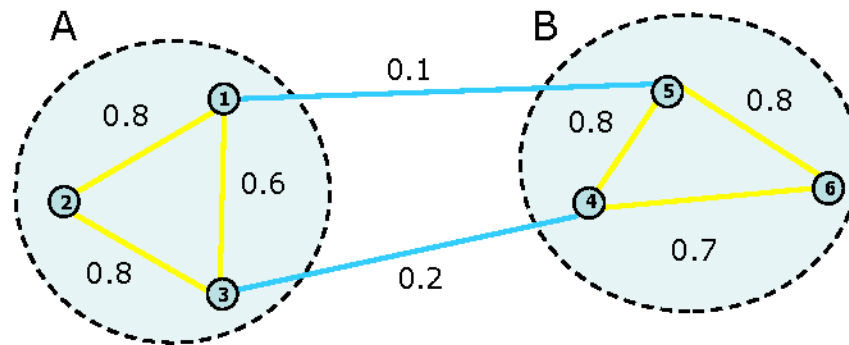
- $L_{rw} = D^{-1} L = I - D^{-1} A$



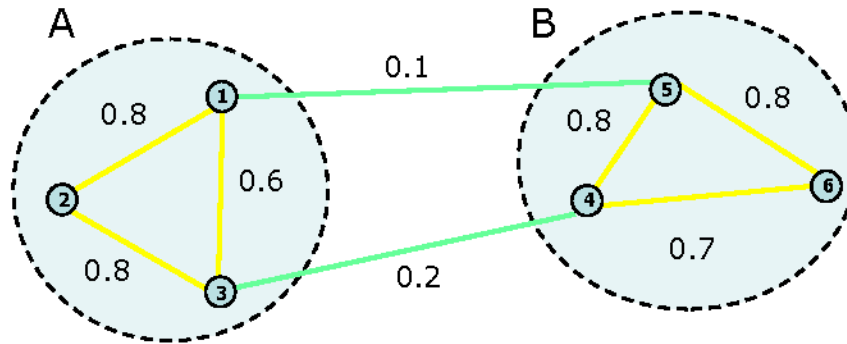
Spectral Clustering

- ❑ Spectral Clustering
 - ❑ Compute the Laplacian matrix
 - ❑ Transform graph vertices into points where coordinates are elements of eigenvectors
 - ❑ Cluster properties become more evident
 - ❑ Cluster vertices in the new metric space
 - ❑ Complexity
 - ❑ $O(n^3)$
 - ❑

Illustrative example



Graph and similarity matrix



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0.8	0.6	0	0.1	0
x_2	0.8	0	0.8	0	0	0
x_3	0.6	0.8	0	0.2	0	0
x_4	0.8	0	0.2	0	0.8	0.7
x_5	0.1	0	0	0.8	0	0.8
x_6	0	0	0	0.7	0.8	0

Illustrative example

Pre-processing

Build Laplacian matrix L of the graph



x_1	1.5	-0.8	-0.8	0	-0.1	0
x_2	-0.8	1.8	-0.8	0	0	0
x_3	-0.8	-0.8	1.8	-0.2	0	0
x_4	-0.8	0	-0.2	2.5	-0.8	-0.7
x_5	-0.1	0	0	0.8	1.7	-0.8
x_6	0	0	0	-0.7	-0.8	1.5

Decomposition : Find

- eigenvalues λ and
- eigenvectors X of matrix L
- Map vertices to the corresponding components of 2nd eigenvector

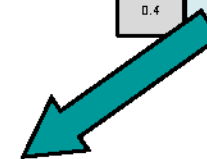


$\lambda =$

0.0
0.3
2.2
2.3
2.5
3.0

$X =$

0.4	0.2	0.1	0.4	-0.2	-0.9
0.4	0.2	0.1	-0.1	0.4	0.3
0.4	0.2	-0.2	0.0	-0.2	0.8
0.4	-0.4	0.9	0.2	-0.4	-0.8
0.4	-0.7	-0.4	-0.8	-0.8	-0.2
0.4	-0.7	-0.2	0.5	0.8	0.9



x_1	0.2
x_2	0.2
x_3	0.2
x_4	-0.4
x_5	-0.7
x_6	-0.7

How do we find the clusters?

Spectral Clustering Algorithms (continued)



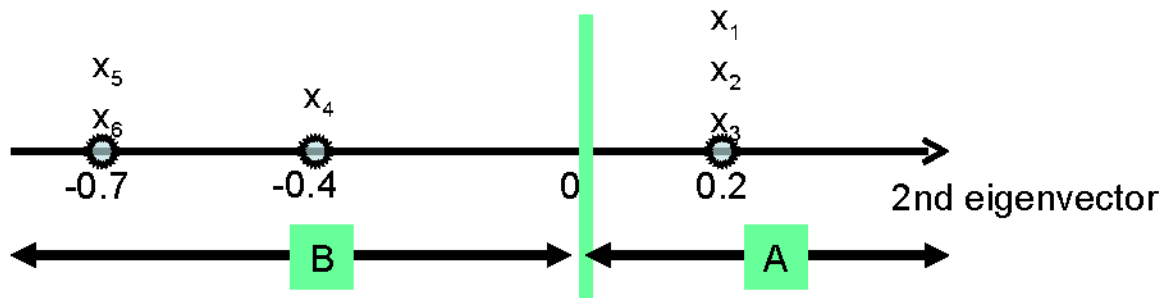
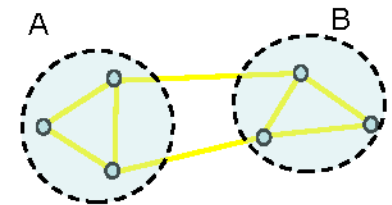
x_1	0.2
x_2	0.2
x_3	0.2
x_4	-0.4
x_5	-0.7
x_6	-0.7



Split at value 0
Cluster A : Positive points
Cluster B : Negative points

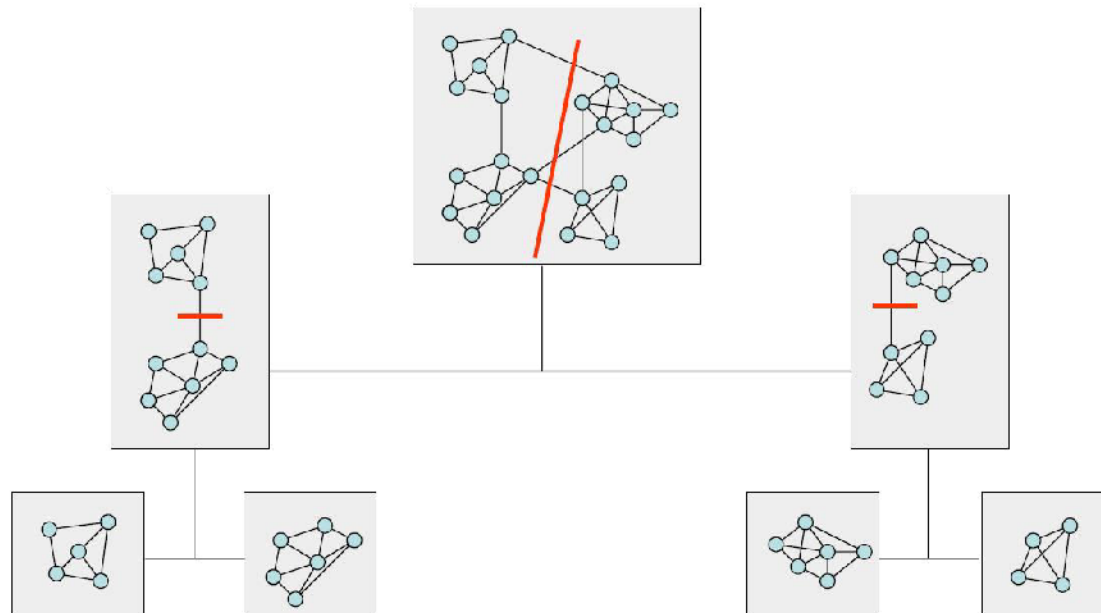
x_1	0.2
x_2	0.2
x_3	0.2

x_4	-0.4
x_5	-0.7
x_6	-0.7



Recursive bi-partitioning

- Recursively apply bi-partitioning algorithm in a hierarchical divisive manner.
- Disadvantages: Inefficient, unstable





Alternative Approach

- ❑ Compute Laplacian as before
- ❑ Compute representation of each point to low rank representation by selecting several eigenvectors up to the desired inertia
 - ❑ Inertia \leftrightarrow Total variance explained by low rank approximation
- ❑ Run k-means or any clustering algorithm on representation

Spectral clustering stages

- **Pre-processing**
 - Construct the graph and the similarity matrix representing the dataset.
- **Spectral representation**
 - Form the associated Laplacian matrix
 - Compute eigenvalues and eigenvectors of the Laplacian matrix.
 - Map each point to a lower-dimensional representation based on one or more eigenvectors.
- **Clustering**
 - Assign points to two or more classes, based on the new representation.



Variants of Betweenness

- ❑ Girvan and Newman's edge centrality algorithm: Iteratively remove edges with high centrality and re-compute the values
- ❑ Define edge centrality:
 - ❑ Edge betweenness: number of all-pair shortest paths that run along an edge
 - ❑ Random-walk betweenness: probability of random walker passing the edge
 - ❑ Current-flow betweenness: current passing the edge in a unit resistance network
- ❑

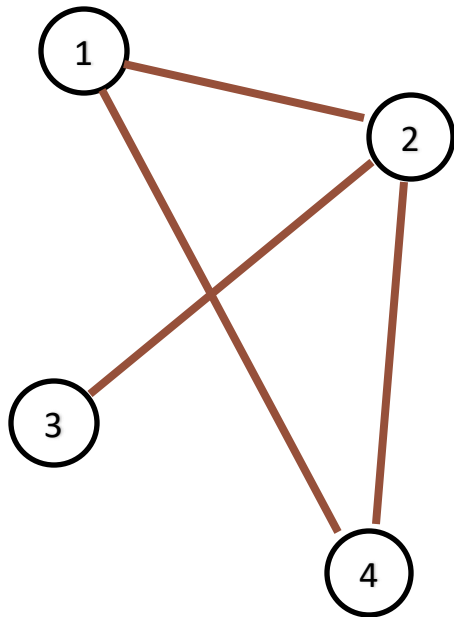


Random Walk Based Approaches

- ❑ A random walker spends a long time inside a community due to the high density of internal edges
- ❑ E.g. 1 : Zhou used random walks to define a distance between pairs of vertices
- ❑ the distance between i and j is the average number of edges that a random walker has to cross to reach j starting from i .

Stochastic (Flow) Matrix: A matrix where each column sums to 1.

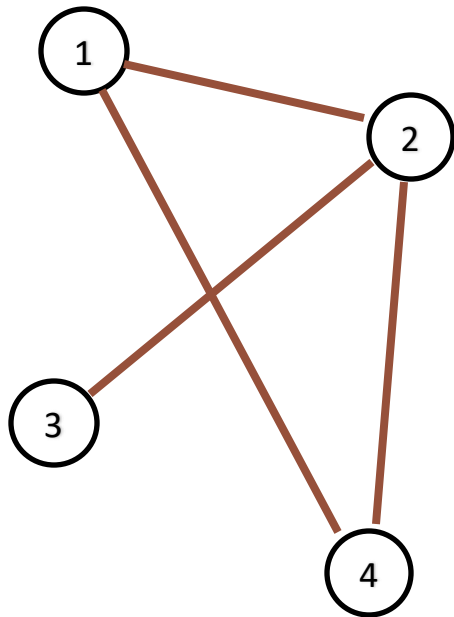
Stochastic Flow: An entry in a stochastic matrix, interpreted as the “flow” or “transition probability”.



	1	2	3	4
1		0.33		0.5
2	0.5		1.0	0.5
3		0.33		
4	0.5	0.33		

Stochastic (Flow) Matrix: A matrix where each column sums to 1.

Stochastic Flow: An entry in a stochastic matrix, interpreted as the “flow” or “transition probability”.

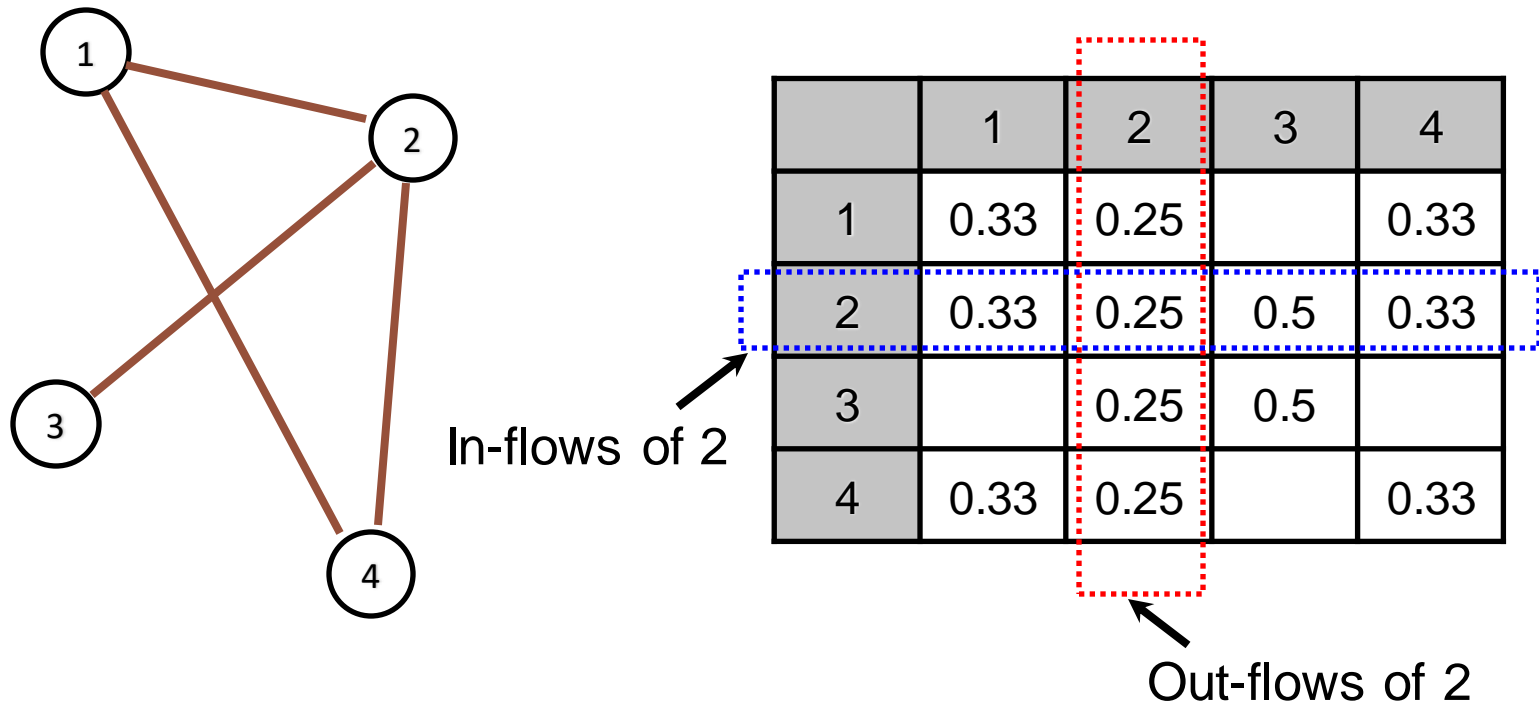


	1	2	3	4
1		0.33		0.5
2	0.5		1.0	0.5
3		0.33		
4	0.5	0.33		

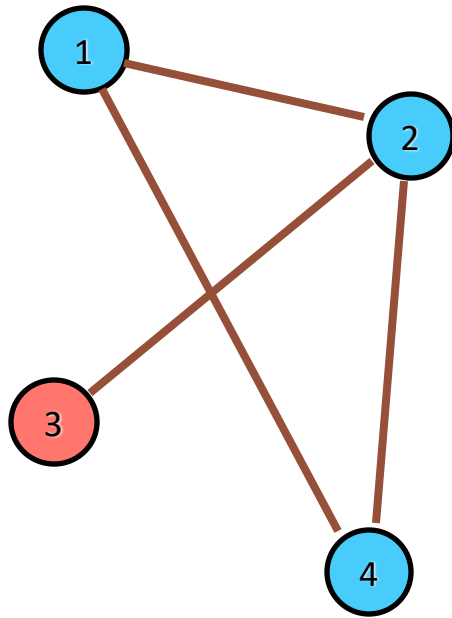
Flow from 2 to 3

Stochastic (Flow) Matrix: A matrix where each column sums to 1.

Stochastic Flow: An entry in a stochastic matrix, interpreted as the “flow” or “transition probability”.



Repeatedly apply certain operations to the flow matrix until the matrix converges and can be interpreted as a clustering.



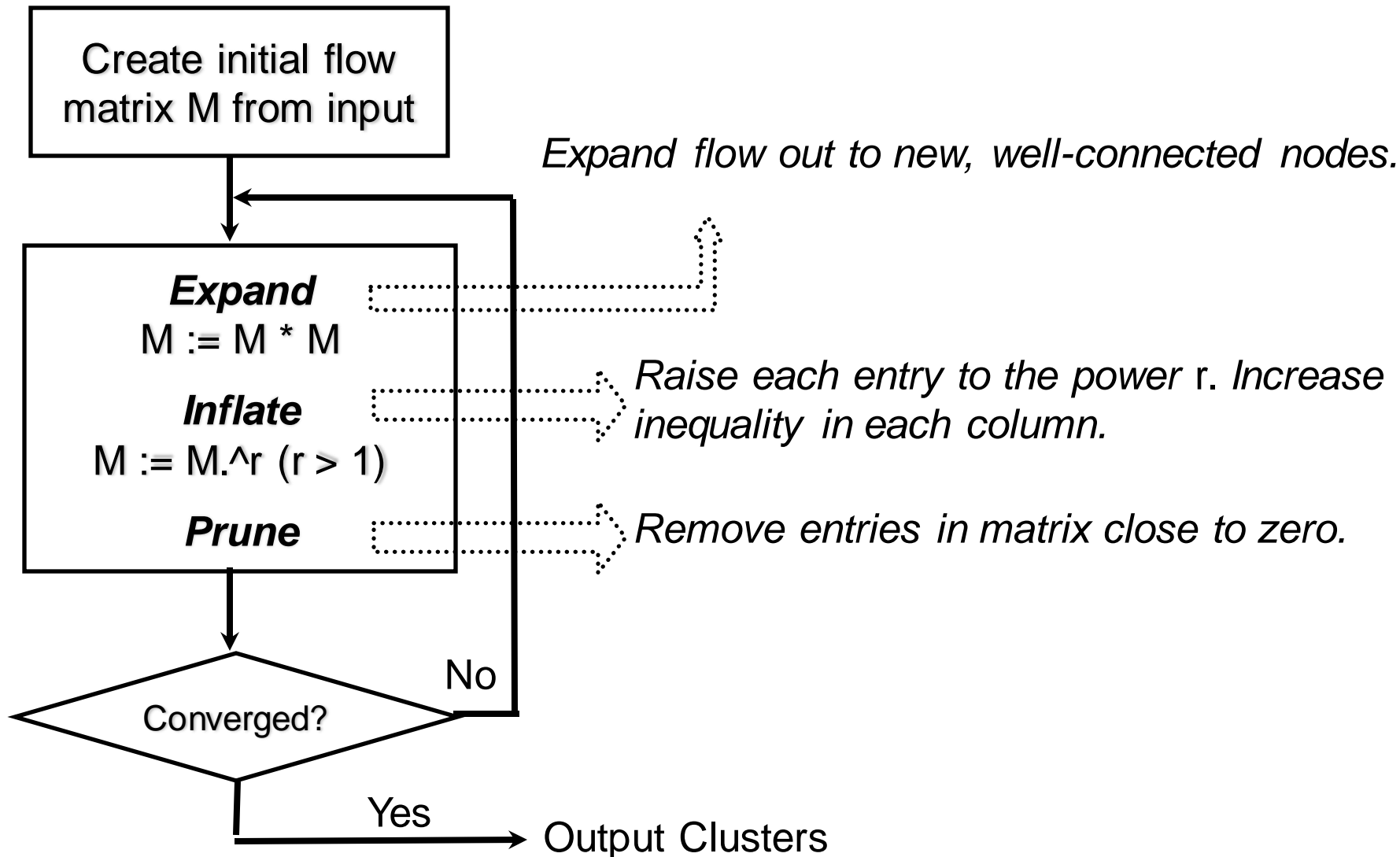
	1	2	3	4
1				
2	1.0	1.0		1.0
3			1.0	
4				

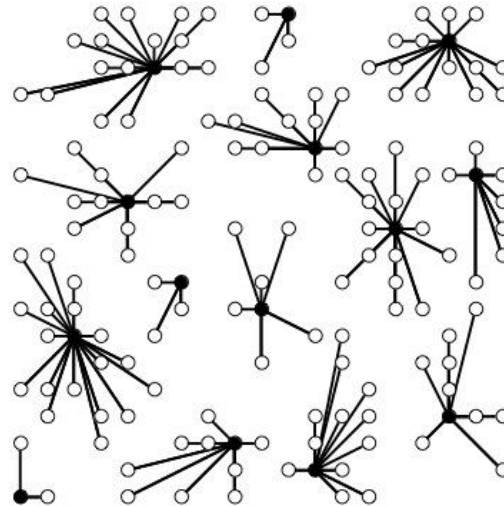
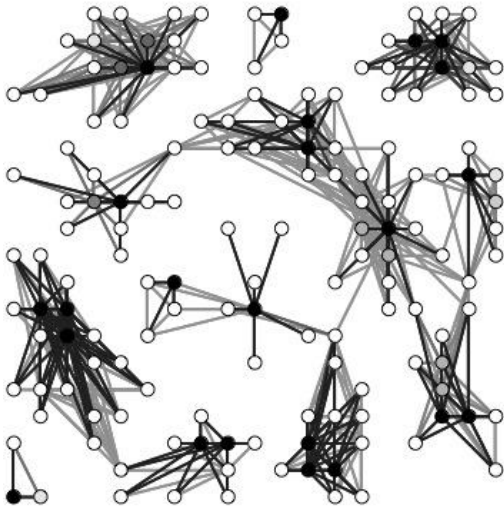
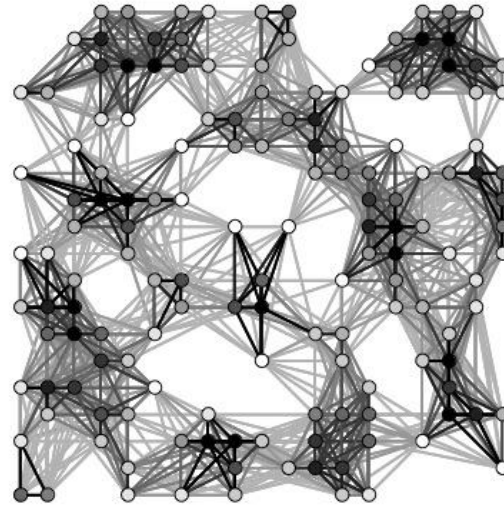
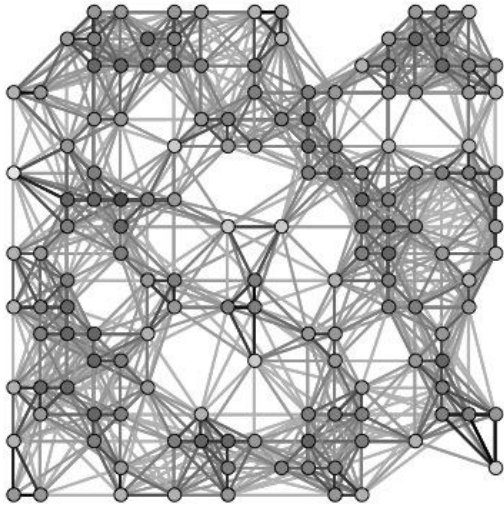
Markov Clustering (MCL)

Stijn van Dongen, 2000

*The original Stochastic flow
clustering algorithm*

The MCL algorithm





[van Dongen '00]

MCL Flaws

1. Outputs many small clusters.
2. Does not scale well.

[Chakrabarti and Faloutsos '06]

MCL Flaws

1. Outputs many small clusters.

Fix: Regularized MCL

2. Does not scale well.

Fix: Multi-Level Regularized MCL

The *Regularize* operator

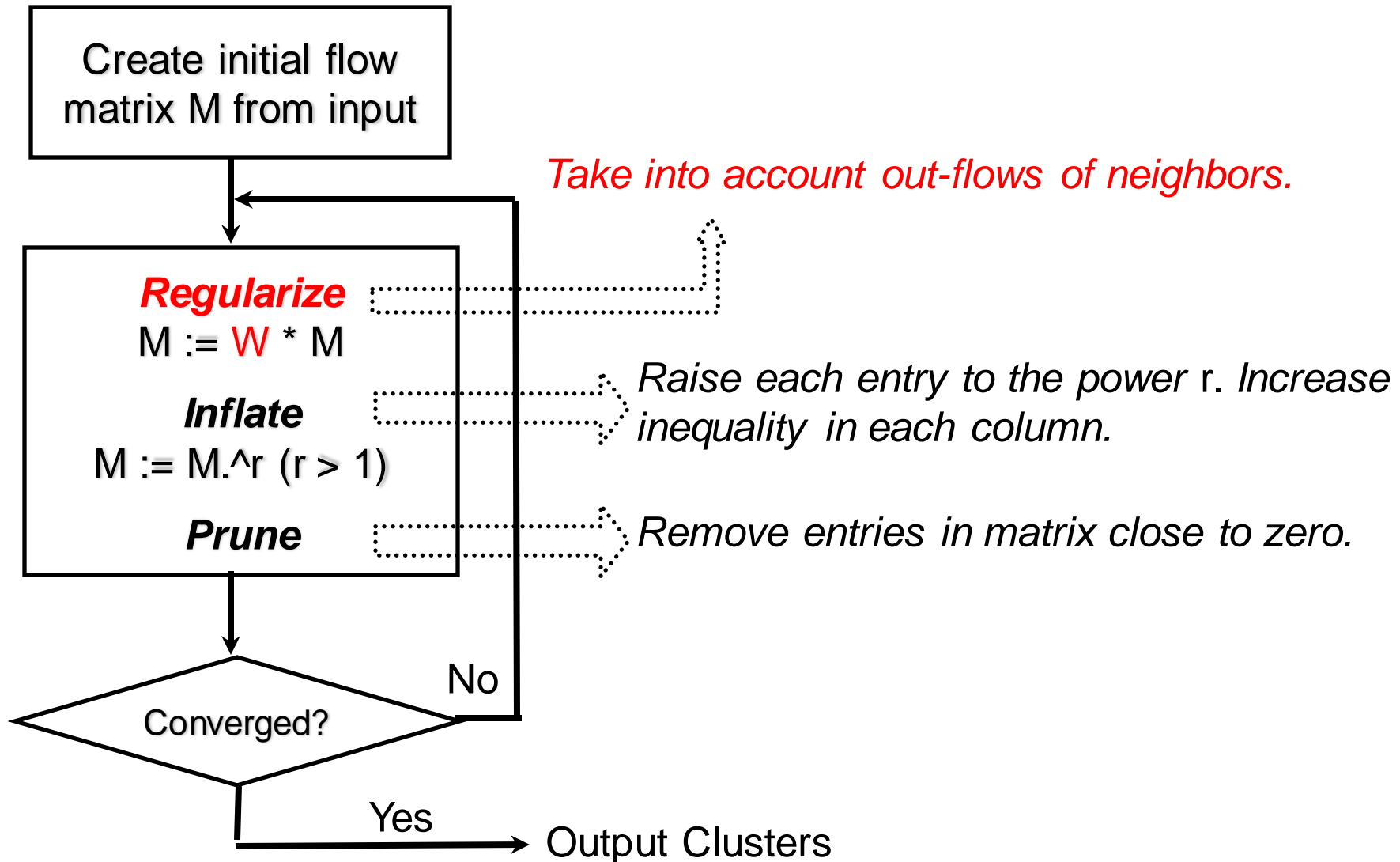
Key idea: Set the out-flows of a node so as to minimize “distance” from neighbors. Distance measured using KL-divergence.

Closed-form solution!

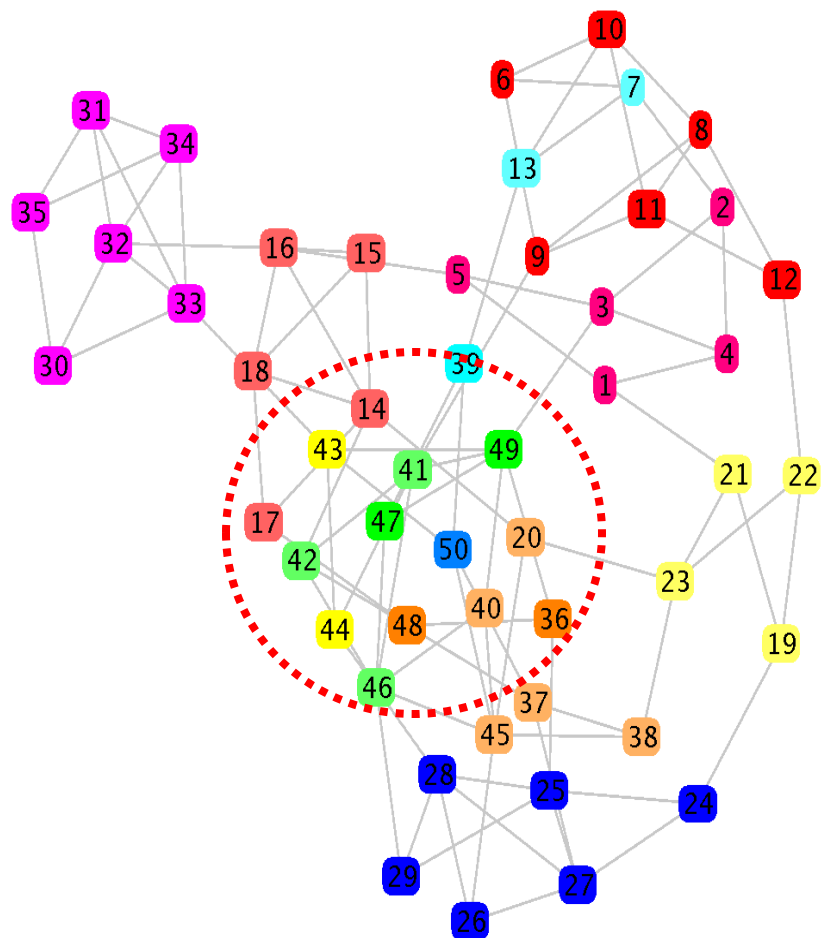
$$M(:, i) := \sum_{j \in \text{adj}(i)} \underset{\substack{\uparrow \\ \text{Weight of} \\ \text{neighbor } j}}{W(j, i)} * M(:, j)$$

In matrix notation, $M := W * M$

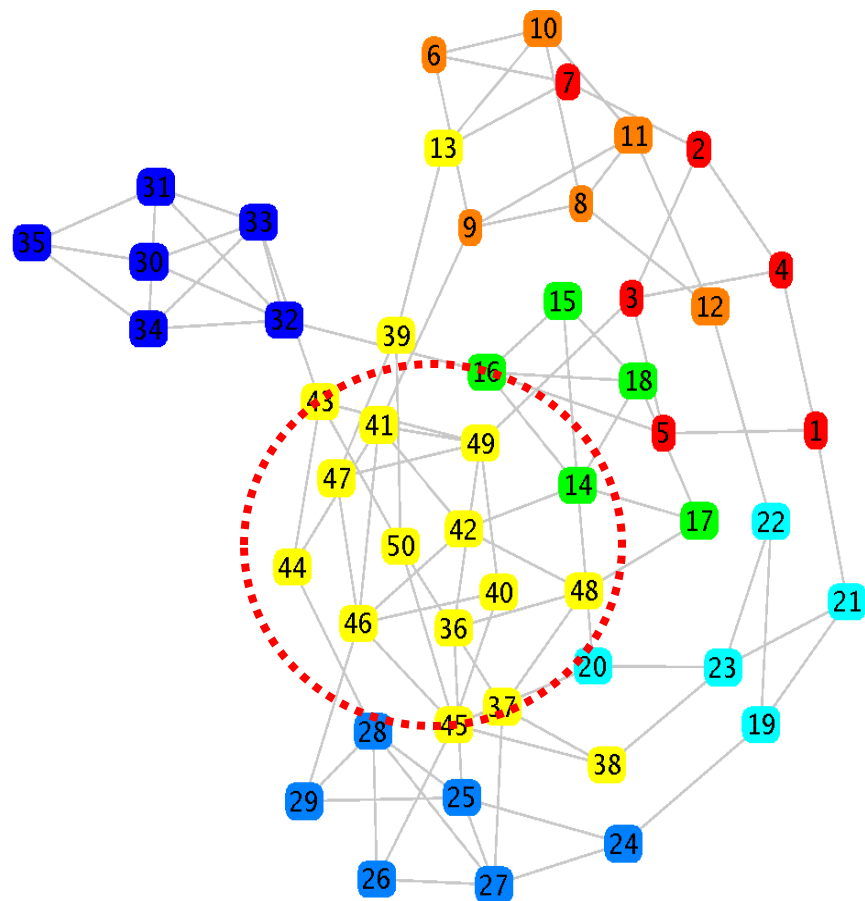
The **R**-MCL algorithm



MCL



R-MCL



[Automtically visualized using Prefuse]

Multi-Level Regularized MCL

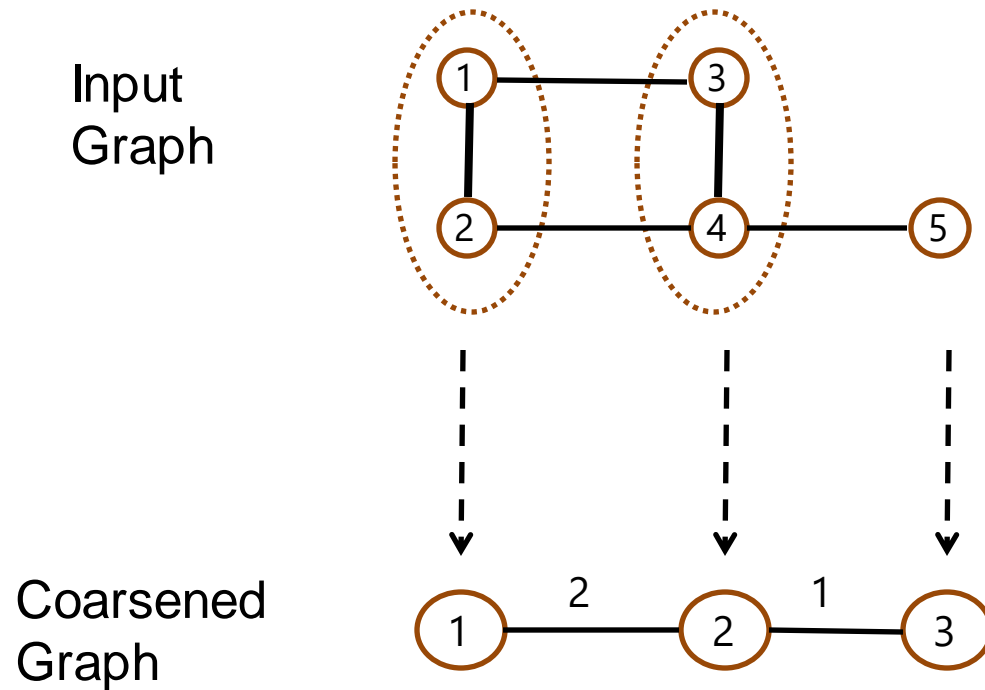
Making R-MCL fast

General idea of multi-level methods:

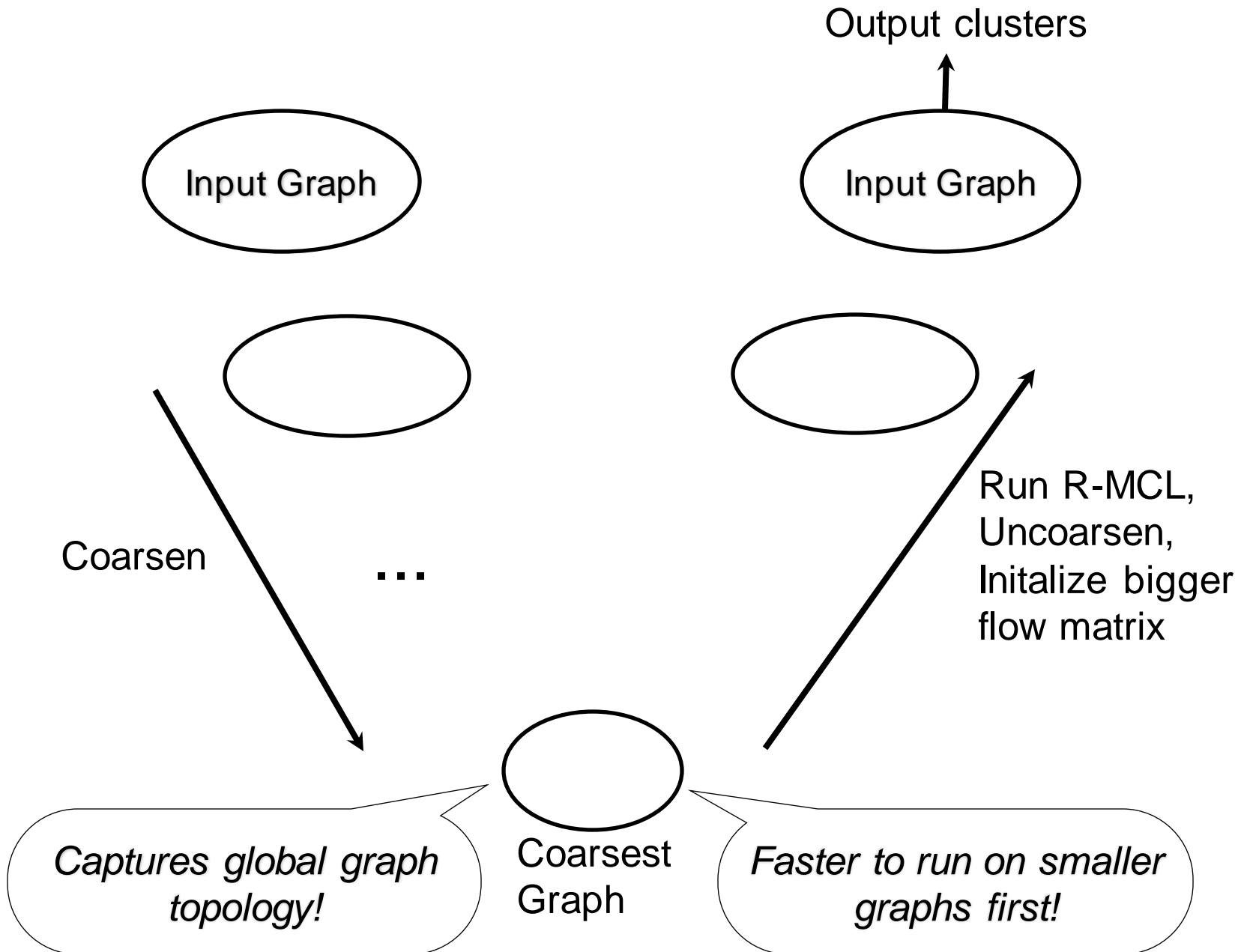
*Create smaller “replicas” of the original problem.
Solving the smaller problem should help us solve the
original problem.*

[Shang-hua Teng '97]

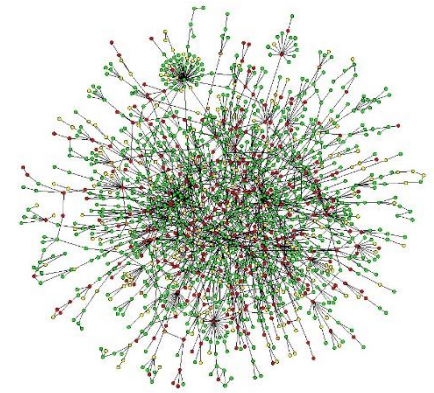
“Coarsening”: Creating smaller replicas



[Karypis and Kumar '98]



Comparison with MCL on Protein Interaction Networks



Dataset (n,m)	Quality Change	Speedup (Time)
Yeast (5k, 15k)	36%	2.5x (0.4s)
Yeast_Noisy (6k, 200k)	300%	57x (8s)
Human (10k, 60k)	21.6%	200x (2s)

[Hardware: Quad-core Intel i5 CPU, 3.2 GHz, with 16GB RAM]

Wikipedia article-article network

~1.1M nodes, ~53M edges



	Quality (Absolute)	Time (minutes)
MLR-MCL	20.2	132
Metis	12.3	125
Metis+MQI	19.2	592

Note: MCL and other methods timed-out or ran out of memory.

[Hardware: Quad-core Intel i5 CPU, 3.2 GHz, with 16GB RAM]

Real World Impact

“

FROM:



SUBJECT: Recognition of your MLR-MCL graph clustering software contribution

Your MLR-MCL software was extensively used for studying and promoting an important research on designing methods for automating large VLSI unit partitioning at Intel

....

It is very rare to find a public domain software with the highest standards of result quality and performance measures. As far as our tests went (and we did examine a large number of programs), MLR-MCL was the fastest program from its kind and fully reliable.

....

”

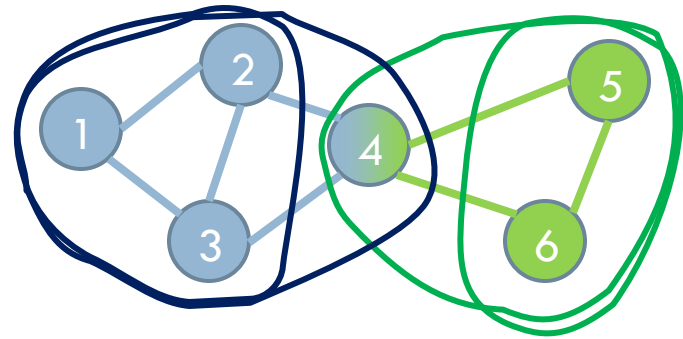
Overlapping community detection

- ❑ Most of previous methods can only generate non-overlapped clusters.
 - ❑ A node only belongs to one community.
 - ❑ Not real in many scenarios.
 - ❑ A person usually belongs to multiple communities.
- ❑ Most of current overlapping community detection algorithms can be categorized into three groups.
 - ❑ Mainly based on non-overlapping communities algorithms.

Overlapping community detection

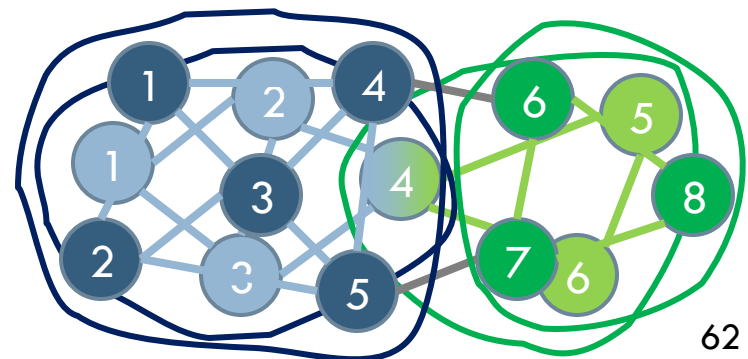
❑ 1. Identifying **bridge nodes**

- ❑ First, identifying bridge nodes and remove or duplicate these nodes.
 - ❑ Duplicate nodes have connection b/t them.
- ❑ Then, apply hard clustering algorithm.
 - ❑ If bridge nodes was removed, add them back.
- ❑ E.g. DECAFF [Li2007], Peacock [Gregory2009]
- ❑ Cons: Only a small part of nodes can be identified as bridge nodes.



Overlapping community detection

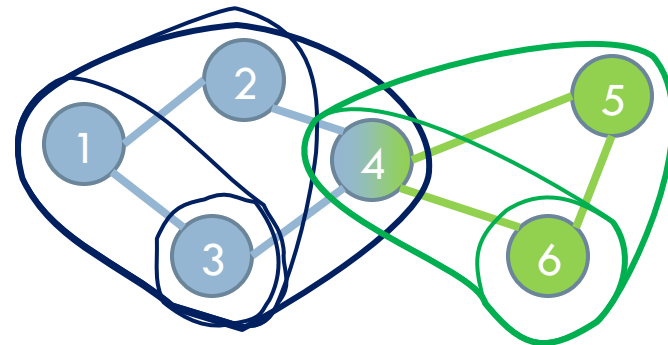
- ❑ 2. **Line graph** transformation
 - ❑ Edges become nodes.
 - ❑ New nodes have connection if they originally share a node.
 - ❑ Then, apply hard clustering algorithm on the line graph.
 - ❑ E.g. LinkCommunity [Ahn2010]
 - ❑ Cons: An edge can only belong to one cluster



Overlapping community detection

❑ 3. Local clustering

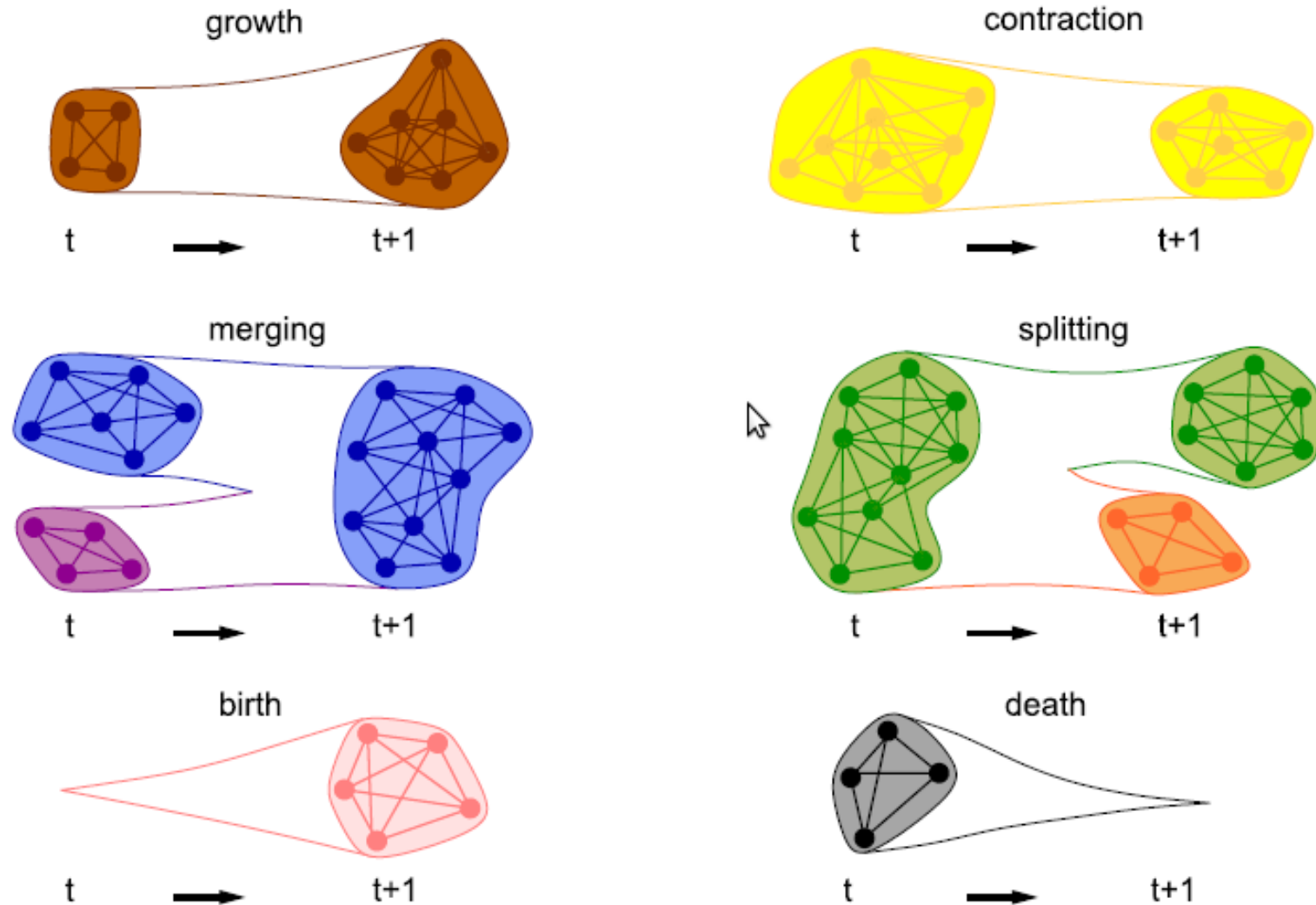
- ❑ (optional) Select seed nodes.
- ❑ Expand seed node according to some criterion.
- ❑ E.g. ClusterOne [Nepusz2012], MCODE [Bader2003], CPM [Adamcsek2006], RRW [Macropol2009]
- ❑ Cons: Not globally consider the topology



Dynamic community

- ❑ Cluster each snapshot independently
- ❑ Then mapping clusters in each clustering.
 - ❑ If two clusters in continuous snapshots share most of nodes, then the next one evolves from the previous one.
- ❑ Detect the **evolution** of communities in a **dynamic graph**.
 - ❑ Birth, Death, Growth, Contraction, Merge, Split.

Dynamic community



Dynamic community

- ❑ Asur et al. (2007) further detect a event involving nodes.
 - ❑ E.g. join and leave
 - ❑ Measure the node behavior.
 - ❑ Sociability: How frequently a node join and leave a community.
 - ❑ Influence: How a node can influence other nodes' activities.
- ❑ Usage
 - ❑ Understand the community behavior.
 - ❑ E.g. age is positively correlated with the size.
 - ❑ Predict the evolution of a community
 - ❑ Predict node (user) behavior, predict link

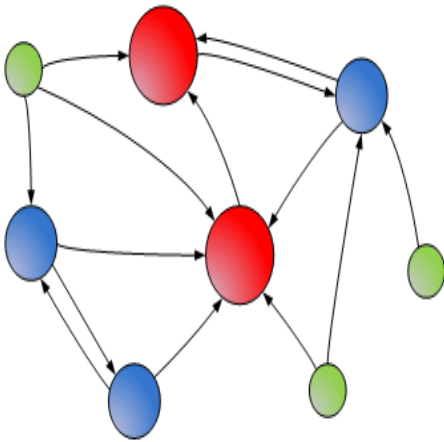
Dynamic community detection

- ❑ Hypothesis: Communities in dynamic graphs are “smooth”.
 - ❑ Detect communities by also considering the previous snapshots.
- ❑ Chakrabarti et al (2006) introduce **history cost**.
 - ❑ Measures the dissimilarity between two clusterings in continuous timestamps.
 - ❑ A smooth clustering has lower history cost.
 - ❑ Add this cost to the objective function.

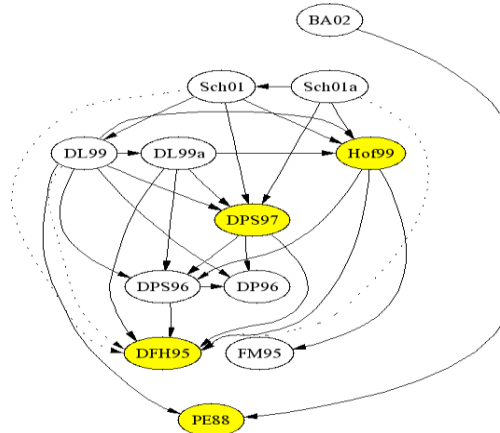
COMMUNITY DISCOVERY IN DIRECTED GRAPHS

- Research on graph clustering is mostly focused on *undirected* graphs, yet the graphs from a number of domains are *directed* in nature.

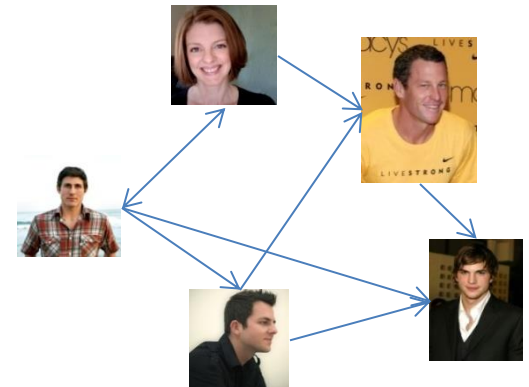
Web graphs



Citation networks



Twitter (follower) network



- Undirected edges indicate similarity/affinity while directed edges need not indicate similarity.

It is important to recognize this difference when clustering.

Existing research

Objective functions such as normalized cuts, originally meant for undirected graphs, have been extended to directed graphs [Zhou et al. '05, Huang et al. '06, Meila '07]

$$Ncut(S) = \frac{\Pr(S \rightarrow \bar{S})}{\Pr(S)} + \frac{\Pr(\bar{S} \rightarrow S)}{\Pr(\bar{S})}$$

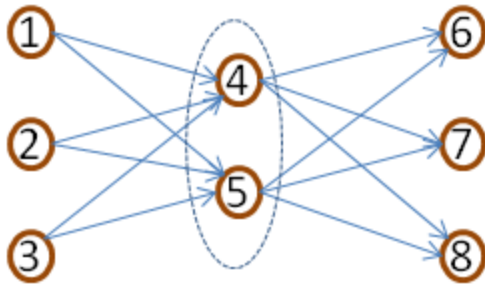
The *Ncut* (normalized cut) of a cluster S is the probability of a random walk escaping from S to the rest of the graph \bar{S} , or vice versa.

Clusters with low *Ncut* are found by spectral methods i.e. by post-processing the eigenvectors of the *directed Laplacian* of the graph.

Drawbacks of Existing Research

Existing measures are biased to find groups of nodes with high inter-connectivity.

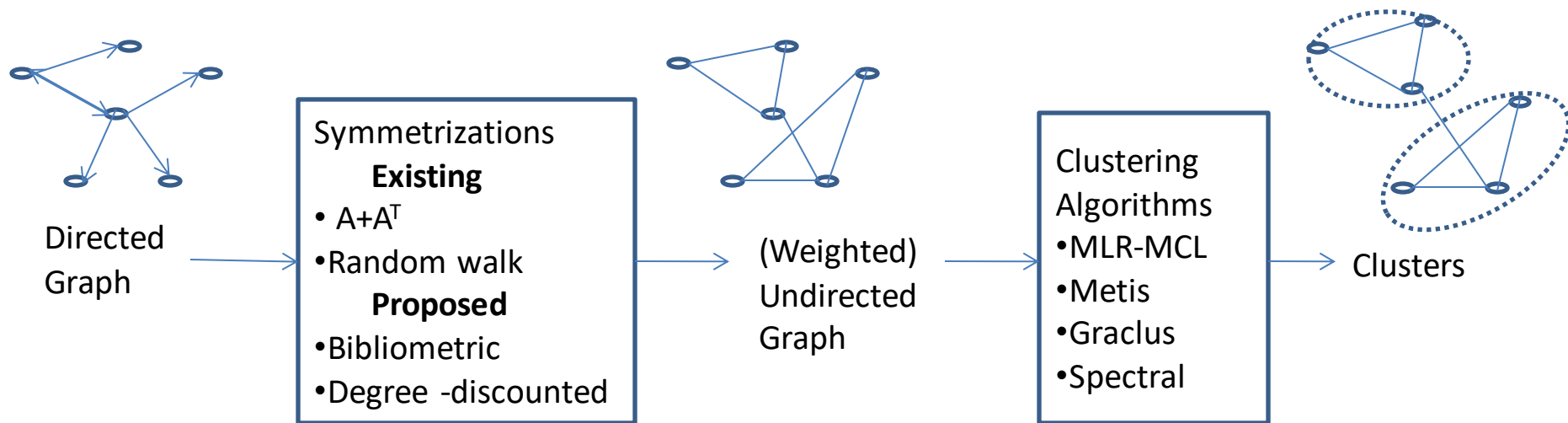
However, directed networks often contain clusters which need not be well inter-connected in the original graph!



Example: Nodes 4 and 5 form a cluster, even though they are not connected to one another.

Real-life analogue: Research papers written on the same topic in a short span of time may not be able to cite one another but may cite (and be cited by) a common set of papers.

Our Framework



By “***Symmetrizations***”, we mean procedures for transforming a directed graph into an undirected graph.

Why a two-stage framework?

Why convert to an undirected graph, and then cluster the undirected graph?

Three reasons:

1. Our framework makes the underlying similarity assumptions explicit.
2. Flexibility: prior methods which directly cluster directed graphs can be re-expressed in our framework.
3. Decouples similarity measure and clustering algorithm, thereby allows use of latest and most suitable clustering algorithms.

Existing symmetrizations

Let the adjacency matrix of the input directed graph be \mathbf{A} .

- **$\mathbf{A} + \mathbf{A}^T$ symmetrization :**

- Corresponds to ignoring directionality.
- Implicit symmetrization that is used widely.

- **Random Walk symmetrization:**

- The directed graph G can be converted into an undirected graph G_U so that the normalized cut on G_U is equal to the normalized cut on G . [Gleich '06]

$$G_U = \frac{\mathbf{\Pi} \mathbf{P} + \mathbf{P}^T \mathbf{\Pi}}{2}$$

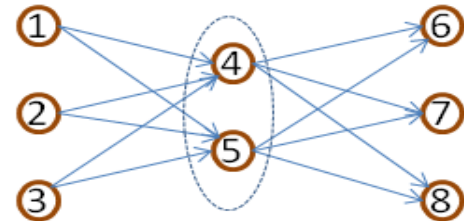
- \mathbf{P} is the Markov transition matrix, and $\mathbf{\Pi}$ is a diagonal matrix with the stationary distribution (PageRank) on the diagonal.
- Clustering G_U is equivalent to the algorithms proposed by [Zhou '05, Huang '06]

Proposed symmetrizations - Bibliometric

Our Approach: Design a suitable similarity measure for pairs of vertices, and set the edge weight between a pair of vertices in the symmetrized graph to be their similarity.

Axioms for similarity:

Axiom 1: Vertices are similar if they point to or are pointed at by common vertices.



Similarity(i, j) = No. of shared in-links + No. of shared out-links

Symmetrized graph

$$G_U = A^T A + A A^T$$

The diagram shows the equation $G_U = A^T A + A A^T$. A blue arrow points from the text "Symmetrized graph" to G_U . Another blue arrow points from the text "No. of shared in-links" (from the line above) to $A^T A$. A third blue arrow points from the text "No. of shared out-links" (from the line above) to $A A^T$.

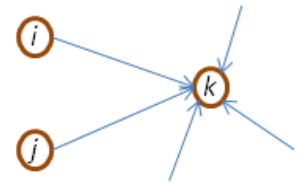
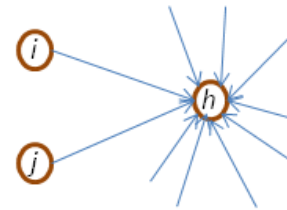
We call this **Bibliometric symmetrization** (for historic reasons)

Proposed Symmetrizations - Degree-discounted

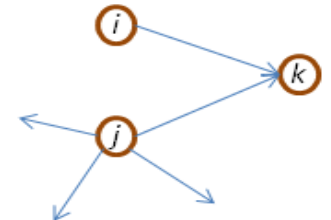
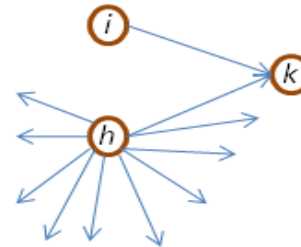
Disadvantage of Bibliometric: Hub nodes can have spuriously high similarity with a lot of nodes.

We propose **Degree-discounted** similarity incorporating the below axioms also:

Axiom 2: Commonly pointing to nodes with high in-degree counts for less than pointing to nodes with low in-degree.



Axiom 3: Being pointed at by nodes with high out-degree counts for less than being pointed at by nodes with low out-degree.



Proposed Symmetrizations - Degree-discounted

A – Adjacency matrix of input directed graph

D_o – Diagonal matrix with out-degrees, **D_i** – Diagonal matrix with in-degrees

Degree-discounted out-link similarity **O_d** between *i* and *j*:

$$O_d(i, j) = \frac{1}{D_o(i)^\alpha D_o(j)^\alpha} \sum_k \frac{A(i, k) A(j, k)}{D_i(k)^\beta}$$

$$O_d = D_o^{-\alpha} A D_i^{-\beta} A^T D_o^{-\alpha}$$

α and **β** are the
degree-discounting
exponents.

Similarly, degree-discounted in-link similarity **I_d** can be derived as:

$$I_d = D_i^{-\beta} A^T D_o^{-\alpha} A D_i^{-\beta}$$

Final degree-discounted similarity matrix: $U_d = O_d + I_d$

We found **α=β=0.5** to work best empirically (similar to L2-normalization).

Pruning Thresholds

For Bibliometric & Degree-discounted, it is critical to prune the symmetrized matrix i.e. remove edges below a threshold.

Two reasons:

1. The full symmetrized matrix is very dense and is difficult to both compute, as well as cluster subsequently.
2. The symmetrization itself can be computed much faster if we only want entries above a certain threshold
 - Large literature on speeding up all-pairs similarity computation in the presence of a threshold e.g. [Bayardo et. al., WWW '07]

It is much easier to set pruning thresholds for Degree-discounted compared to Bibliometric.

Experiments

Datasets:

1. **Cora:** Citation network of ~17,000 CS research papers. Classified manually into 70 research areas. (Thanks to Andrew McCallum.)
2. **Wikipedia:** Article-article hyperlink graph with 1.1 Million nodes. Category assignments at bottom of each article used as the ground truth.

Evaluation Metric:

Avg. F score = Weighted Average of F scores of individual clusters.

F score of a cluster = Harmonic mean of Precision and Recall

w.r.t. ground truth cluster (the best matched one).

Algorithms for clustering symmetrized graphs:

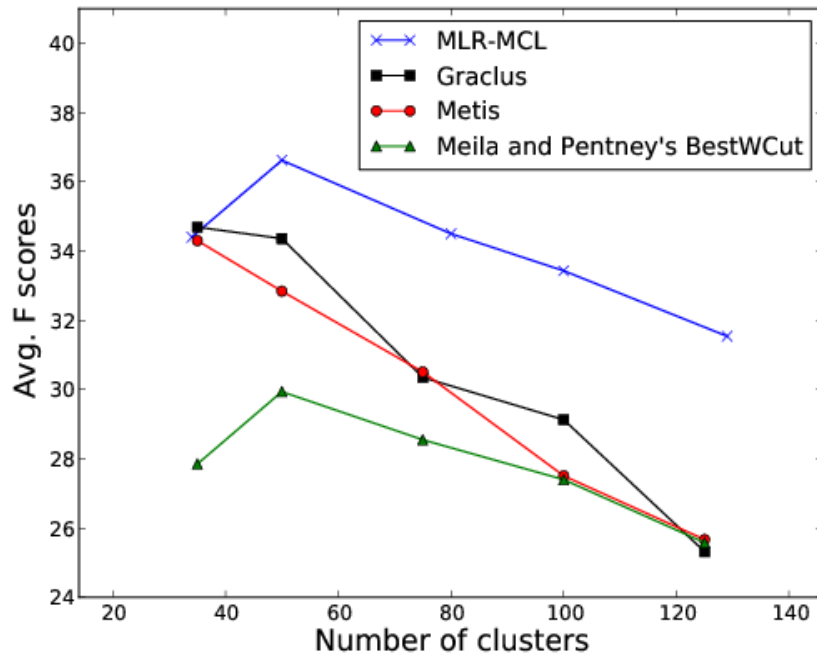
MLR-MCL [Satuluri and Parthasarathy '09]

Graclus [Dhillon et al. '07]

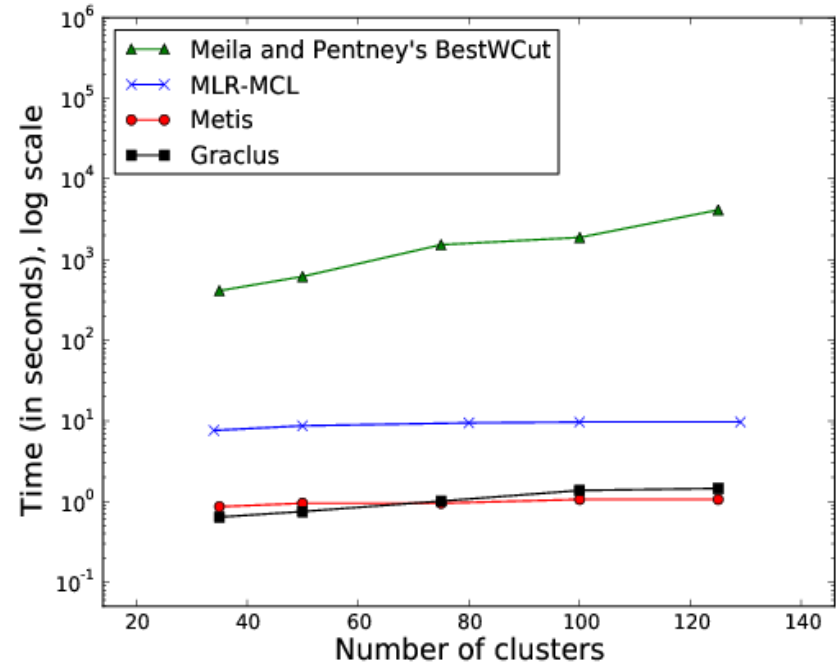
Metis [Karypis and Kumar '98]

Results on Cora – Comparison with BestWCut

Degree-discounted vs BestWCut on Cora

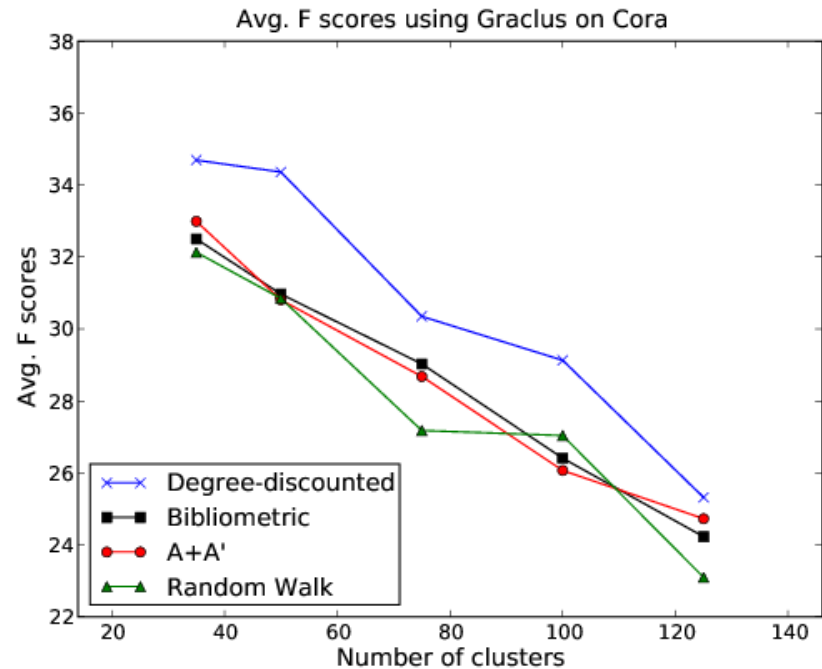
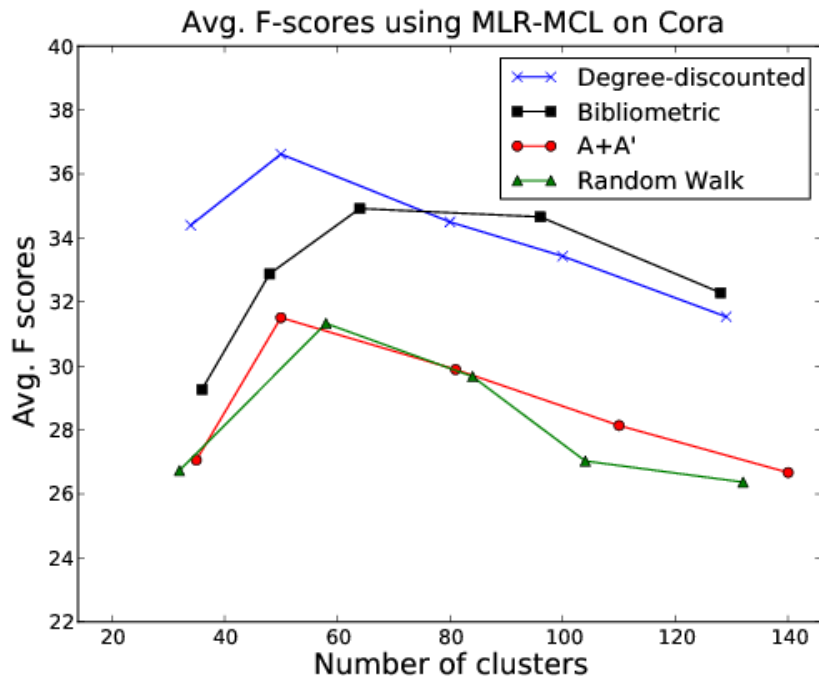


Clustering times on Cora



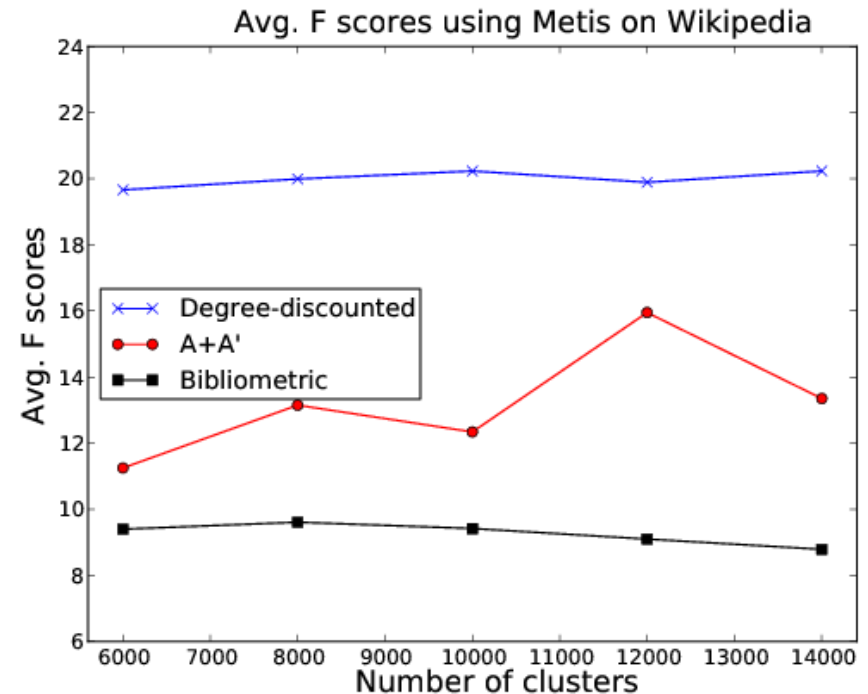
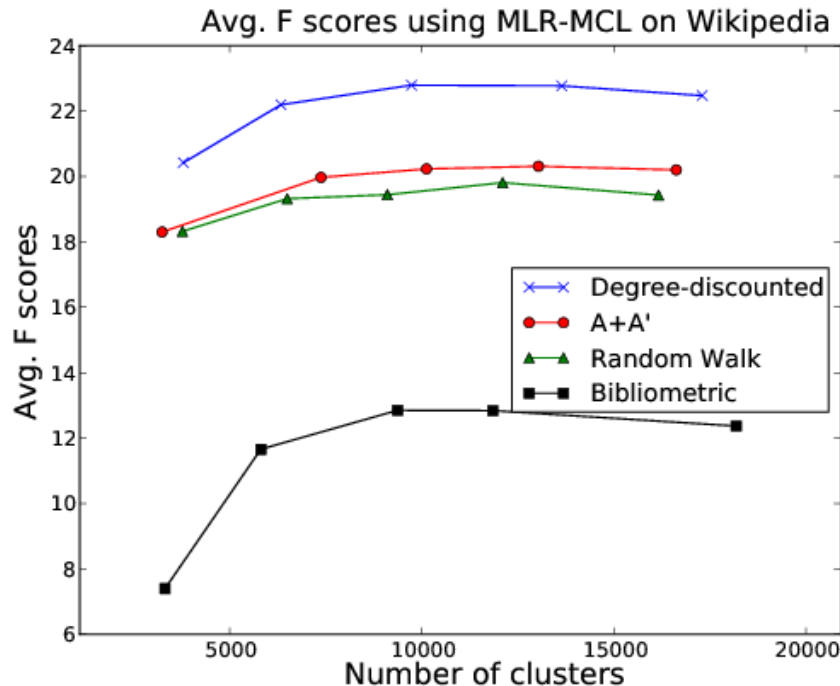
Degree-discounted is 2-3 orders of magnitude faster and also gives higher-quality clusters compared to BestWCut [Meila & Pentney '07].

Results on Cora – Comparison of Symmetrizations



Degree-discounted performs the best among all symmetrizations, when used with either MLR-MCL or Graclus.

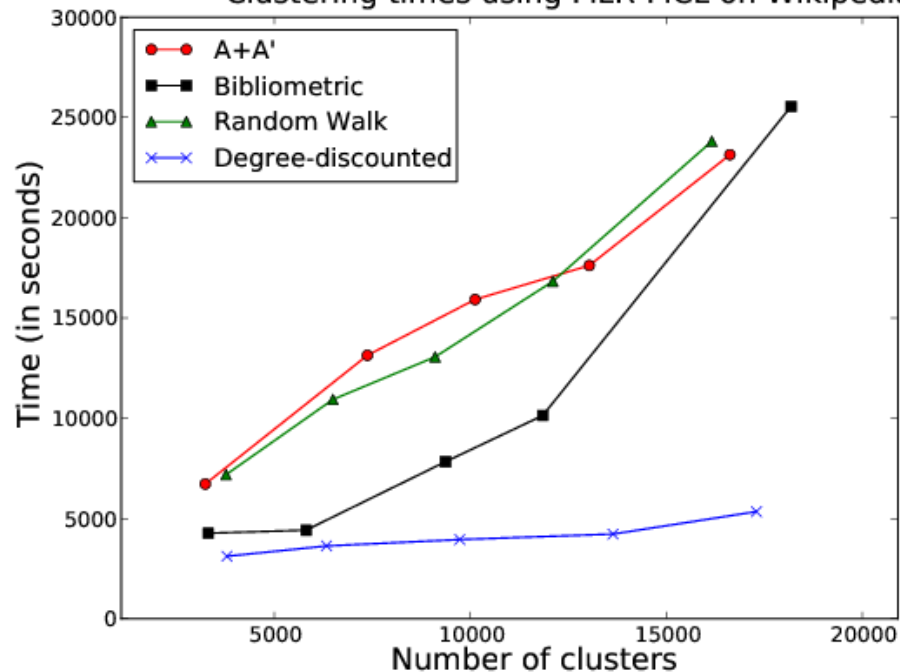
Results on Wikipedia (Quality)



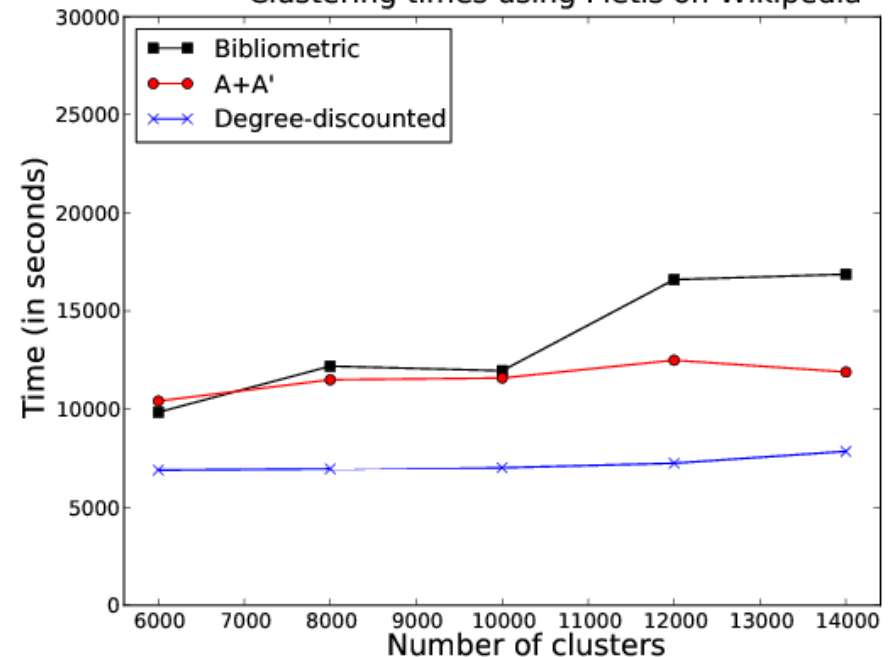
MLR-MCL and Metis show improvements of 12% and 25% on the Degree-discounted graph over the baseline.

Timing results on Wikipedia

Clustering times using MLR-MCL on Wikipedia

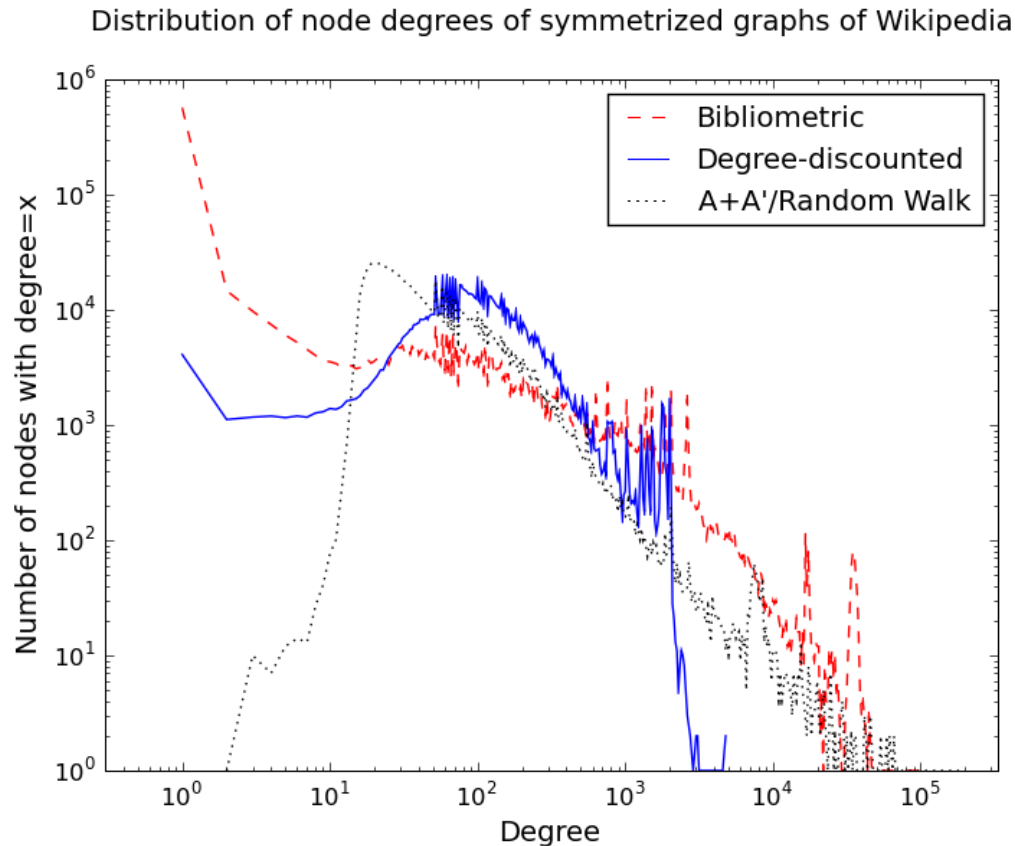


Clustering times using Metis on Wikipedia

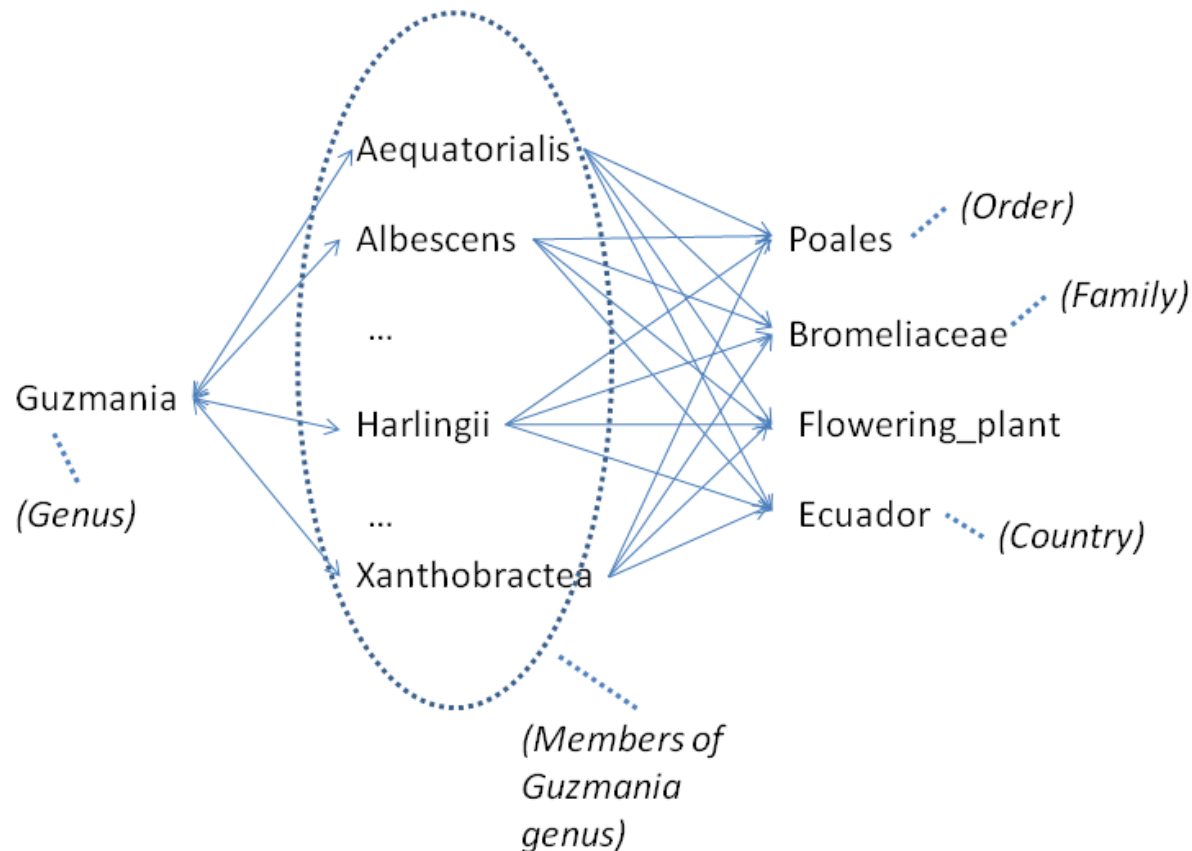


Both MLR-MCL and Metis run 2-4 times faster on Degree-discounted similarity graph.

Degree distribution on Wikipedia



Example Wikipedia cluster



Top similarity pairs in Wikipedia

Symmetrization method	Node 1	Node 2	Edge weight
Random walk	Area	Square mile	3354848
	Mile	Square mile	2233110
	Geocode	Geographic coordinate system	1788953
	Degree (angle)	Geographic coordinate system	1766339
	Area	Octagon	1457427
Bibliometric	Area	Population density	2465
	Record label	Music genre	2423
	Population density	Geographic coordinate system	2301
	Square mile	Population density	2129
	Area	Time zone	2120
Degree-discounted	Cyathea	Cyathea (Subgenus Cyathea)	68
	Roman Catholic dioceses in England & Wales	Roman Catholic dioceses in Great Britain	57
	Sepiidae	Sepia (genus)	55
	Szabolcs-Szatmár-Bereg	Szabolcs-Szatmár-Bereg-related topics	53
	Canton of Lizy-sur-Ourcq	Communauté de communes du Pays de l'Ourcq	52

Testing algorithms

- ❑ 1. Real data w/o gold standards:
 - ❑ 2. Real data w/ gold standard
 - ❑ 3. Synthetic data
-
- ❑ Hard to say which algorithm is the best.
 - ❑ In different scenarios, different algorithms might be best choices.
 - ❑ 1 and 2 are practical, but hard to determine which kinds of graphs / clusters an algorithm is suitable.
 - ❑ Sparse/Dense, power-law, overlapping communities.

Real data w/o gold standards

- Almeida et al. (2011) discuss many metrics.
- Modularity, normalized cut, Silhouette Index, conductance, etc.

$$Ncut(C) = \frac{\sum_{v_i \in C, v_j \notin C} A(i, j)}{\sum_{v_i \in C} degree(v_i)} \quad Ncut(\{C_1, C_2, \dots, C_k\}) = \sum_{i=1}^k Ncut(C_i)$$

- Each metric has its own bias.
 - Modularity, conductance are biased toward small number of clusters.
- Should not choose the algorithms which is designed for that metric, e.g. modularity-based method.

Real data w/ gold standard

- ❑ Examples of gold standard clusters
 - ❑ “Network”tags in Facebook.
 - ❑ Article tags in Wiki
 - ❑ Protein annotations.
- ❑ Evaluate how closely the clusters are matched to the gold standard.
- ❑ Cons: Overfitting – biased towards the clustering with similar cluster size.
- ❑ Cons: Gold standard might be noisy, incomplete.

Metrics

□ F-measure

- Harmonic mean of precision and recall

$$precision = \frac{|\{c_i \in \mathcal{C} | \exists g_j \in \mathcal{G}, NA(c_i, g_j) \leq \theta\}|}{|\mathcal{C}|}$$

$$recall = \frac{|\{g_j \in \mathcal{G} | \exists c_i \in \mathcal{C}, NA(c_i, g_j) \leq \theta\}|}{|\mathcal{G}|},$$

- Need a parameter θ (usually 0.25)

□ Accuracy

- Square root of PPV * S_n

- T_{ij} : common nodes in community i and cluster j

$$S_n = \frac{\sum_{i=1}^n \max_j \{T_{ij}\}}{\sum_{i=1}^n N_i}$$

$$PPV = \frac{\sum_{j=1}^m \max_i \{T_{ij}\}}{\sum_{j=1}^m T_j}$$

- Normalized Mutual Information

$$I_{norm}(\mathcal{X}, \mathcal{Y}) = \frac{2I(X, Y)}{H(X) + H(Y)}$$

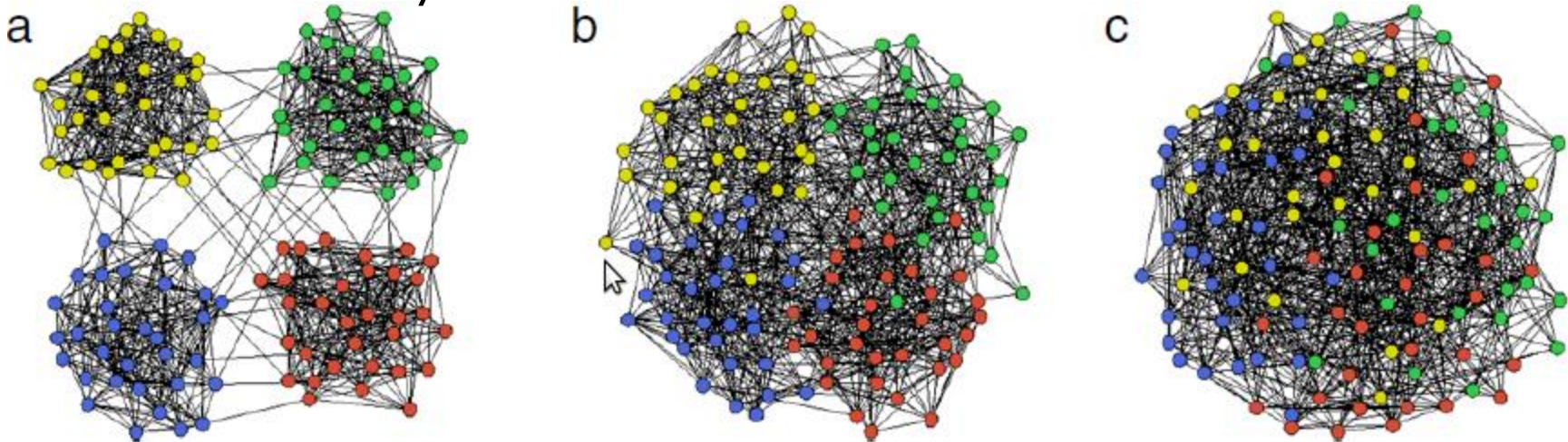
- $H(X)$: Entropy of X
 - $I(X, Y)$: $H(X) - H(X|Y)$, $H(X|Y)$ is the conditional entropy
- Some metrics need to be adjusted for overlapping clustering.

Synthetic data

- ❑ Girvan and Newman (2002) Benchmark

- ❑ Fixed 128 nodes and 4 communities

- ❑ Can tune noisy level



- ❑ Cons: All nodes have the same expected degree;
All communities have the same size, etc

Synthetic data

- ❑ **LFR** (Lancichinetti 2009)
 - ❑ Generate power-law, weighted/unweighted, directed/undirected graph with gold standard
 - ❑ Pros: can generate various graphs.
 - ❑ # nodes, average degree, power-law exponent.
 - ❑ Average/Min/Max community size, # bridge nodes.
 - ❑ Noisy level, etc.
 - ❑ Cons: The number of communities each bridge nodes belonging to is fixed.
- ❑ Use the above metrics to evaluate the result.