Graphs and Social Networks

- Social Network data
- Community Detection
- Influence Maximization
- Modeling Distance and Similarity
- Linked data and RDF

Social Network Data

- A social network is a graph of users (nodes) connected based on their social relationships (edges)
- Users (nodes) may carry features
  - User attributes (profiles)
- Edges can be undirected (Facebook) or directed (Twitter)
- Edges may be labeled and/or may carry weights of importance

Other examples of Large Networks

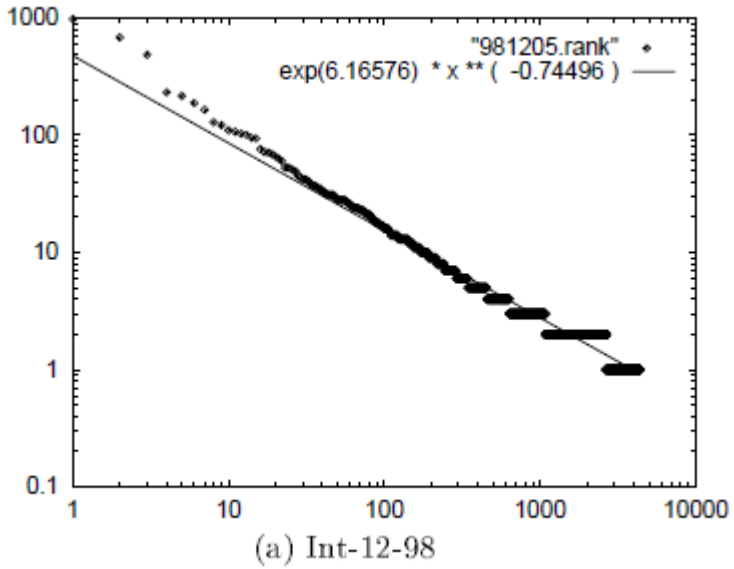
- Communication networks
  - Nodes: people, Links: email exchange
  - Nodes: internet nodes, Links: data/message exchanges
- Financial networks
  - Nodes: companies,
    Links: financial/collaboration relationships
- Biological networks
  - Nodes: proteins, Links: interactions
- WWW
  - Nodes: pages, Links: hyperlinks

Typical tasks in network data search and analysis

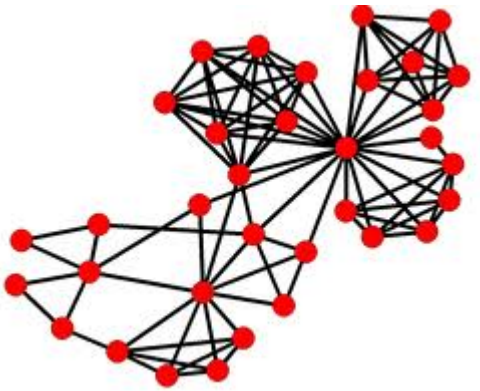
- Given two nodes n_i and n_j find the distance from n_i to n_j
  - Can be used as similarity/influence measure in some contexts
  - Given a node n, find the set of nodes that n influences most (influence ranking)
- Find the general influence of a node in the network
  - Information/virus propagation
  - Find important nodes in the network
- Find structural properties (e.g. communities)
- Link prediction
  - Based on distance, influence, structure of the network
  - Based on content of nodes (homophily)

Understanding large graphs

- What does a network look like?
  - Measure different properties to understand the structure



degree of nodes



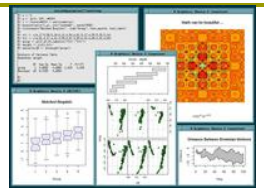
Triangles in the graph

Real network properties

- Most nodes have only a small number of neighbors (degree), but there are some nodes with very high degree (power-law degree distribution)
  - scale-free networks
- If a node x is connected to y and z, then y and z are likely to be connected
  - high clustering coefficient
- Most nodes are just a few edges away on average.
  - small world networks
- Networks from very diverse areas (from internet to biological networks) have similar properties
  - Is it possible that there is a unifying underlying generative process?

Tools

R: free software environment for statistical computing and graphics.
http://www.r-project.org/



Gephi: interactive visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs

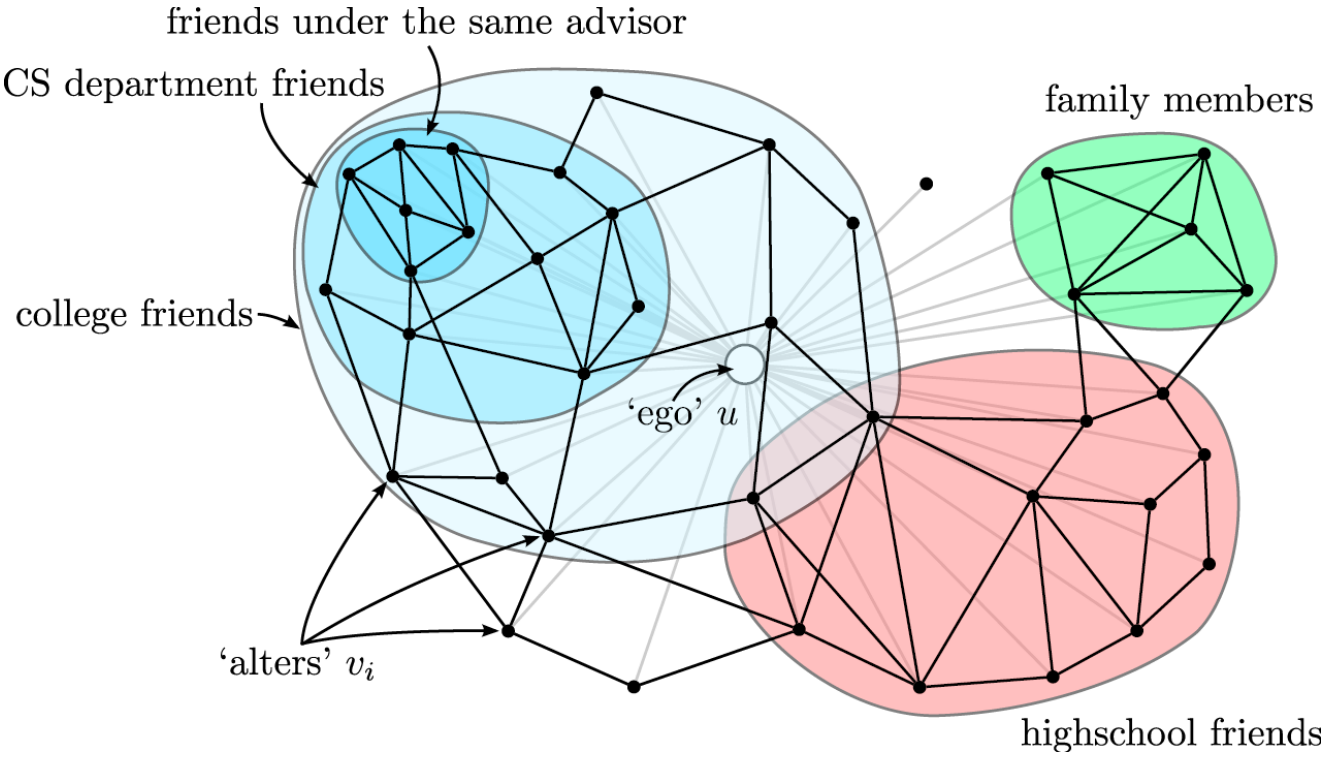
http://gephi.org/

Stanford Network Analysis Platform (SNAP):
general purpose, high performance system for analysis and manipulation of large networks written in C++ http://snap.stanford.edu/snap/index.html

NetworkX: a Python language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. http://networkx.lanl.gov/

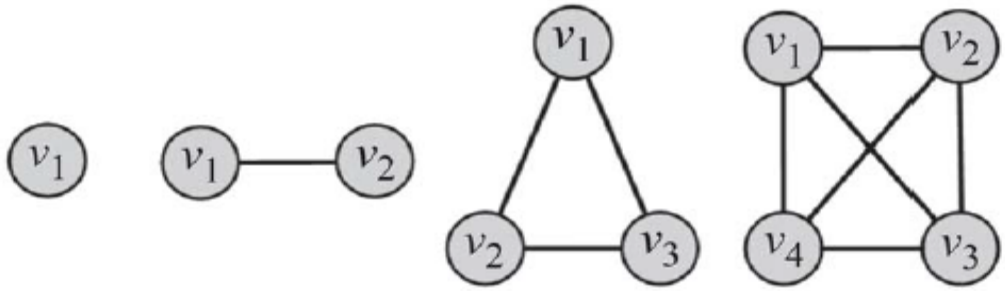
Community detection

- Real networks are *not random graphs*
- Communities
  - aka: groups, clusters, cohesive subgroups, modules



Community detection: Cliques

- Clique: a maximum *complete subgraph* in which all pairs of vertices are connected by an edge.
- A *clique of size k* is a subgraph of *k* vertices where the degree of all vertices in the induced subgraph is *k -1*.



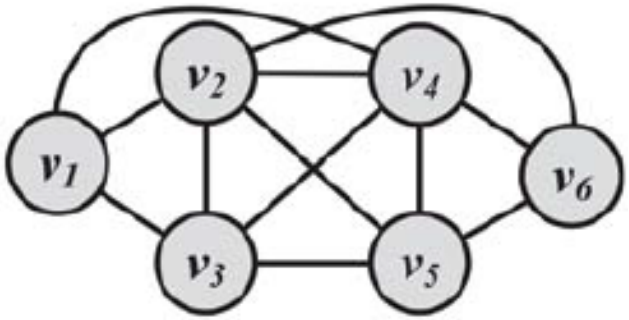
- Search for the *maximum clique* or all *maximal cliques* is NP-hard.

Relaxing Cliques

All vertices have *a minimum degree* but not necessarily *k* -1

k-plex

For a set of vertices V, for all u, $d_u \geq |V| - k$
where d_u is the degree of v in the induced subgraph



Maximal

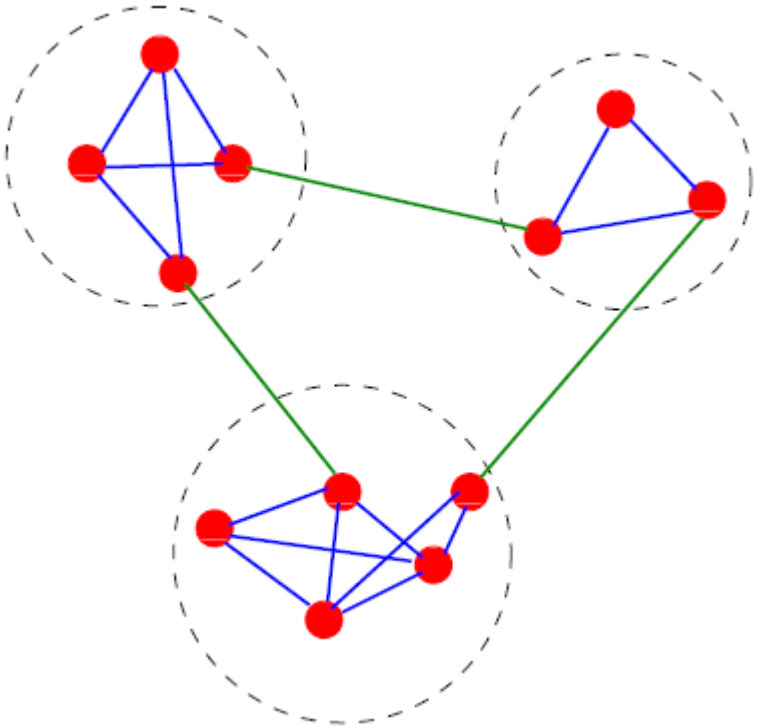
1-plex : $\{v_2, v_3, v_4, v_5\}$

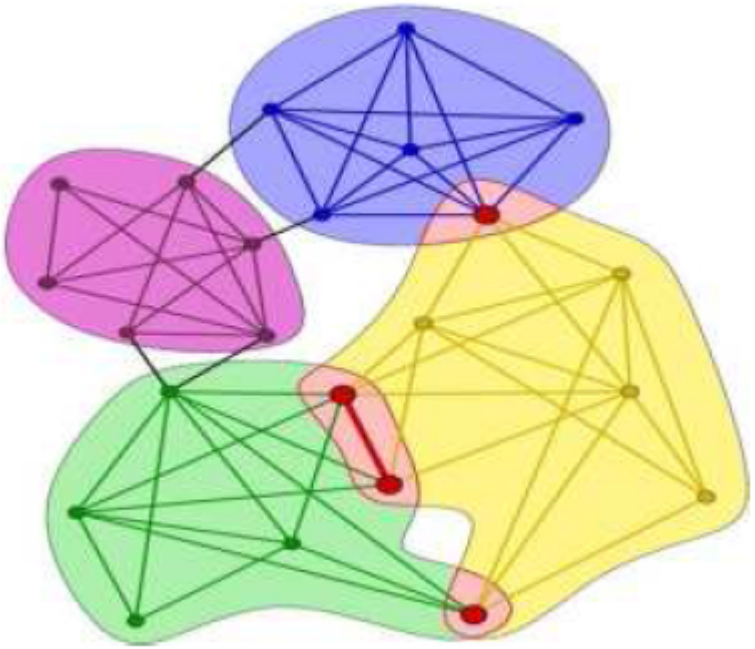
2-plex : $\{v_1, v_2, v_3, v_4, v_5\}, \{v_2, v_3, v_4, v_5, v_6\}$

3-plex : $\{v_1, v_2, v_3, v_4, v_5, v_6\}$

Clique Percolation Method (CPM): Using cliques as seeds

Assumption: communities are formed from a set of cliques and edges that connect these cliques.





Clique Percolation Method (CPM): Using cliques as seeds

1. Given k, find all cliques of size k.
2. Create graph (clique graph) where all cliques are vertices, and two cliques that share k - 1 vertices are connected via an edge.
3.

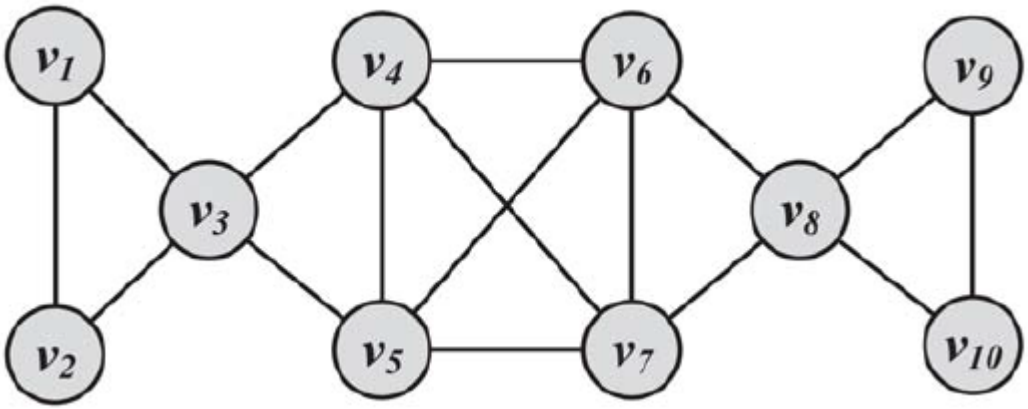
Algorithm 6.2 Clique Percolation Method (CPM)

Require: parameter k

1: **return** Overlapping Communities
2: $Cliques_k$ = find all cliques of size k
3: Construct clique graph $G(V, E)$, where $|V| = |Cliques_k|$
4: $E = \{e_{ij} \mid$ clique i and clique j share $k - 1$ nodes$\}$
5: Return all connected components of G

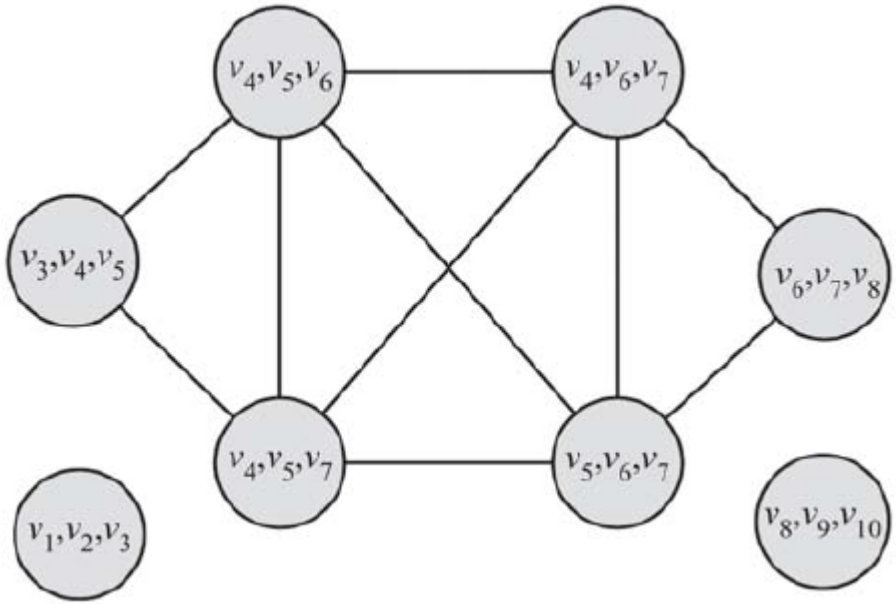
Clique Percolation Method (CPM): Using cliques as seeds

Input graph, let $k = 3$



Clique Percolation Method (CPM): Using cliques as seeds

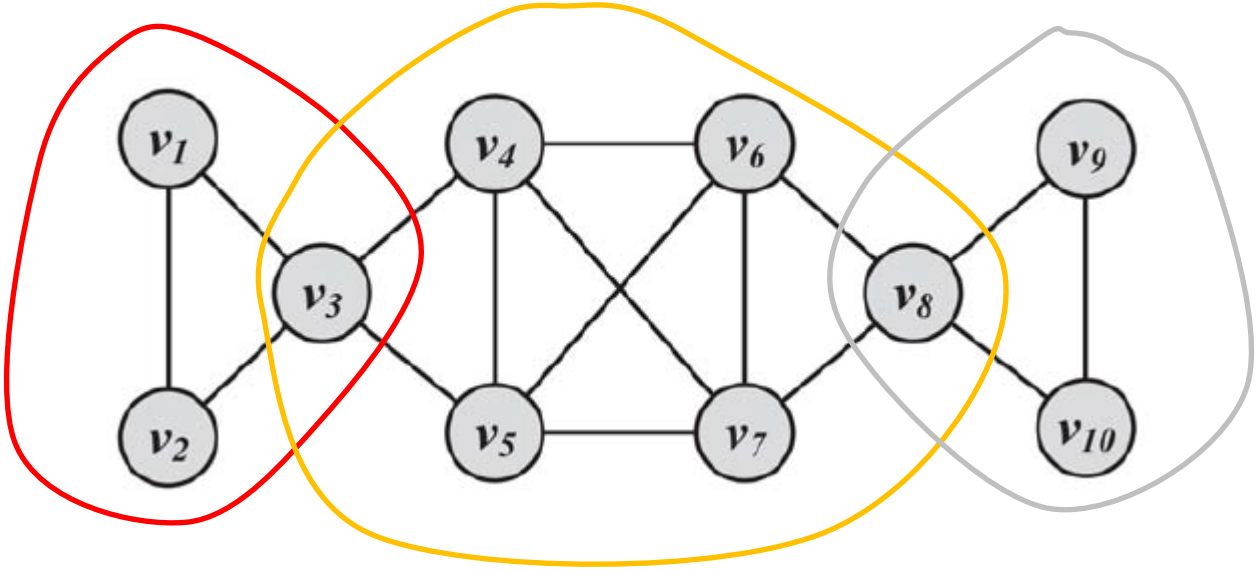
Clique graph for $k = 3$



(v1, v2, v3), (v8, v9, v10), and (v3, v4, v5, v6, v7, v8)

Clique Percolation Method (CPM): Using cliques as seeds

Result



(v1, v2, ,v3), (v8, v9, v10), and (v3, v4, v5, v6, v7, v8)

Clique Percolation Method (CPM): Using cliques as seeds

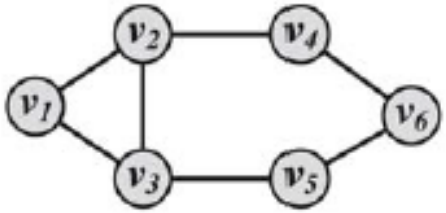
- Two *k*-cliques are adjacent, if they share *k* - 1 vertices.
- The union of adjacent *k*-cliques is called *k*-clique chain.
- Two *k*-cliques are *connected* if they are part of a *k*-clique chain.
- A ***k-clique community*** is the largest connected subgraph obtained by the union of a *k*-clique and of all *k*-cliques which are connected to it.

Clique Percolation Method (CPM): Using cliques as seeds

- A *k*-clique community is identified by making a *k*-clique "roll" over adjacent *k*-cliques, where rolling means rotating a *k*-clique about the *k*-1 vertices it shares with any adjacent *k*-clique.
- By construction, *overlapping* communities
- There may be vertices belonging to nonadjacent *k*-cliques, which could be reached by different paths and end up in different clusters. There are also vertices that cannot be reached by any *k*-clique
- Theoretical complexity grows exponential with size, but *efficient on sparse graphs*

Community detection: vertex similarity

- Define similarity between two vertices
  - Structural equivalence: based on the overlap between their neighborhoods: Jaccard(N(v_1),N(v_2))



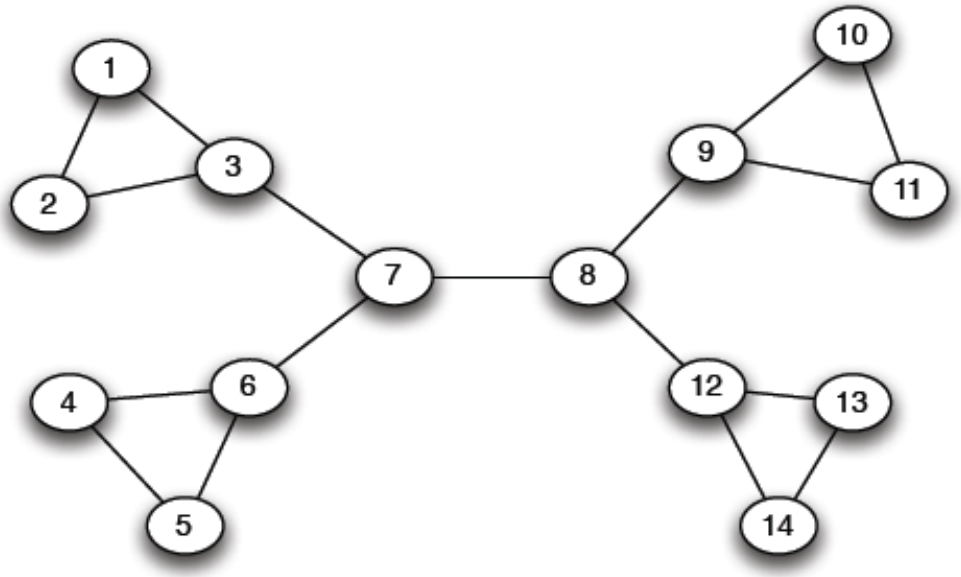
$$\sigma_{\text{Jaccard}}(v_2, v_5) = \frac{|\{v_1, v_3, v_4\} \cap \{v_3, v_6\}|}{|\{v_1, v_3, v_4, v_6\}|} = 0.25$$

- Place similar vertices in the same cluster
- Use traditional *cluster analysis*

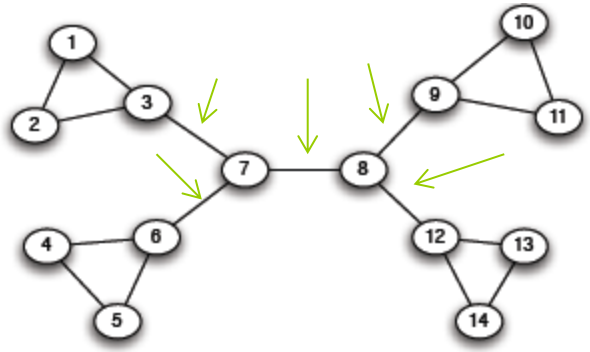
The Girvan Newman method

Hierarchical divisive method

- Start with the whole graph
- Find edges whose removal "partitions" the graph
- Repeat with each subgraph until single vertices



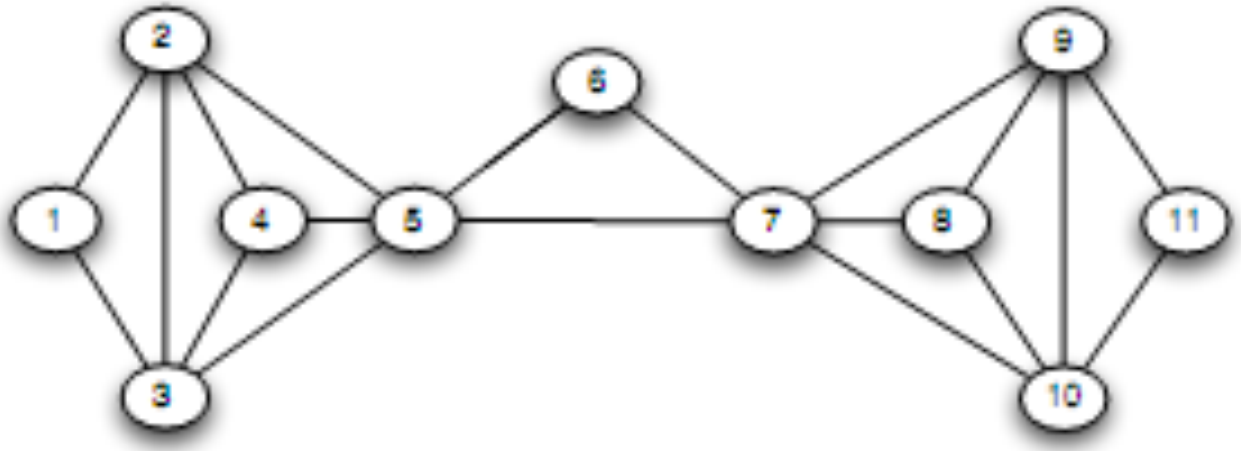
The Girvan Newman method

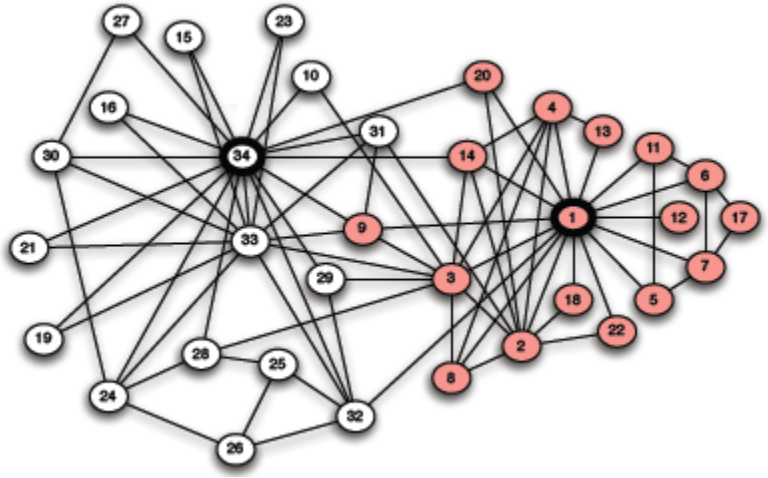


Use bridges or cut-edge (if removed, the nodes become disconnected)

Which one to choose?

The Girvan Newman method

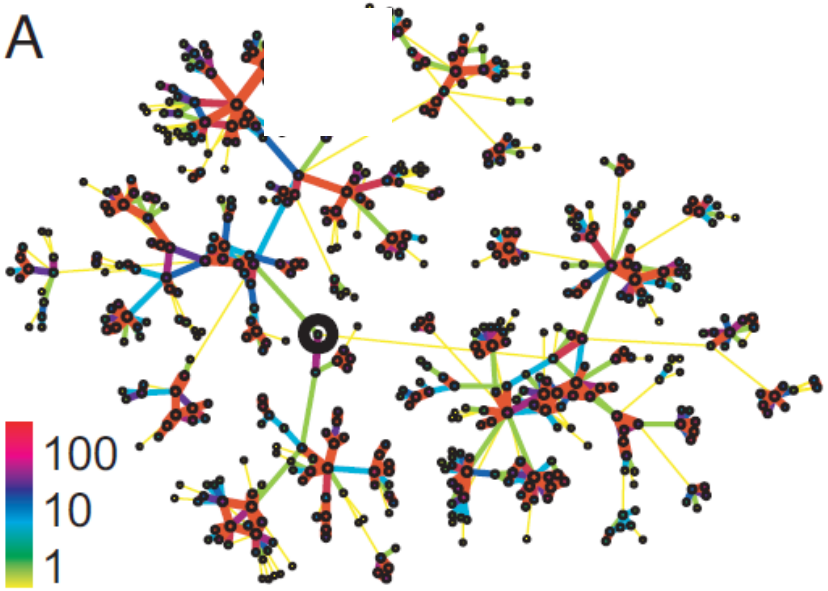




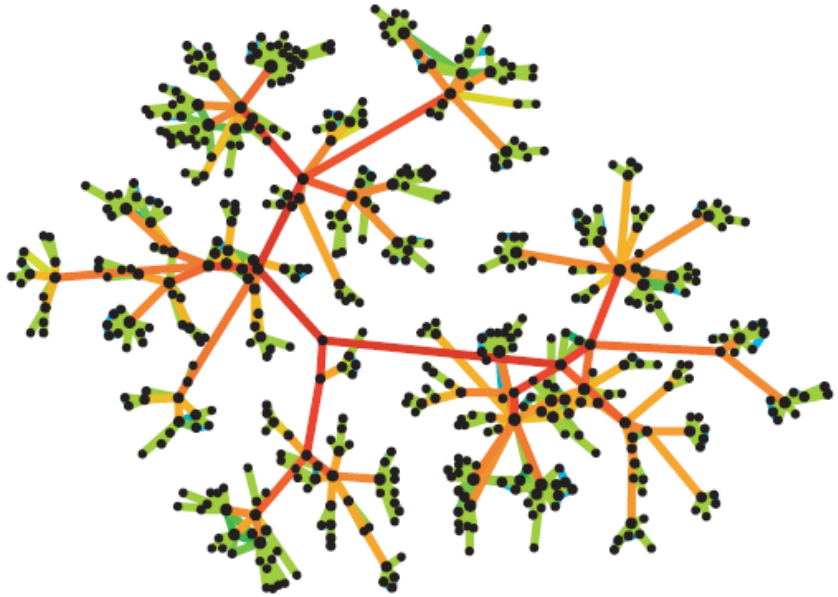
There may be none!

Strength of Weak Ties

- Edge betweenness: Number of shortest paths passing over the edge
- Intuition:



Edge strengths (call volume) in a real network

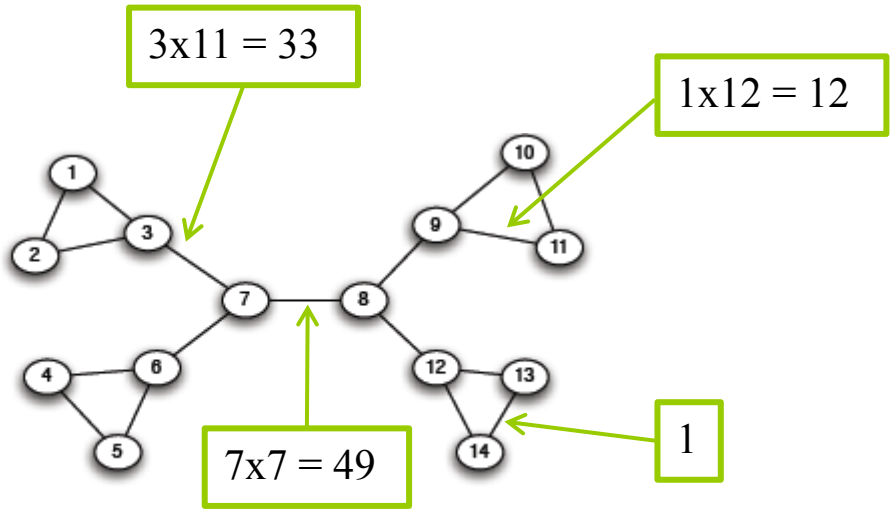


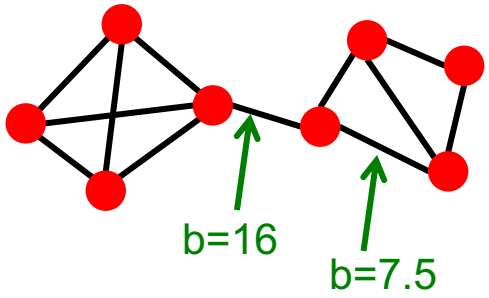
Edge betweenness in a real network

Edge Betweenness

Betweenness of an edge (a, b): number of pairs of nodes x and y such that the edge (a, b) lies on the shortest path between x and y

$$\mathrm{bt}(a,b) = \sum_{x,y} \frac{\#shortest\_paths(x,y)through(a,b)}{\#shortest\_paths(x,y)}$$





Edges that have a high probability to occur on a randomly chosen shortest path between two randomly chosen nodes have a high betweenness.

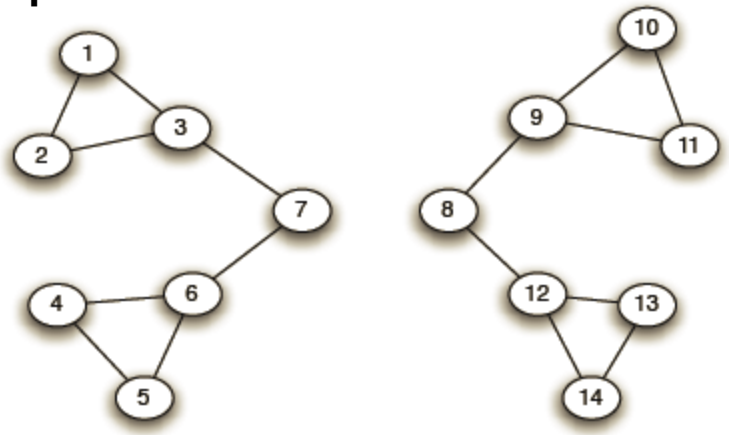
The Girvan Newman method

(for undirected unweighted networks)

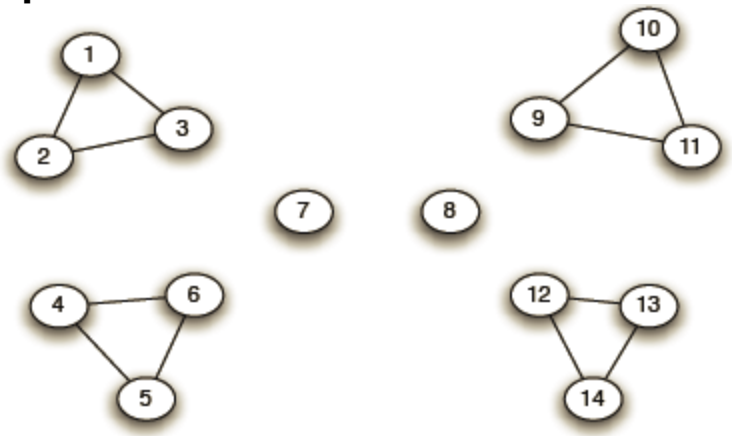
- Repeat until no edges are left**:**
  - Calculate betweenness of edges
  - Remove edges with highest betweenness
- Connected components are communities
- Gives a hierarchical decomposition of the network

Girvan-Newman: Example

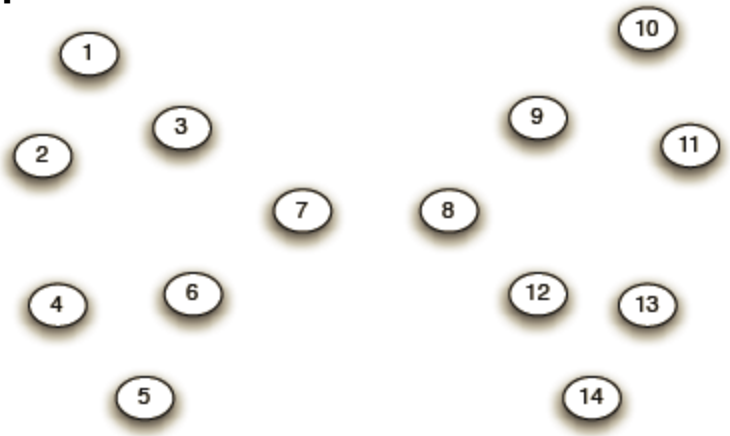
Step 1:



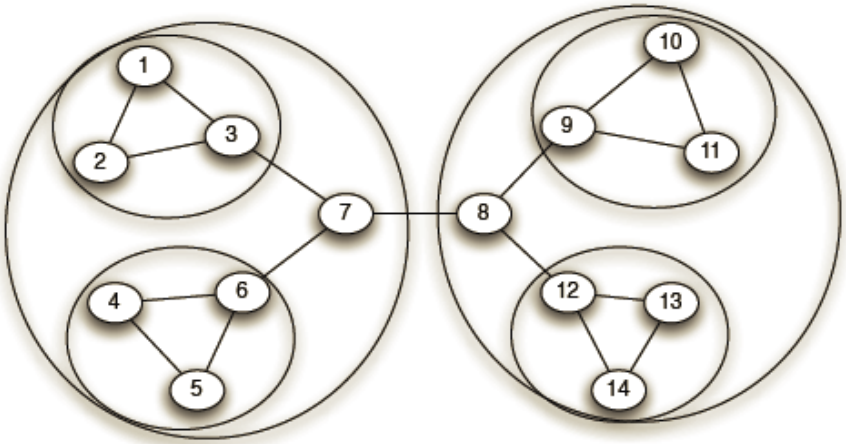
Step 2:



Step 3:

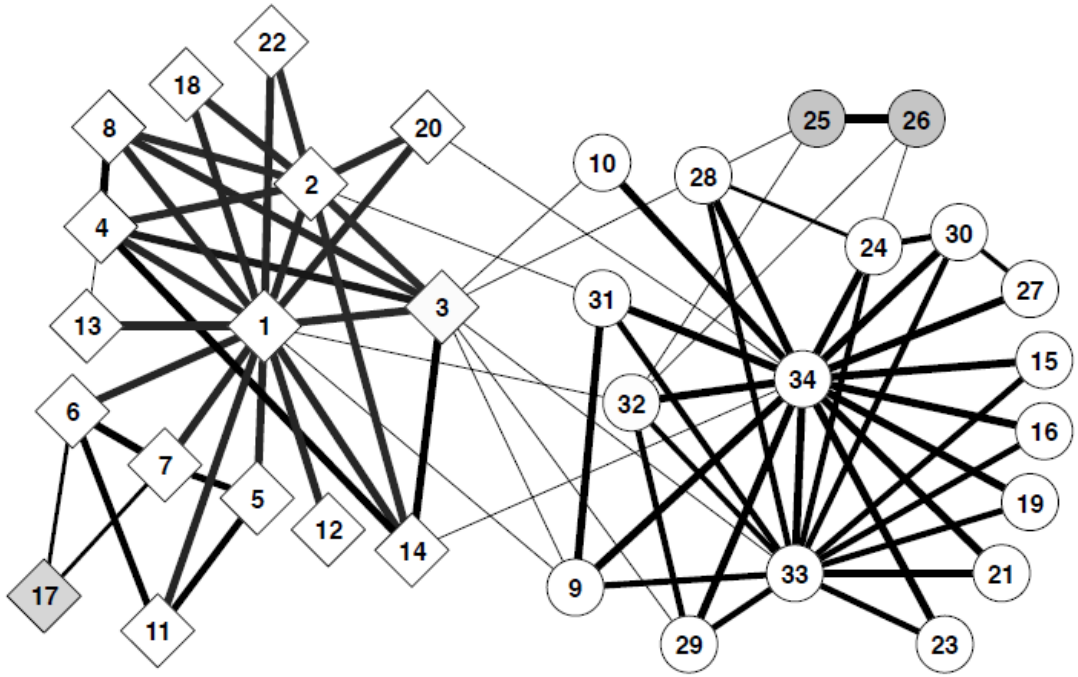


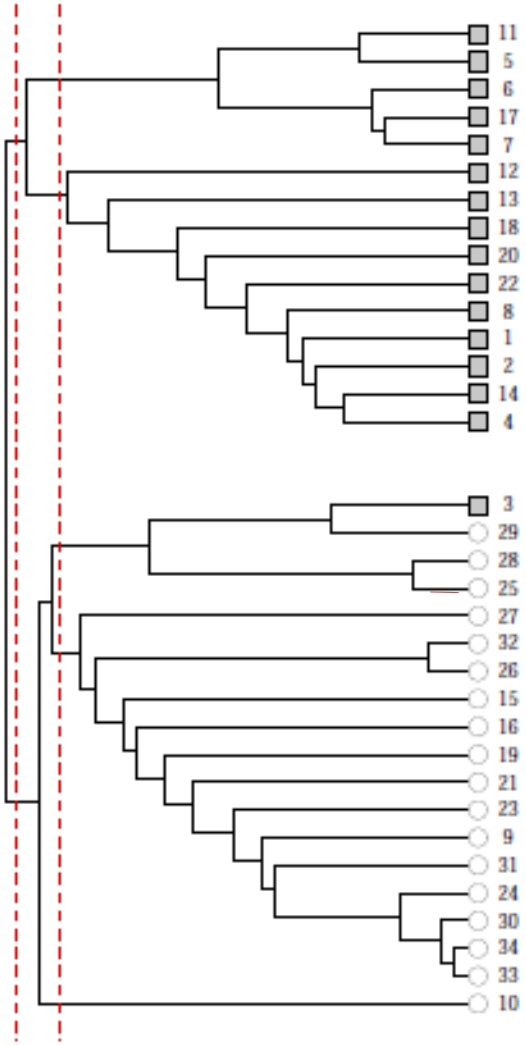
Hierarchical network decomposition:



Girvan-Newman: Results

- **Zachary's Karate club:**
  Hierarchical decomposition





Computing Betweenness

1. Perform a *BFS* starting from a node A
2. Determine the number of shortest path from A to each other node
3. Based on these numbers, determine the amount of flow from A to all other nodes that uses each edge
4. Repeat the process for all nodes
5. Sum over all BFSs

Computing betweenness issues

- Re-compute all paths, or only those affected?
- Redundant SP computation
- Parallel computation?
- Sampling?

Community detection: additional issues

- Evaluation: use modularity measures
  - $Q \propto \sum_{s \in S}$ [(# edges within group s) – (expected # edges within group s)]
- Graph partitioning
  - optimization problem: Partition the nodes in the graph such that nodes *within clusters* are *well interconnected* (high edge weights), and nodes *across clusters* are *sparsely interconnected* (low edge weights)
  - Based on cuts, spectral clustering, dense subgraphs

Innovation Diffusion in Networks

How new behaviors, practices, opinions and technologies *spread* from person to person through a social network as people *influence* their friends to adopt new ideas

Why? Two classes of rational reasons:

- *Direct-Benefit Effect*: there are direct *payoffs* from copying the decisions of others (relative advantage)

  E.g., Phone becomes more useful if more people use it

- *Informational effect*: choices made by others can provide indirect information about what they know (e.g., choosing restaurants)

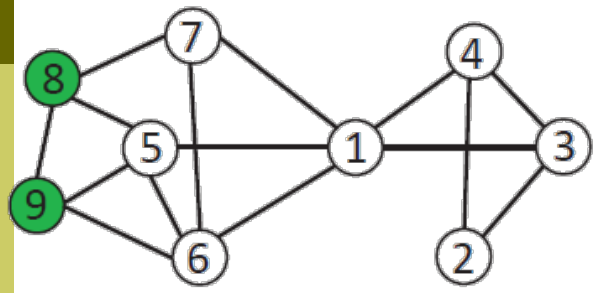
Maximizing spread

- Suppose we have an item (product, idea, video) that propagates in the network
  - Word of mouth propagation.
- An advertiser is interested in maximizing the spread of the item in the network
  - The holy grail of "viral marketing"
- Question: which nodes should we "infect" so that we maximize the spread?

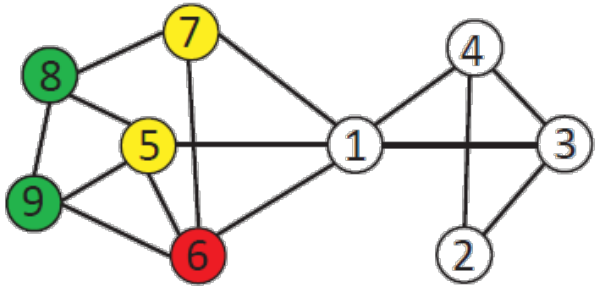
Independent cascade model

- Each node may be active (has the item) or inactive (does not have the item)
- Time proceeds at discrete time-steps.
- At time t, every node v that became active in time t-1 activates a non-active neighbor w with probability p_{uw}. If it fails, it does not try again

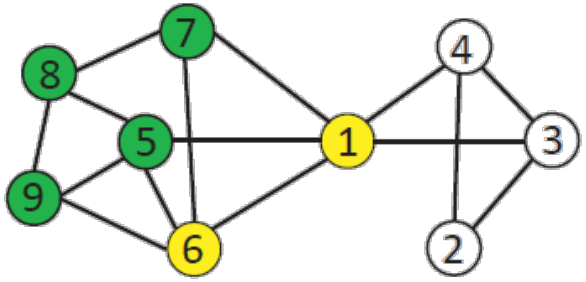
Independent cascade



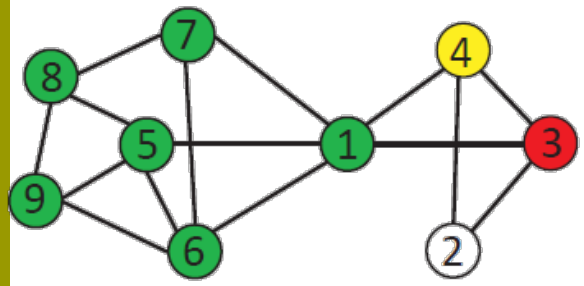
Step 0



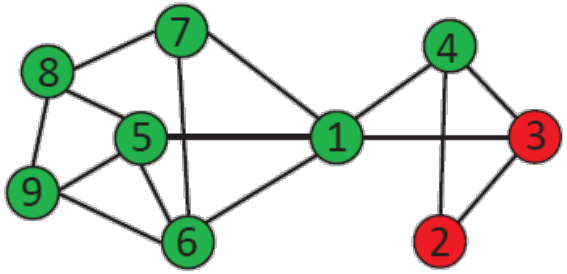
Step 1



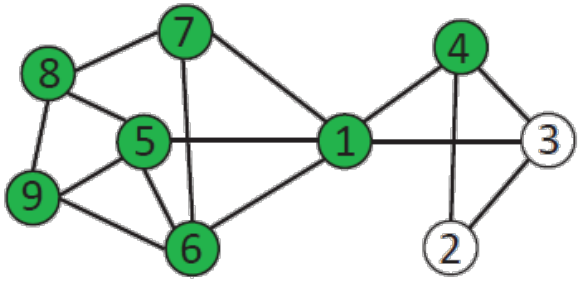
Step 2



Step 3



Step 4



Final Stage

Influence maximization

- Influence function: for a set of nodes A (target set) the influence s(A) (spread) is the expected number of active nodes at the end of the diffusion process if the item is originally placed in the nodes in A.

- Influence maximization problem: Given an network, a diffusion model, and a value k, identify a set A of k nodes in the network that maximizes s(A).

- The problem is NP-hard

A Greedy algorithm

- What is a simple algorithm for selecting the set A?

Greedy algorithm

Start with an empty set A

Proceed in k steps

At each step add the node u to the set A the maximizes the increase in function s(A)

- The node that activates the most additional nodes

- Computing s(A): perform multiple Monte-Carlo simulations of the process and take the average.
- The Greedy algorithm has approximation ratio $\alpha = 1 - 1/e$

Linear threshold model

- Again, each node may be active or inactive
- Every directed edge (v,u) in the graph has a weight b_{vu}, such that

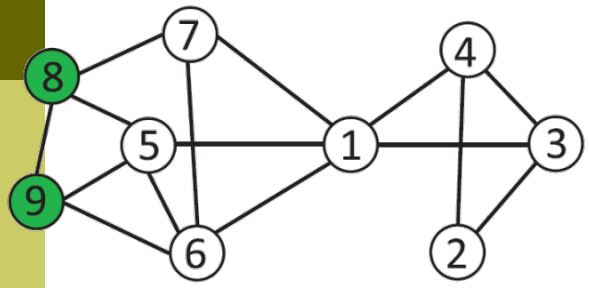
$$\sum_{v \text{ is a neighbor of } u} b_{vu} \leq 1$$

- Each node u has a randomly generated threshold value T_u
- Time proceeds in discrete time-steps. At time t an inactive node u becomes active if

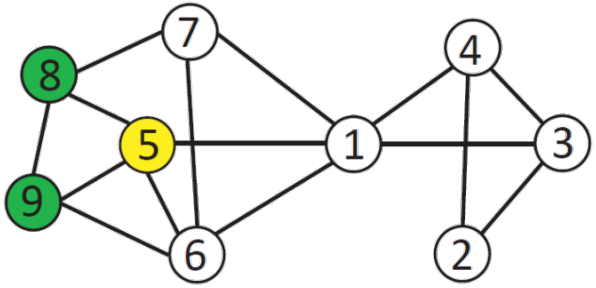
$$\sum_{v \text{ is an active neighbor of } u} b_{vu} \geq T_u$$

- Related to the game-theoretic model of adoption.
- Greedy algorithm also has approximation ratio 1-1/e

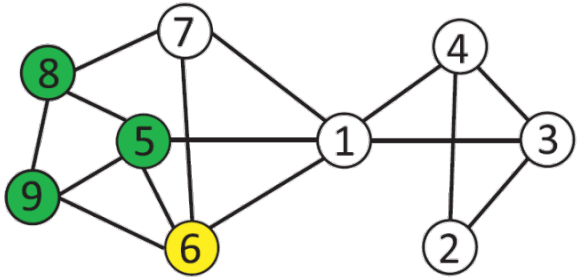
Linear threshold model



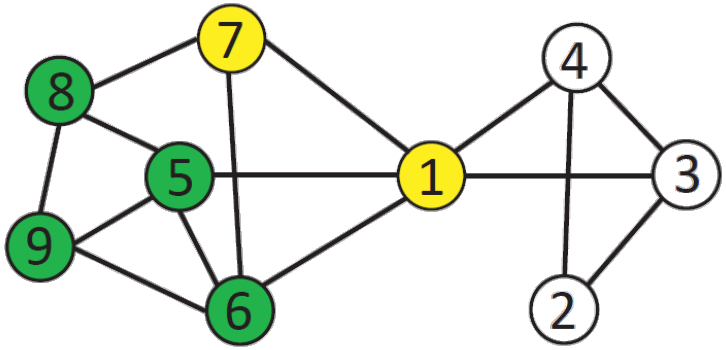
Step 0



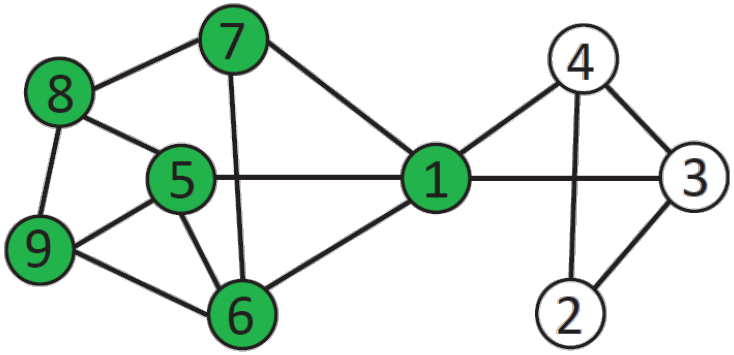
Step 1



Step 2



Step 3

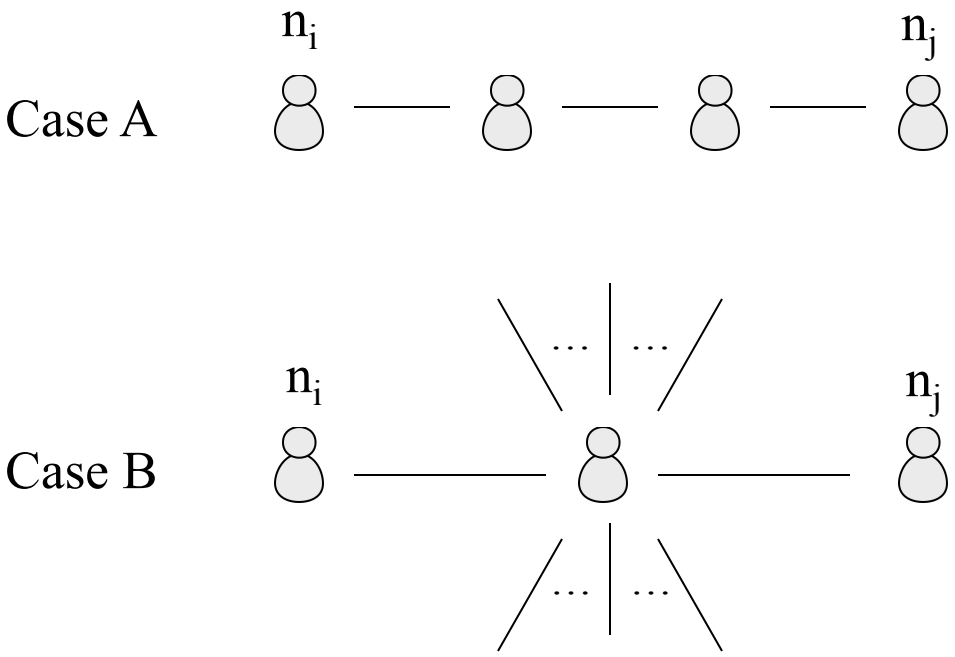


Final Stage

Distance between network nodes

- Models the influence of one node to another
- Link analysis can help in measuring the importance (authority) of a node
  - More important nodes have more influence
- Distance between nodes depends on
  - Set of paths that connect them
  - Authority of nodes in-between

Shortest-path distance is inappropriate



PageRank: Link-based Importance

- Originally defined by Google as a measure of importance for web-pages
- A probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page.

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

- d: damping factor (0.85)
- $M(p_i)$: pages that link to p_i
- $L(p_j)$: no. of outgoing links from p_j

PageRank computation methods

- Iterative:
  - Initialize $PR(p_i) = 1/N$ for all p_i
  - Update $PR(p_i)$ by applying the equation, using the previous $PR(p_i)$ values
  - Stop when values converge
- Algebraic:
  - R: column vector of pageranks to be computed

$$\mathbf{R} = d\mathcal{M}\mathbf{R} + \frac{1-d}{N}\mathbf{1}$$

  - M is a stochastic matrix ($M_{ij} = 1/L(p_j)$ if j links to i; 0 otherwise); 1 is column vector with all 1's.

$$\mathbf{R} = (\mathbf{I} - d\mathcal{M})^{-1}\frac{1-d}{N}\mathbf{1}$$

- Iterative method can also be applied at the matrices (power method)

$$\mathbf{R} = \left(d\mathcal{M} + \frac{1-d}{N}\mathbf{E}\right)\mathbf{R} =: \widehat{\mathcal{M}}\mathbf{R}$$

Personalized PageRank

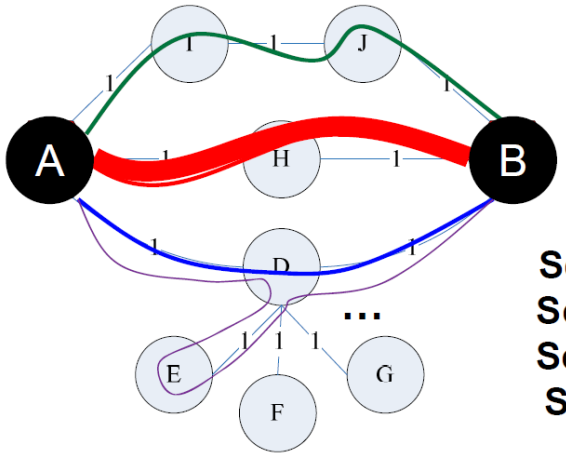
- PageRank assumes that a surfer starts from any node
  - $PR(p_i)$ is the probability that a random surfer from any node will visit p_i.
- Personalized PageRank assumes that the surfer can only start from a given node p_j
- $PPR_{pj}(p_i)$ = probability that a random walk with restart from a given node p_j will reach p_i
- Thus, $PPR_{pj}(p_i)$ models the importance of p_i with respect to p_j
  - $PPR_{pj}(p_i)$ can be used as a measure for the proximity from p_j to p_i
  - In fact, PPR is used by Twitter to present users with other accounts they may wish to follow

Queries based on PPR

- PPR can replace shortest path distance in modeling proximity between nodes in a graph
- Nearest neighbor search for a node p_j in a graph can be modeled as finding the node p_i with the maximum $PPR_{pj}(p_i)$
  - k-NN set for node p_j = top-k nodes with maximum $PPR_{pj}(p_i)$
- Problem: Given a node p_j, compute set of top-k nodes with highest proximity from p_j
- Additional queries based on PPR node-to-node proximity can be defined
  - Range selection: find nodes with PPR node-to-node proximity at least e
  - Joins: find pairs of nodes p_i,p_j such that $PPR_{pj}(p_i)>e$ and $PPR_{pi}(p_j)>e$

Advantages of PPR

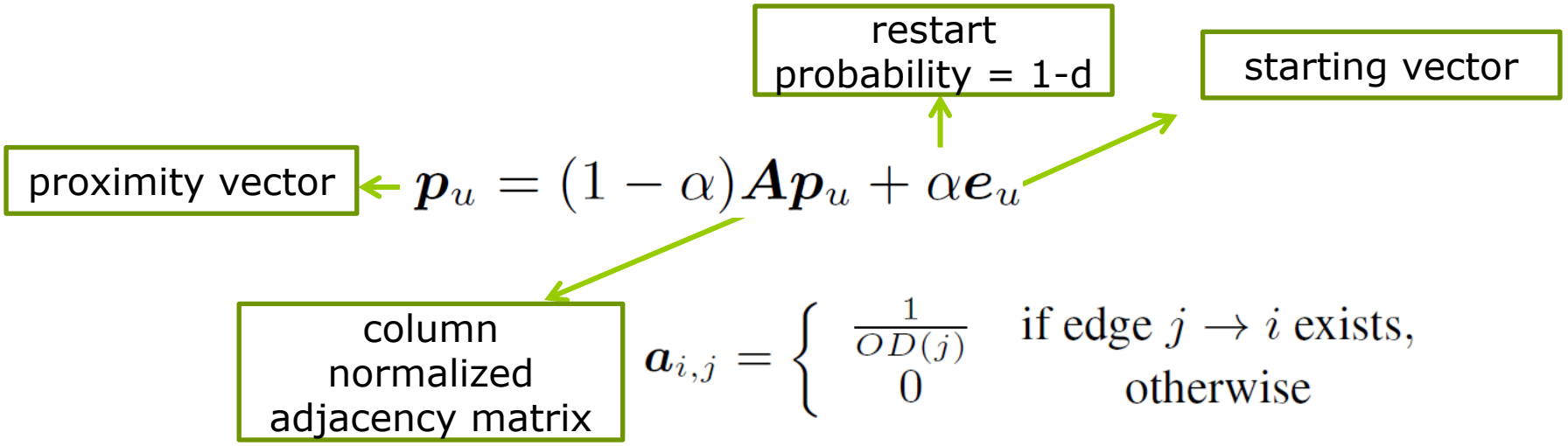
- Considers all the possible paths from node to node
- Captures the global structure of the graph



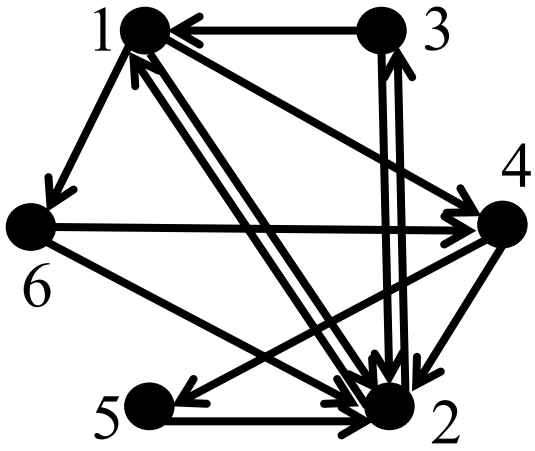
Prox (A, B) =
Score (Red Path) +
Score (Green Path) +
Score (Blue Path) +
Score (Purple Path) +
...

PPR modeled as RWR (random walk with restart)

- p_u = PPR-based proximity vector for node u
- $p_u[v]$ = probability of a random walk with restart from u to hit v



Example: RWR Proximity Matrix



$\boldsymbol{P}$

$\boldsymbol{p}_1$	$\boldsymbol{p}_2$	$\boldsymbol{p}_3$	$\boldsymbol{p}_4$	$\boldsymbol{p}_5$	$\boldsymbol{p}_6$
0.32	0.24	0.24	0.19	0.20	0.18
0.28	0.39	0.29	0.31	0.33	0.30
0.12	0.17	0.27	0.13	0.14	0.13
0.13	0.10	0.10	0.23	0.08	0.14
0.06	0.04	0.04	0.10	0.18	0.06
0.09	0.07	0.07	0.05	0.06	0.20

How to compute p_u

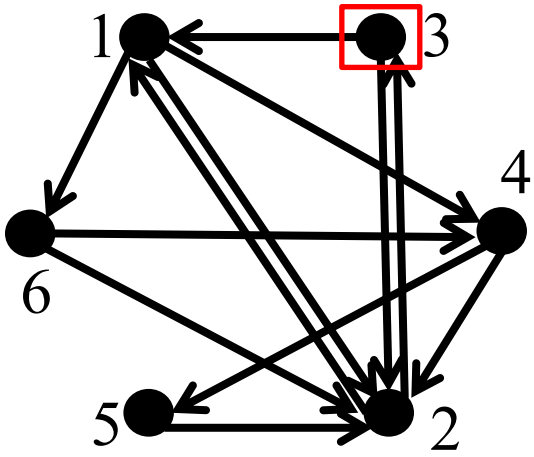
- Iterative computation (Power Method)
  - Accurate, but slow
- Monte-carlo simulation
  - Applies a number of random walks and aggregates the results to estimate probabilities
  - Fast, but less accurate
- Bookmark Coloring Algorithm
  - Iterative method for computing lower bounds for the values of p_u

Bookmark Coloring Algorithm (BCA)

- Model RWR as a ink propagation process.
- Starting from u, for each iteration,
  - α portion is retained in the current node.
  - (1- α) portion evenly propagated to its out-neighbors.
- RWR proximity: the ink retained in each node.

Example

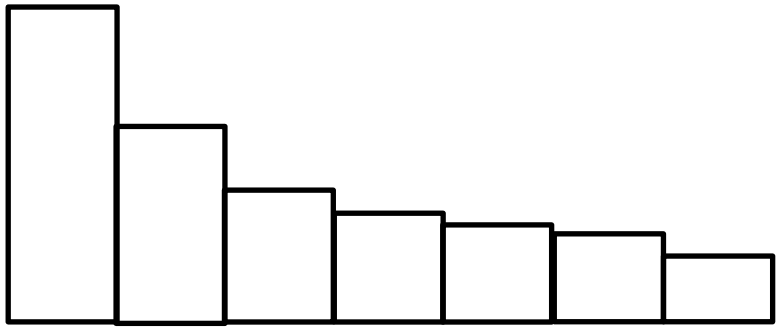
Starting from node 3, $\alpha = 0.15$



Iter 1	Iter 2	Iter 3	Iter 4
0	0.06	0.06	0.06
0	0	0.06	0.06
0.15	0.15	0.15	0.15
0	0	0	0.04
0	0	0	0
0	0	0	0

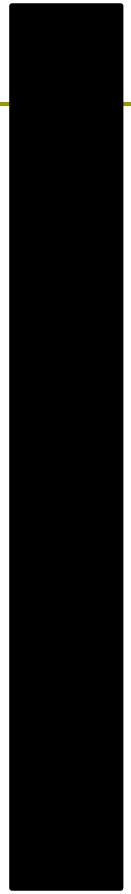
.....	Iter 14	.....	Iter 55
	0.14		0.22
	0.13		0.26
	0.21		0.26
	0.06		0.11
	0.03		0.05
	0.03		0.06

Iteration 5



Current retained ink

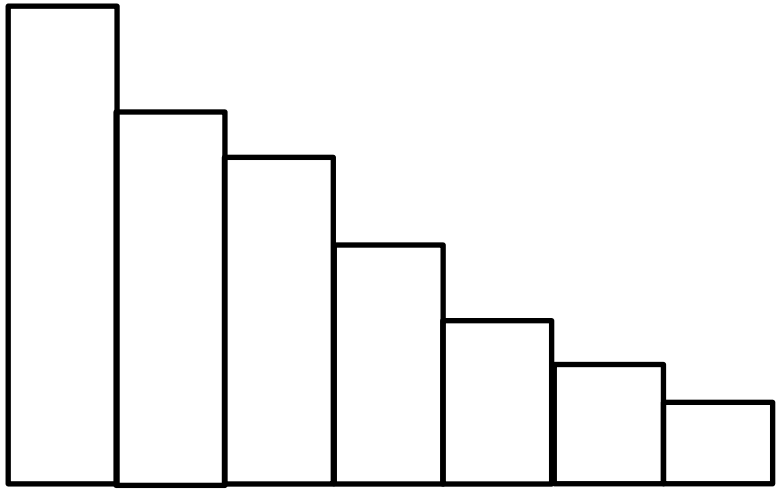
in descending order



Current total

residue ink

Iteration 10



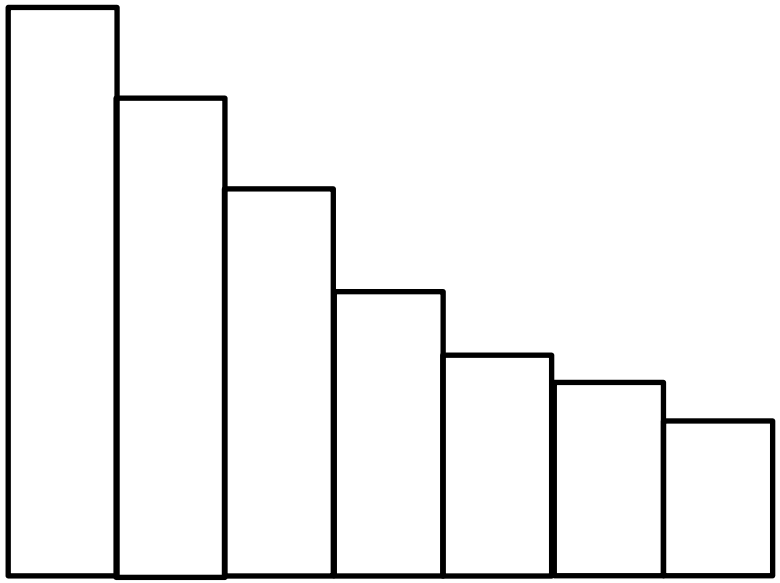
Current retained ink

in descending order

Current total

residue ink

Iteration 20



Current retained ink
in descending order

Current total
residue ink

Computing top-k values of p_u

- Objective: find the top-k values of p_u
  - The nodes that have these values are the k-NN of u in the graph, based on RWR/PPR proximity
- Naïve approach
  - Compute p_u exactly and get the top-k values in it
  - p_u can be computed by any of Power Method, Monte Calro, BCA

Computing top-k values of p_u: early termination approach

- At each iteration of BCA, observe that:
  - the retained ink at each node v is a lower bound of $p_u[v]$
  - the retained ink at each node v plus the total residue ink is an upper bound of $p_u[v]$
- Method:
  - At each iteration, keep track of the top-k lower bounds (set W_k)
  - Also, compute the upper bound U for the (k+1)-th node
    - add the total residue ink to the (k+1)-th lower bound
  - If k-th lower bound is higher than (k+1)-th upper bound, terminate

Other similarity measures: SimRank

- SimRank measures the similarity between two nodes in a graph based on the following principle:
  - two objects are similar if they are related to similar objects
- Recursive definition:

$$s(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b))$$

  - where I(a) is the set of in-neighbors of node a, C is a decay factor
- Iterative computation:

$$R_0(a,b) = \begin{cases} 1 \ , \ \text{if } a = b \ , \\ 0 \ , \ \text{if } a \neq b \ . \end{cases} \qquad R_{k+1}(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_k(I_i(a), I_j(b))$$

Resource Description Framework (RDF)

- Models information as a collection of <subject, predicate, object> triples

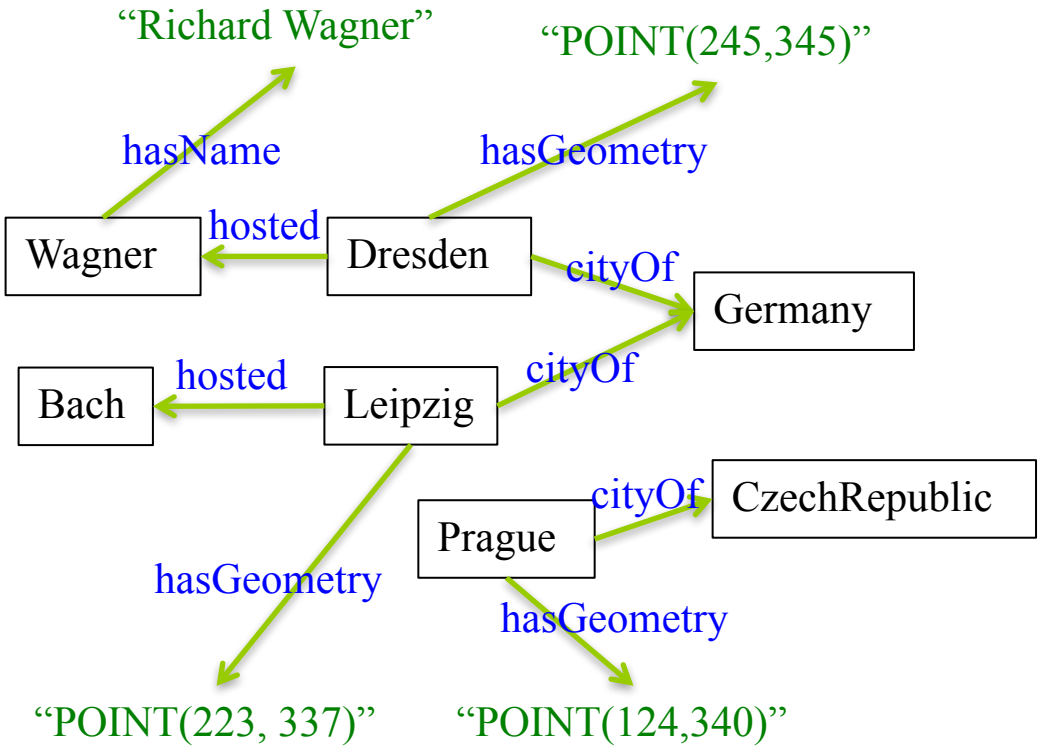
  Dresden hosted Wagner

  Wagner hasName "Richard Wagner"

- Subjects are *resources* (entities)
- Objects can be resources or literals
- Resources are identified by URIs

RDF data as a graph

subject	*property*	*object*
Dresden	cityOf	Germany
Prague	cityOf	CzechRepublic
Leipzig	cityOf	Germany
Dresden	hosted	Wagner
Leipzig	hosted	Bach
Wagner	hasName	"Richard Wagner"
Wagner	performedIn	Leipzig
Dresden	hasGeometry	"POINT (...)"
Prague	hasGeometry	"POINT (...)"
Leipzig	hasGeometry	"POINT (...)"
...	...	...



RDF in practice

- DBpedia - Extracts facts from Wikipedia articles and publishes them as RDF data.
- Creative Commons - Uses RDF to embed license information in web pages and mp3 files.
- YAGO is a huge semantic knowledge base derived
  from Wikipedia, WordNet and GeoNames.
  - more than 10 million entities (e.g. persons, organizations, cities, etc.)
  - more than 120 million facts about these entities.
- …

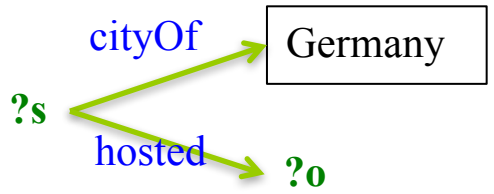
Queries on RDF data

- SPARQL is the standard query language
- Queries in the form of subgraph patterns

SPARQL query

```
Select ?s ?o
Where {
        ?s cityOf Germany .
        ?s hosted ?o .
}
```

graph representation of query



result

?s	?o
Dresden	Wagner
Leipzig	Bach

RDF-3X [Neumann & Weikum, 2008]

- A prototype open-source RDF store that efficiently supports generic SPARQL queries
- URIs and Literals are mapped to unique IDs, by a dictionary. RDF data are represented as triples of IDs.

Dictionary

ID	*URI/literal*
1	Dresden
2	cityOf
3	Germany
4	Prague
5	CzechRepublic
6	Leipzig
...	...

ID-encoded triples

subject	*property*	*object*
1	2	3
4	2	5
6	2	3
...	...	...

RDF-3X [Neumann & Weikum, 2008]

- A clustered B+-tree index for each SPO permutation

SPO

subject	*property*	*object*
1	2	3
4	2	5
6	2	3
...	...	...

SOP

subject	*property*	*object*
1	2	3
4	2	5
6	2	3
...	...	...

PSO

subject	*property*	*object*
1	2	3
4	2	5
6	2	3
...	...	...

POS

subject	*property*	*object*
1	2	3
6	2	3
4	2	5
...	...	...

OSP

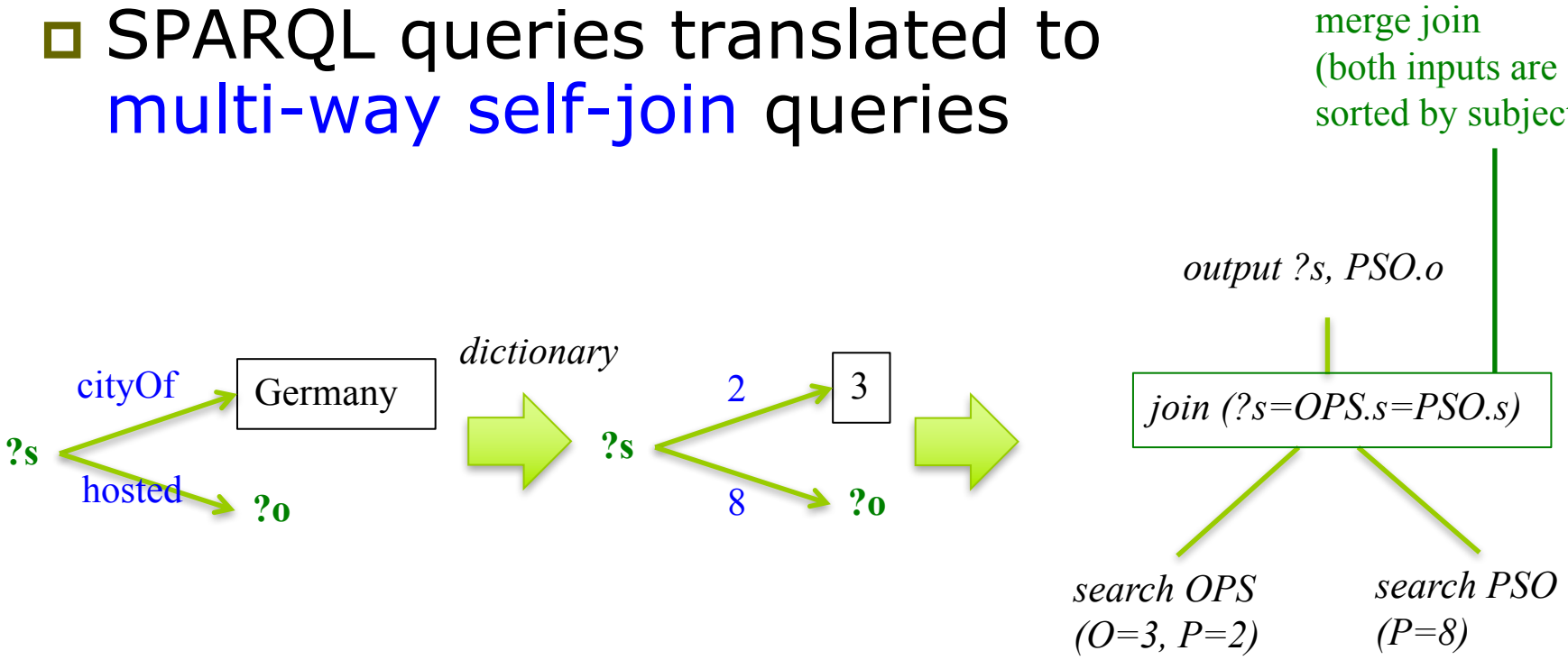
subject	*property*	*object*
1	2	3
6	2	3
4	2	5
...	...	...

OPS

subject	*property*	*object*
1	2	3
6	2	3
4	2	5
...	...	...

RDF-3X [Neumann & Weikum, 2008]

- SPARQL queries translated to multi-way self-join queries



- RDF-3X favors plans that produce *interesting orders*, (merge joins are pipelined without intermediate sorts)

Summary

- Social networks generate large volumes of data that are useful for analysis
- Community detection and influence maximization are important network analysis tasks
- Applications, like recommender systems, use search operations based on node-to-node proximity
  - Shortest path distance not appropriate; better measures based on PageRank (and SimRank)
- RDF knowledge graphs are indexed for the purpose of answering pattern queries expressed in SPARQL

Acknowledgements

- Some slides taken from Online Social Networks and Media course at the University of Ioannina http://www.cs.uoi.gr/~tsap/teaching/cs-l14/references.html