Mining Connection Pathways for Marked Nodes in Large Graphs

Leman Akoglu

Computer Science · Stony Brook University

Hanghang Tong

CUNY

Jilles Vreeken

Universiteit Antwerpen

Polo Chau

GT

Nikolaj Tatti

Katholieke Universiteit Leuven

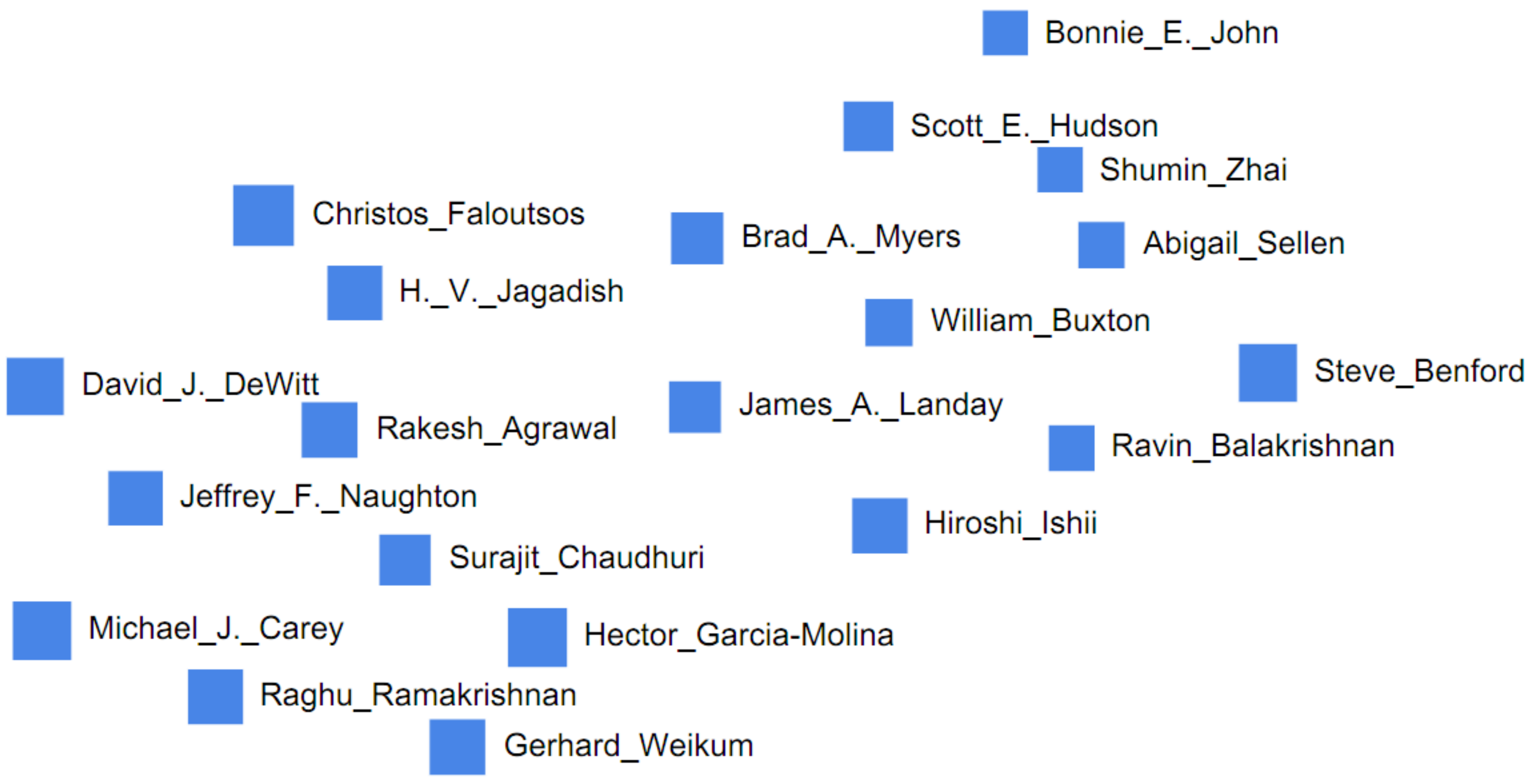
Christos Faloutsos

Carnegie Mellon University

2013 SIAM International Conference on Data Mining (SDM 2013)

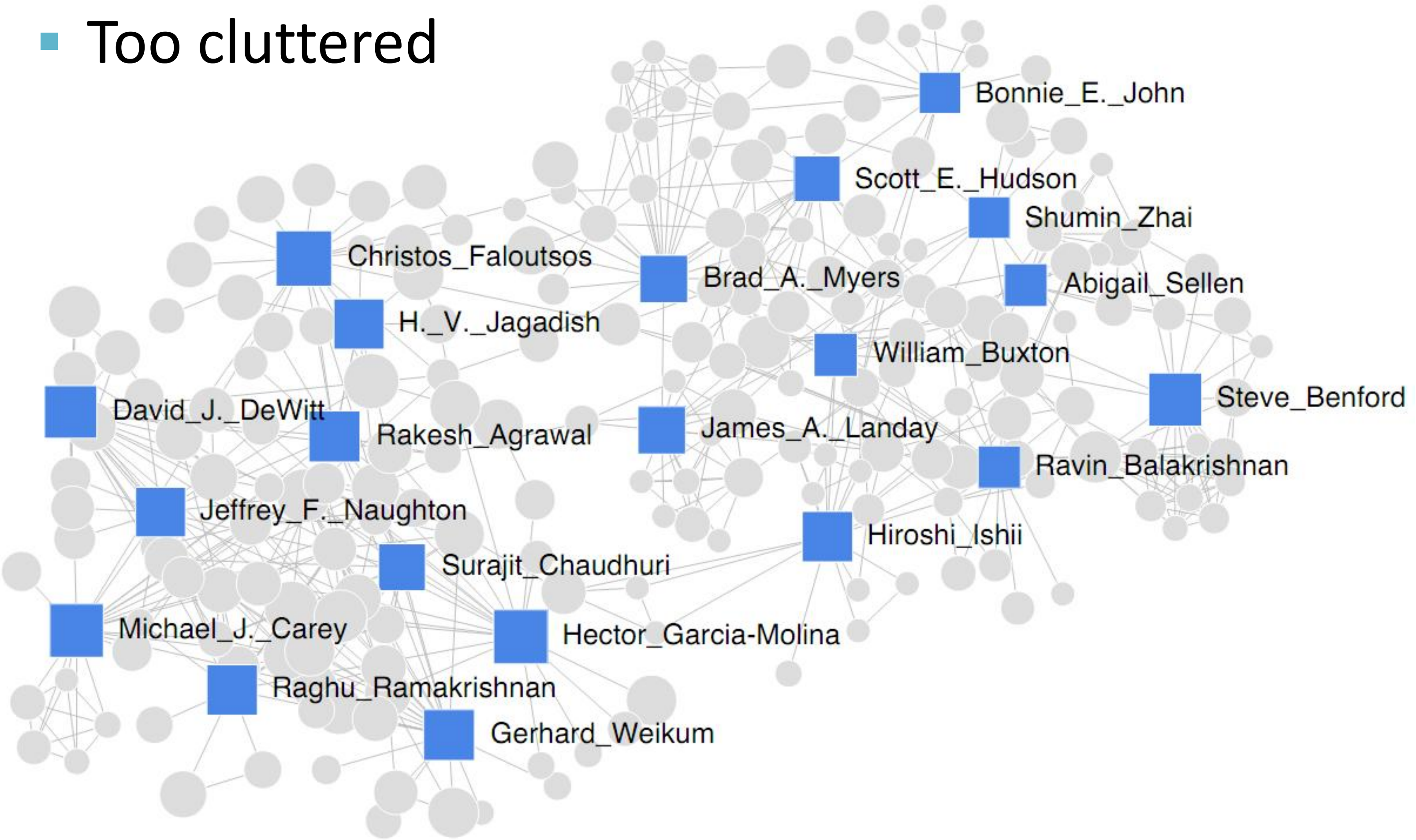
Example

- What can we say about this “list” of authors?
  - Use relational information



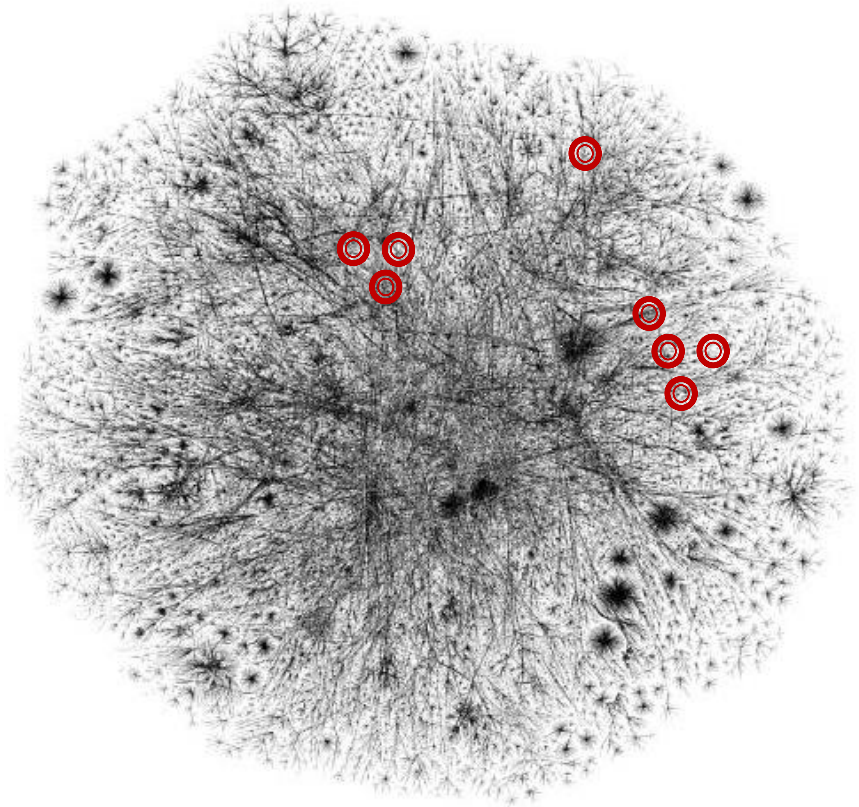
Example

- Any patterns in the co-authorship graph?
  - Too cluttered



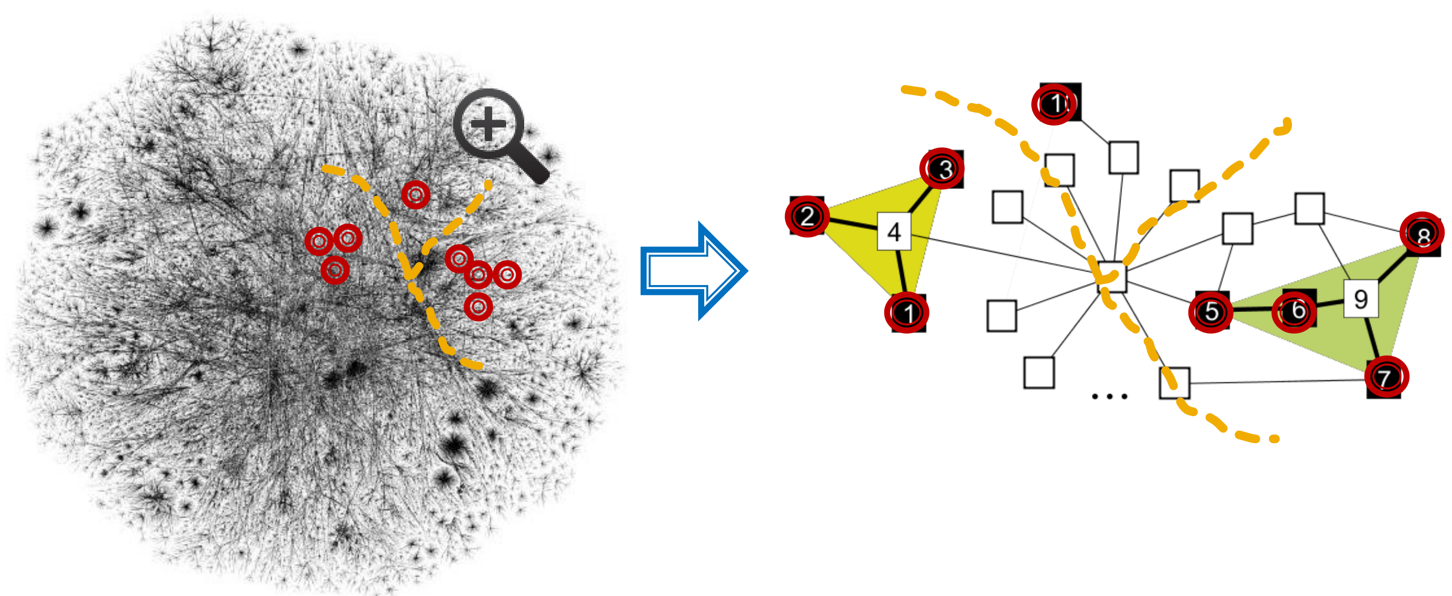
Problem

- Given
  - a **large graph G**
  - a **handful of nodes S** marked by an external process
- What can we say?
  - are **S** close by?
  - are **S** segregated?
  - how many groups do they form?
- How can we connect them?
  - with "simple" paths
  - who are "good" connectors?



Our approach

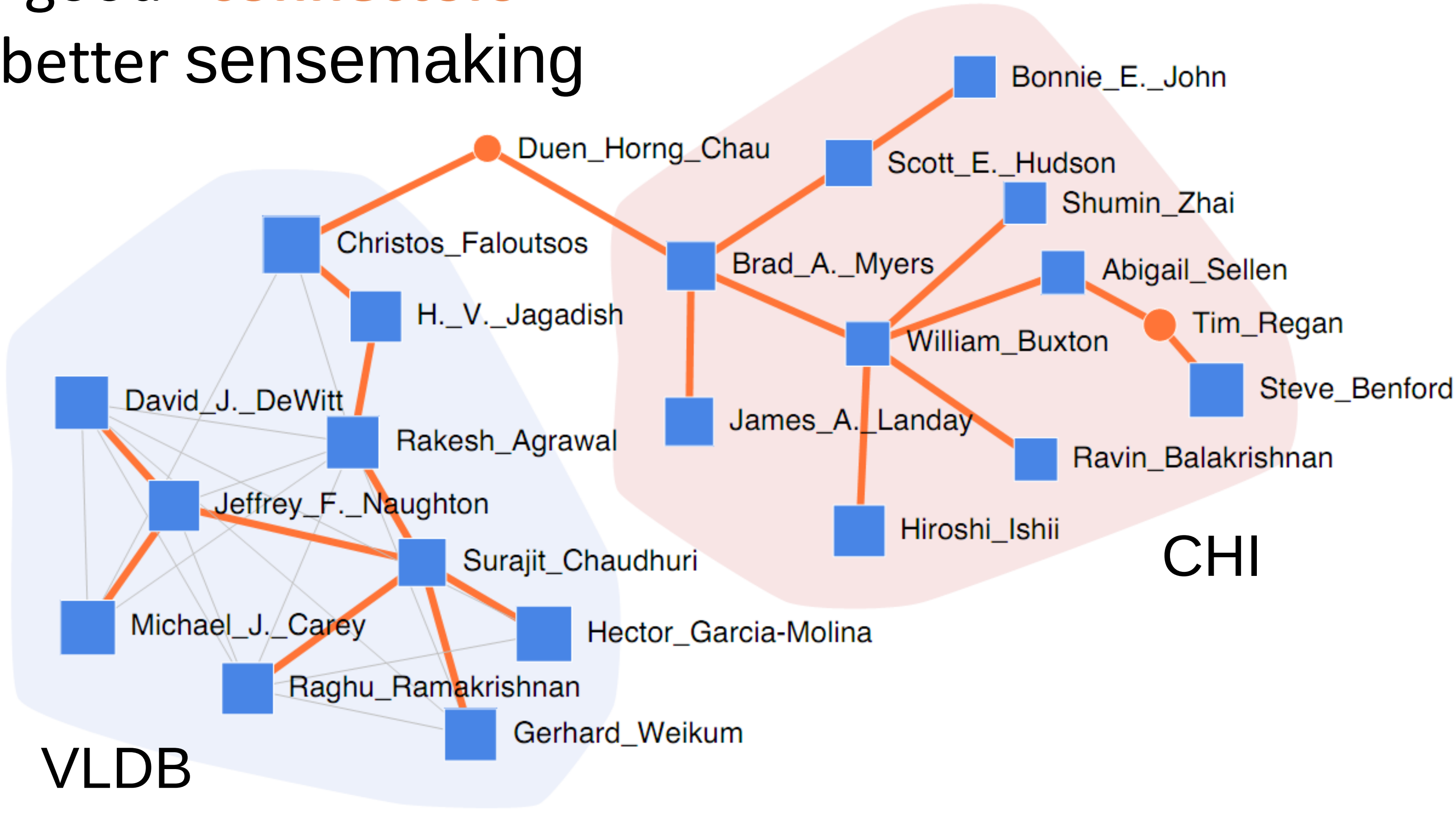
- Use the network structure to explain **S**
- Partition **S** into **groups** of nodes, such that:
  - "simple" **paths** connect the nodes in each **group**,
  - nodes in different groups are "not easily reachable"



- Use the Minimum Description Length principle
  - Best partitioning requires the "least number of bits"

Example

- “Simple” connection **pathways**
  - “good” **connectors**
  - better sensemaking

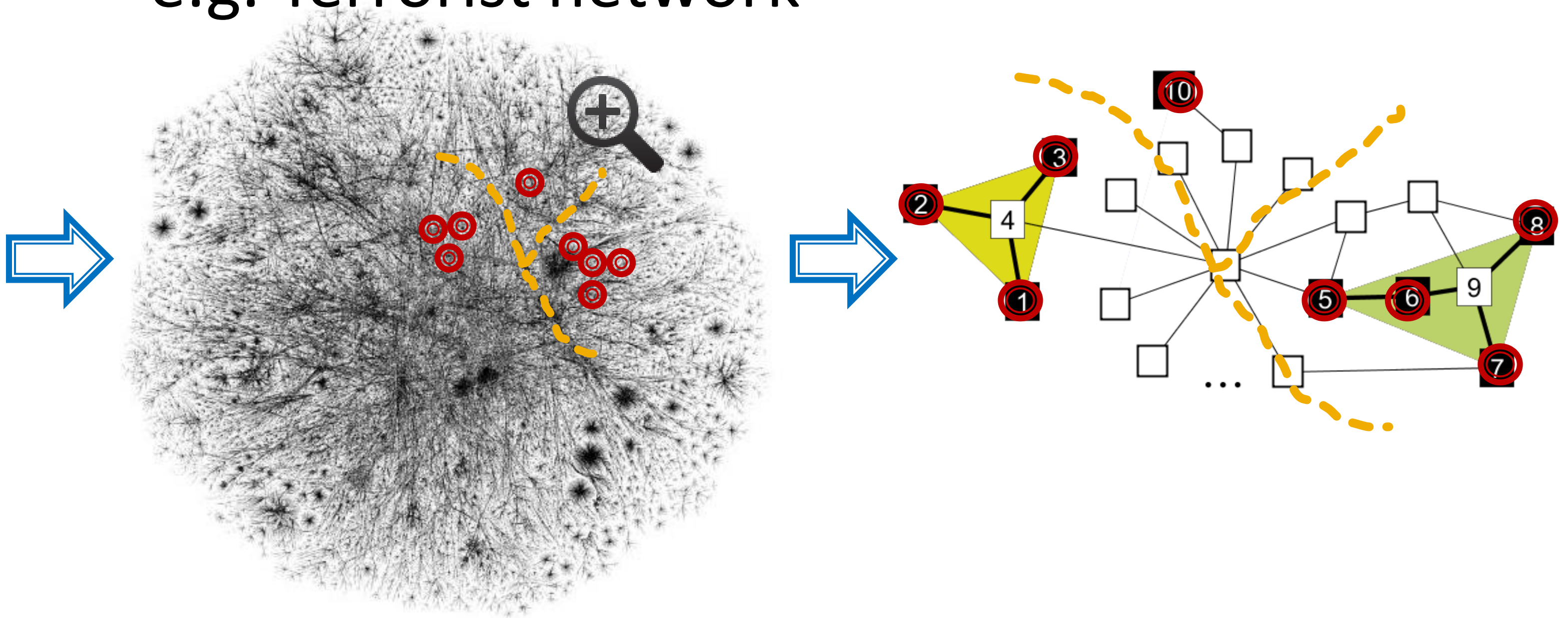


Applications

- 1. Graph anomaly description/summarization

e.g. Terrorist network

Top-k anomalies



✓ Summarize top-k node anomalies by groups

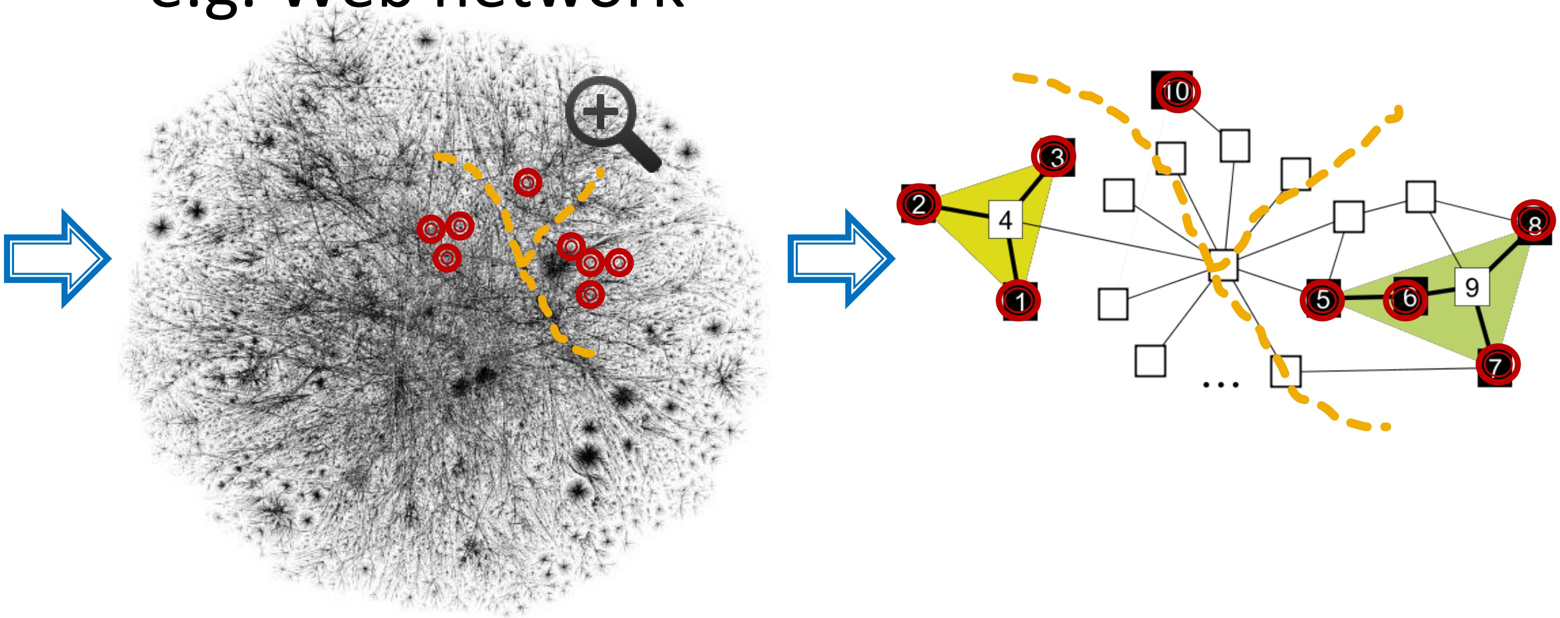
✓ Find connections/connectors among groups

Applications

- 2. Query summarization

e.g. Web network

Top-ranked Web pages



✓ Summarize top-k query pages by groups

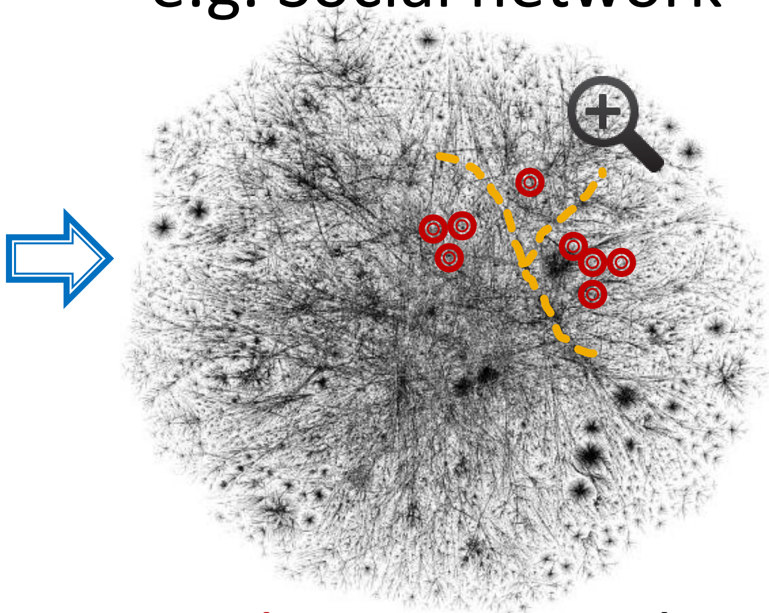
✓ Find connections/connectors among groups

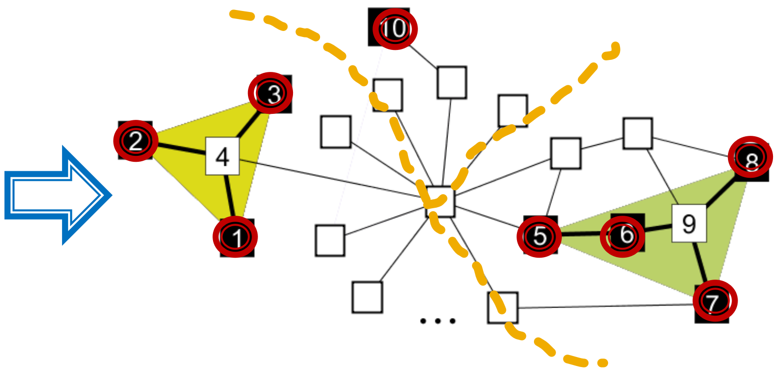
Applications

- 3. Understanding dynamic events on graphs

e.g. Social network

Affected people





Group people s.t. network structure can be associated with the spread of event

✓ within groups (number of points of infection)

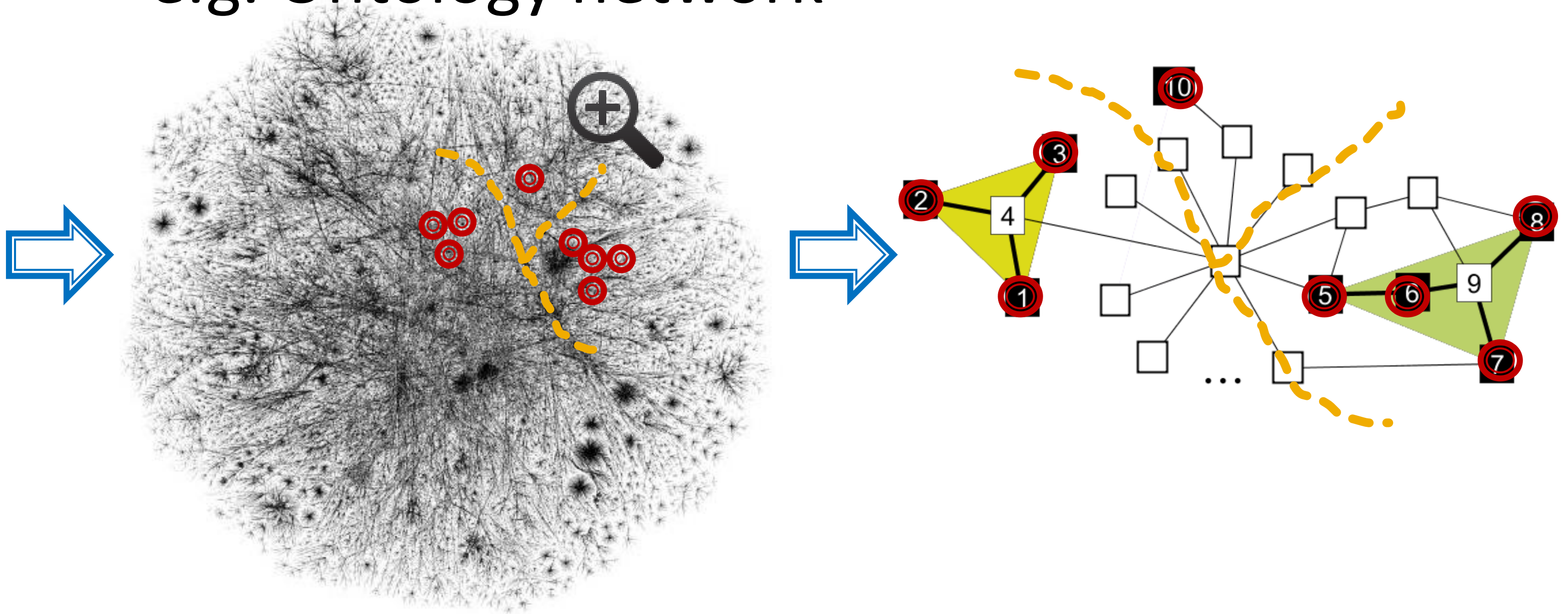
✓ but not quite across groups

Applications

- 4. Understanding semantic coherence

e.g. Ontology network

Set of words



✓ Summarize words by semantically coherent groups

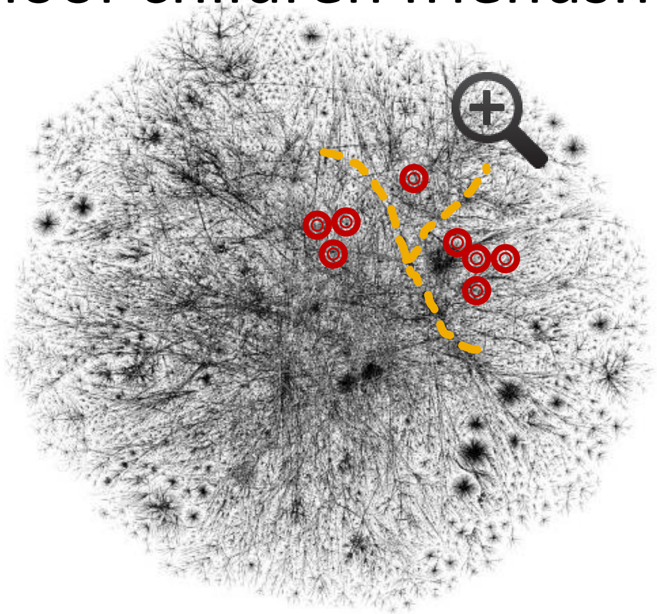
✓ Find connectors (other relevant words) among groups

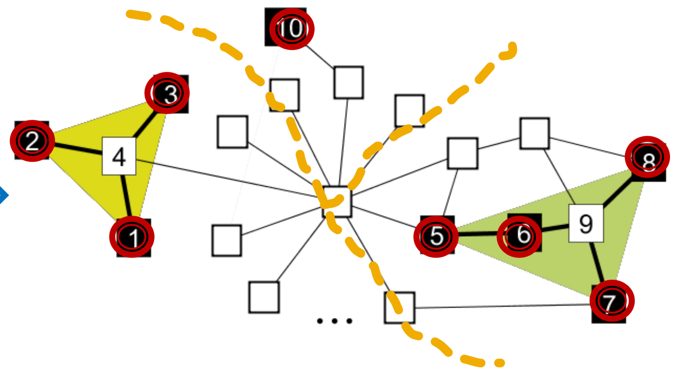
Applications

- 5. Understanding segregation (social science)

e.g. School-children friendship network

Students with attributes of interest





✓ Summarize students by their social "circles"

✓ Study groups (and groups within groups)

Roadmap

- Problem description
- Approach
- Applications
- Problem formulation
  - Problem definition
  - MDL intuition
  - Objective formulation
- Algorithms
- Experiments

Problem (formally)

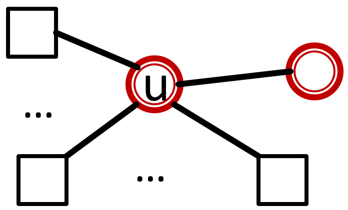
Problem Definition Given a graph $G = (V, E)$ and a set of marked nodes $M \subseteq V$

Problem 1. Optimal partitioning Find a coherent partitioning P of M. Find the optimal number of partitions $|P|$.

Problem 2. Optimal connection subgraphs Find the minimum cost set of subgraphs connecting the nodes in each part $p_i \in P$ efficiently.

Objective formulation (intuition)

- Our key idea is to use an encoding scheme
  - Imagine a sender and a receiver. Assume:
  - Both sender and receiver know graph structure G
  - Only sender knows the set of marked nodes M
  - Goal of sender:
    - transmit to the receiver the info. of which nodes are marked, using as few bits as possible.
- Why would encoding work?
  - Naïvely: encode ID of each marked node with $\log |V|$ bits
  - Better: exploit "close-by" nodes, restart for farther nodes



$2 \log |V|$ vs. $\log |V| + \log(d(u))$

Objective formulation (intuition)

- We think of encoding as
  - hopping from node to node to encode close-by nodes
  - and flying to a new node to encode farther nodes
- until all marked nodes are encoded. (hence Dot2Dot)

- Simplicity (or the description length) of connection graph T (which is a **tree**) determined by:
  - number of unmarked nodes we visit
  - how easily per visited node we can identify which edge to follow next;
    - nodes with (very) high degree make the path more complex

Objective function

DETAILS

minimize
P, T_i

$$L(P, M \mid G) = L(|P|) + \sum_i L(p_i)$$

NP-hard

(reduces from the Steiner tree problem)

- encode #partitions $L(|P|) = \log |V|$
- encode each part:

$$L(p_i) = \log |V| + L(t) + \log |T| + \log \binom{|T|}{||T||}$$

root node

spanning tree t of p_i

number of marked nodes in p_i

identities of marked nodes

- encoding of tree of each part:

$$L(t) = L_{\mathbb{N}}(|t| + 1) + \log \binom{d(v_t)}{|t|} + \sum_j^{|t|} L(b(t, j))$$

#branches of node t

identities of branch nodes

recursively encode all tree nodes

Roadmap

- Problem description
- Approach
- Applications
- Problem formulation
- Algorithms
  - Graph transformation
  - Finding bounded paths
  - Connected components
  - Minimum arborescences
  - Level-k trees
- Experiments

Algorithms - preliminaries

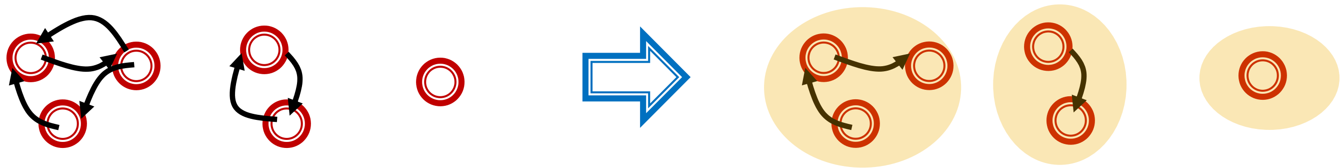
- Graph transformation
  - Given undirected unweighted G(V,E),
  - We transform it into directed weighted G'(V,E,W)
    - w(u,v) = log d(u) and w(v,u) = log d(v)

Given G', problem becomes: find *the set of trees* with minimum total cost on the marked nodes.

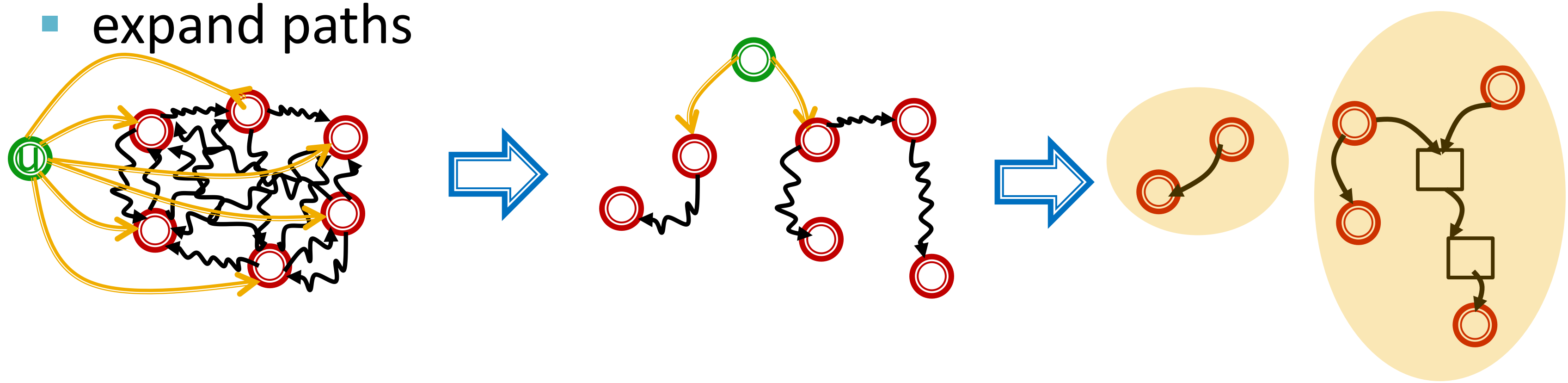
- Finding bounded-length paths
  - (multiple) short paths of length up to log |V| between marked nodes in G'
  - employ BFS-like expansion

Algorithms

- Connected components (CC)
  - find induced subgraph(s) on marked nodes in G'
  - find minimum cost directed tree(s)

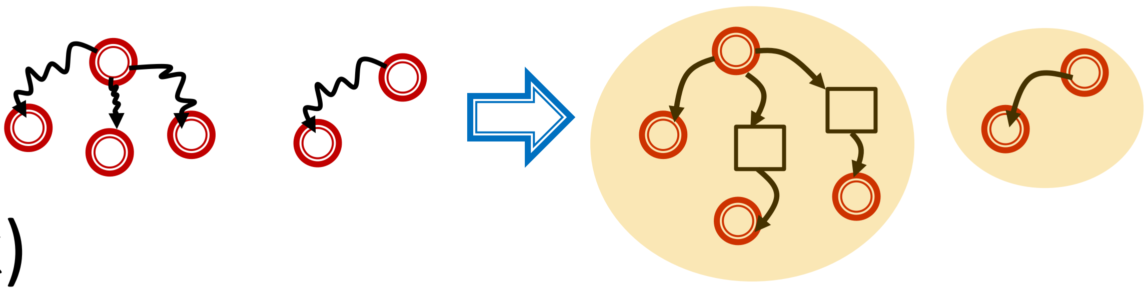


- Minimum arborescences (ARB)
  - construct transitive closure graph CG (with bounded paths)
  - add universal node u with out-edges w(u,m) = log |V|
  - find minimum cost directed tree(s), remove u
  - expand paths

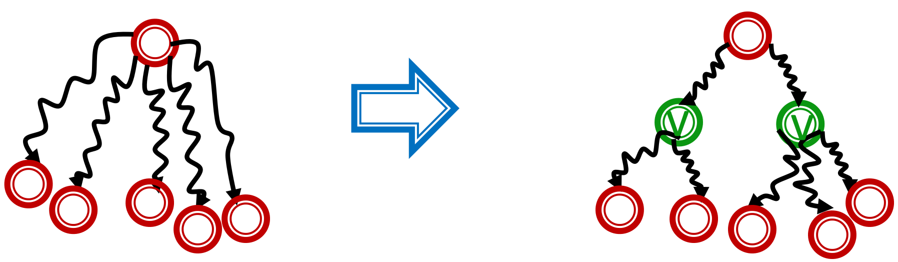


Algorithms

- Level-1 trees (L1)
  - find minimum cost depth-1 trees in CG
  - expand paths



- Level-k trees (Lk)
  - refine level-(k-1) trees by finding intermediate node v's
  - such that total cost (i.e. cost from root r to each v + costs of subtrees rooted at v's) is less

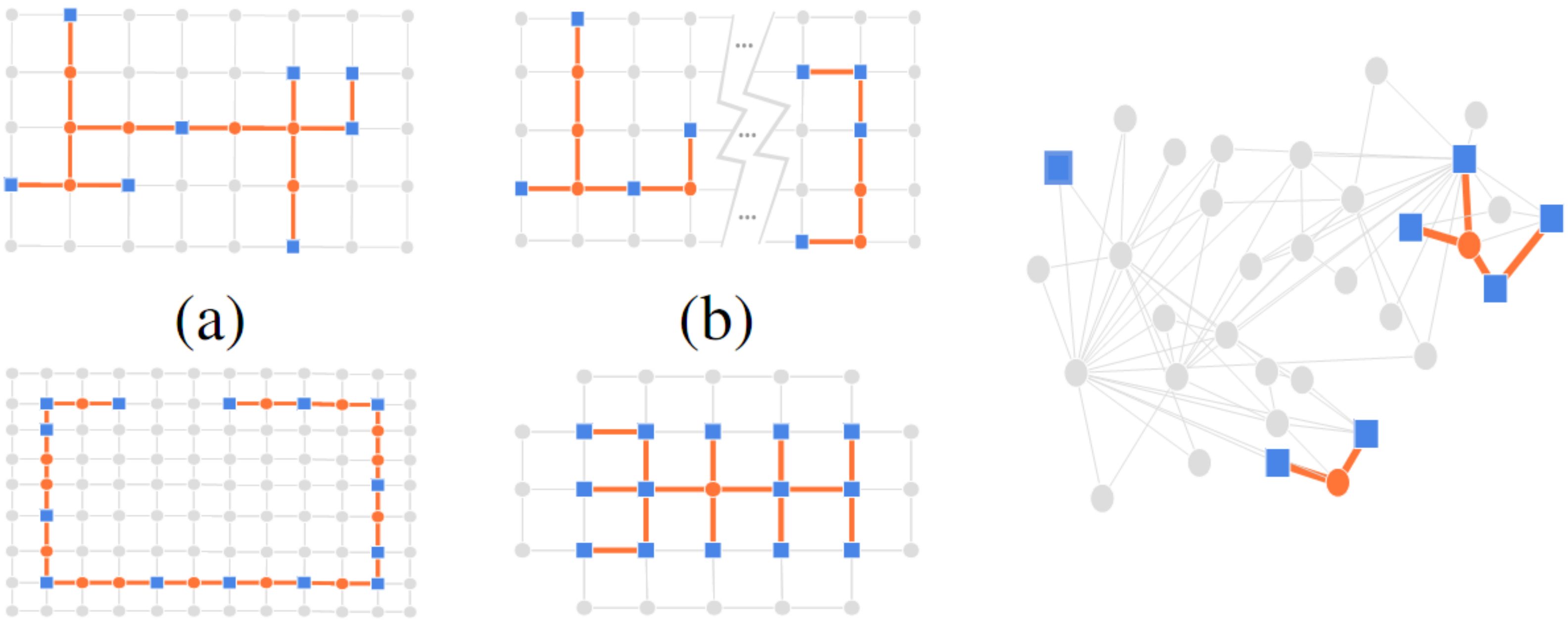


Roadmap

- Problem description
- Approach
- Applications
- Problem formulation
- Algorithms
- Experiments

Experiments

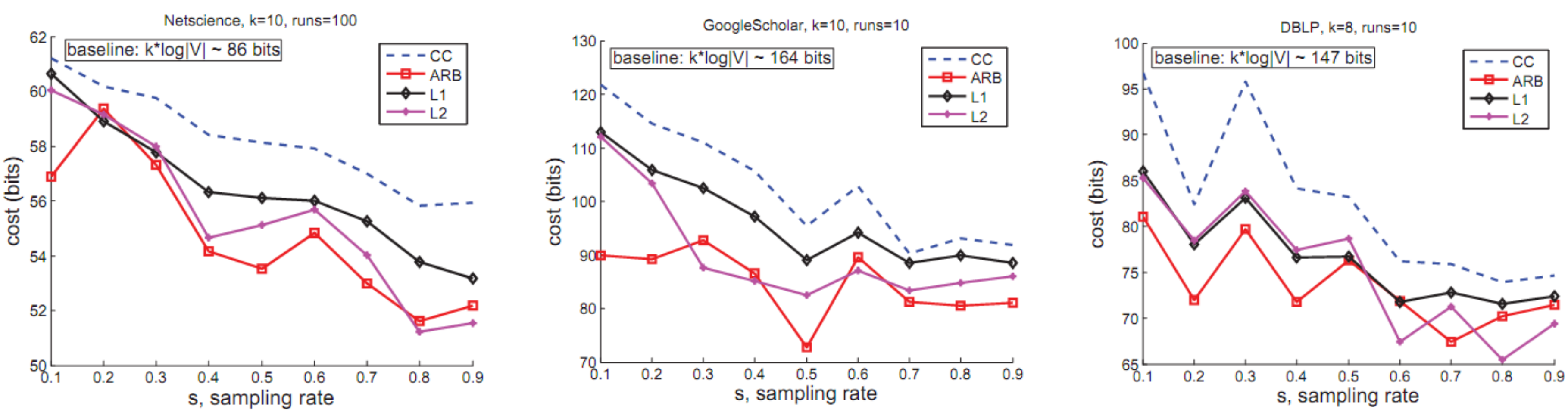
- Synthetic examples



(a) (b) (c) (d) (e)

Experiments

- Comparing the algorithms
- Real networks: Netscience, GoogleScholar, DBLP
- Random walk sampling to mark k nodes:
  - pick a random node, visit its k'<k neighbors, mark them with prob. s, pick a random node already visited

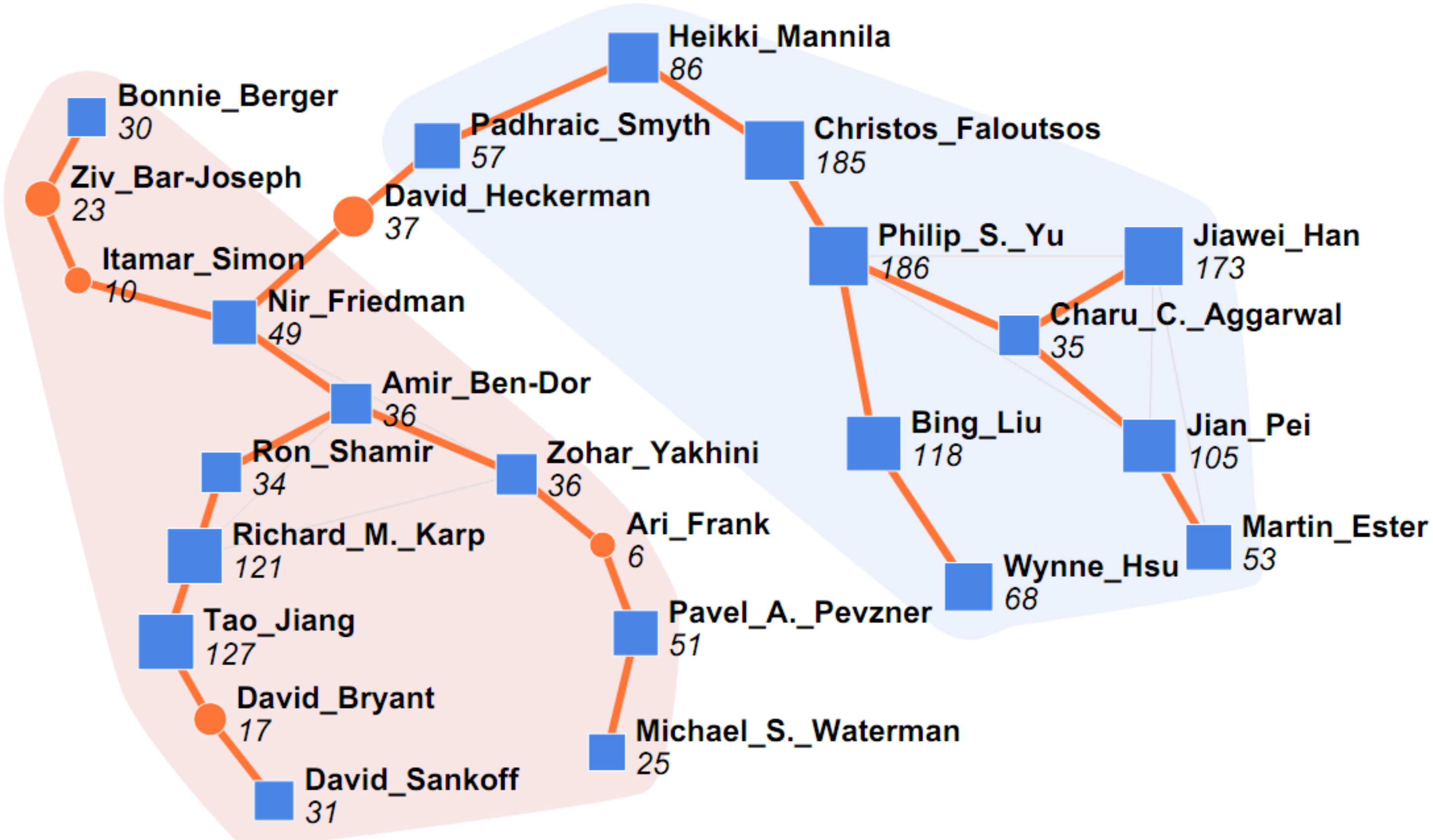


(a) *Netscience* (b) *GScholar* (c) *DBLP*

More separated ← s → More close-by

Experiments

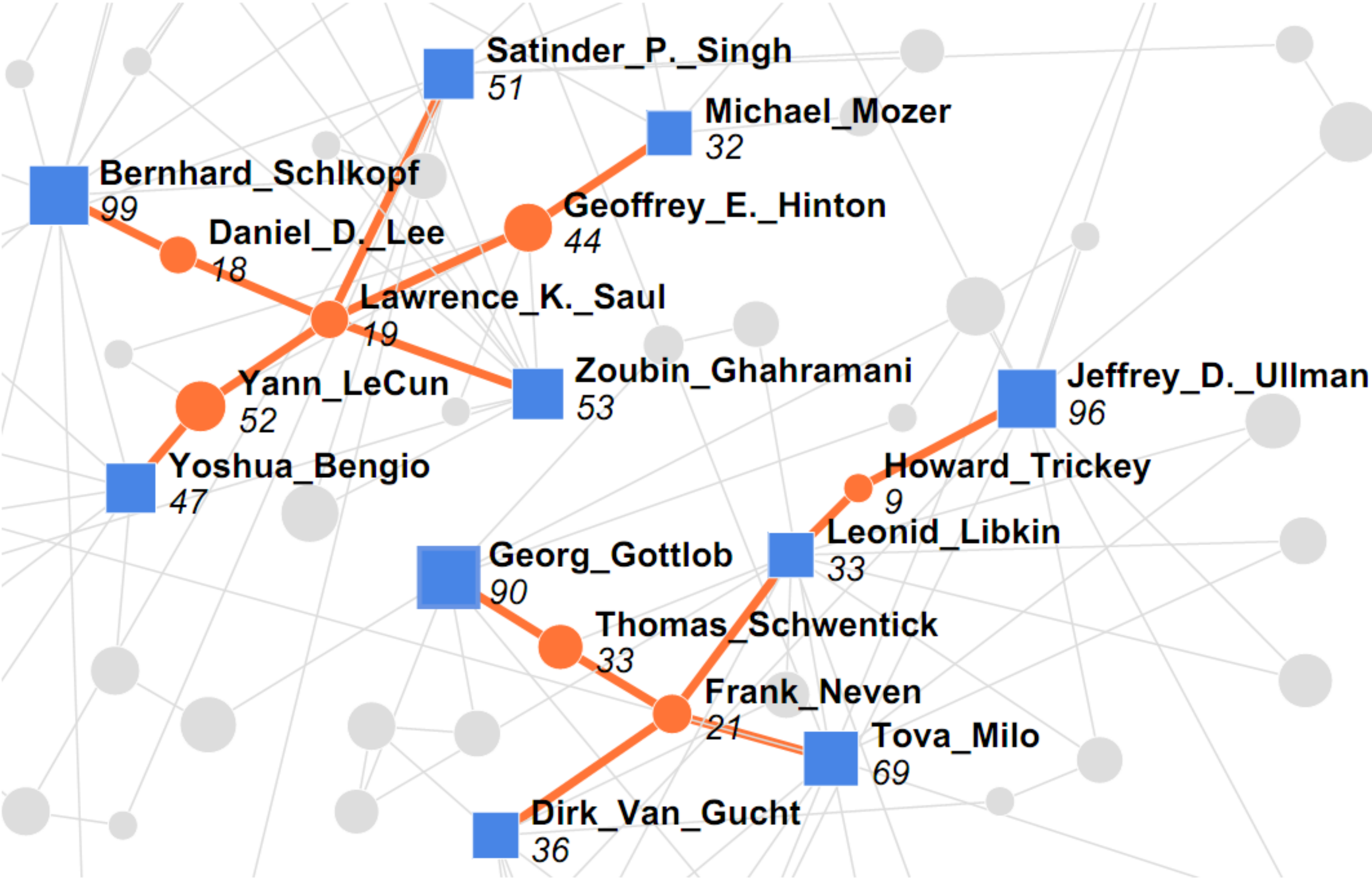
- Case studies on DBLP



(a) *DBLP*: RECOMB vs. KDD

Experiments

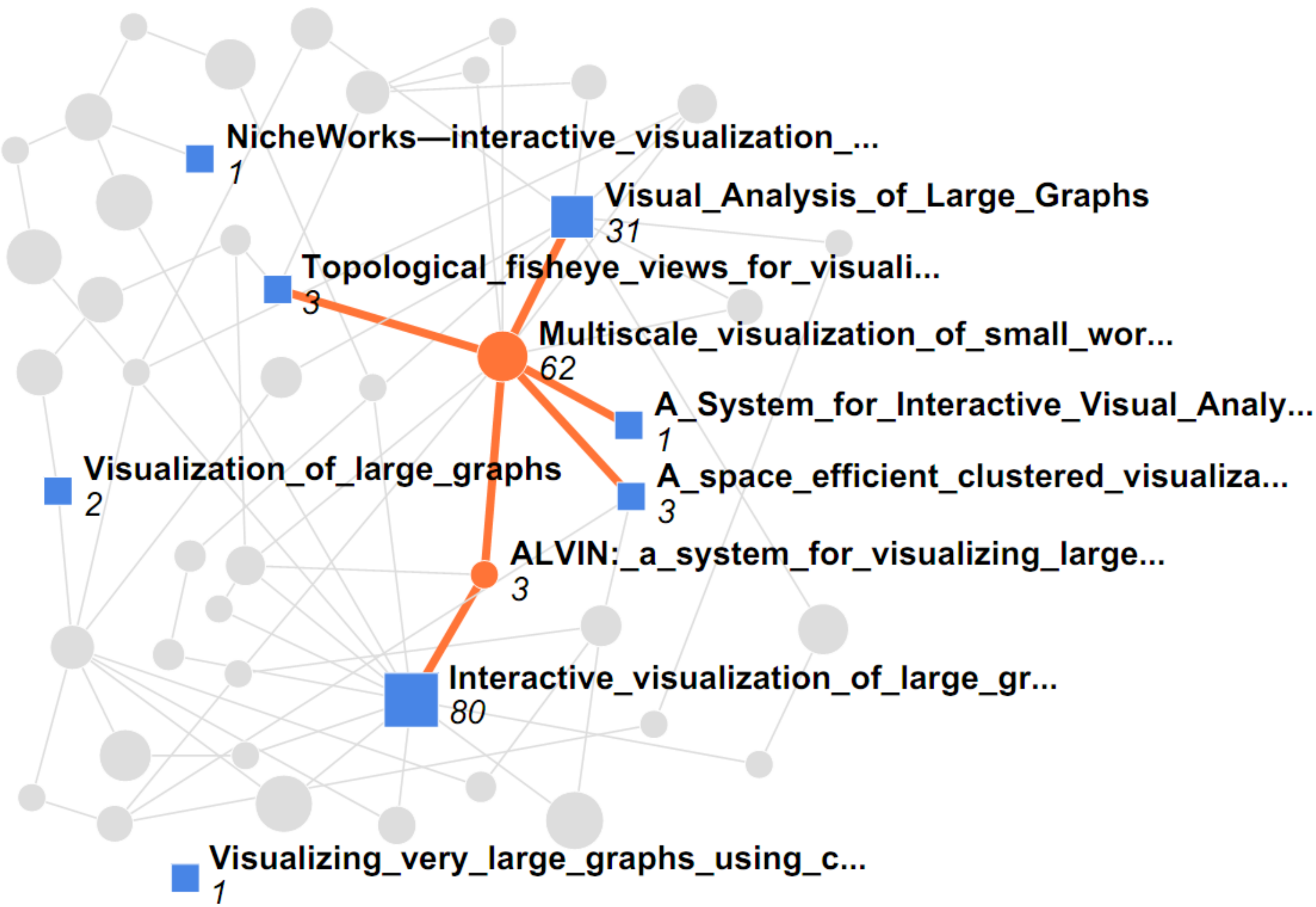
- Case studies on DBLP



(b) *DBLP*: NIPS vs. PODS

Experiments

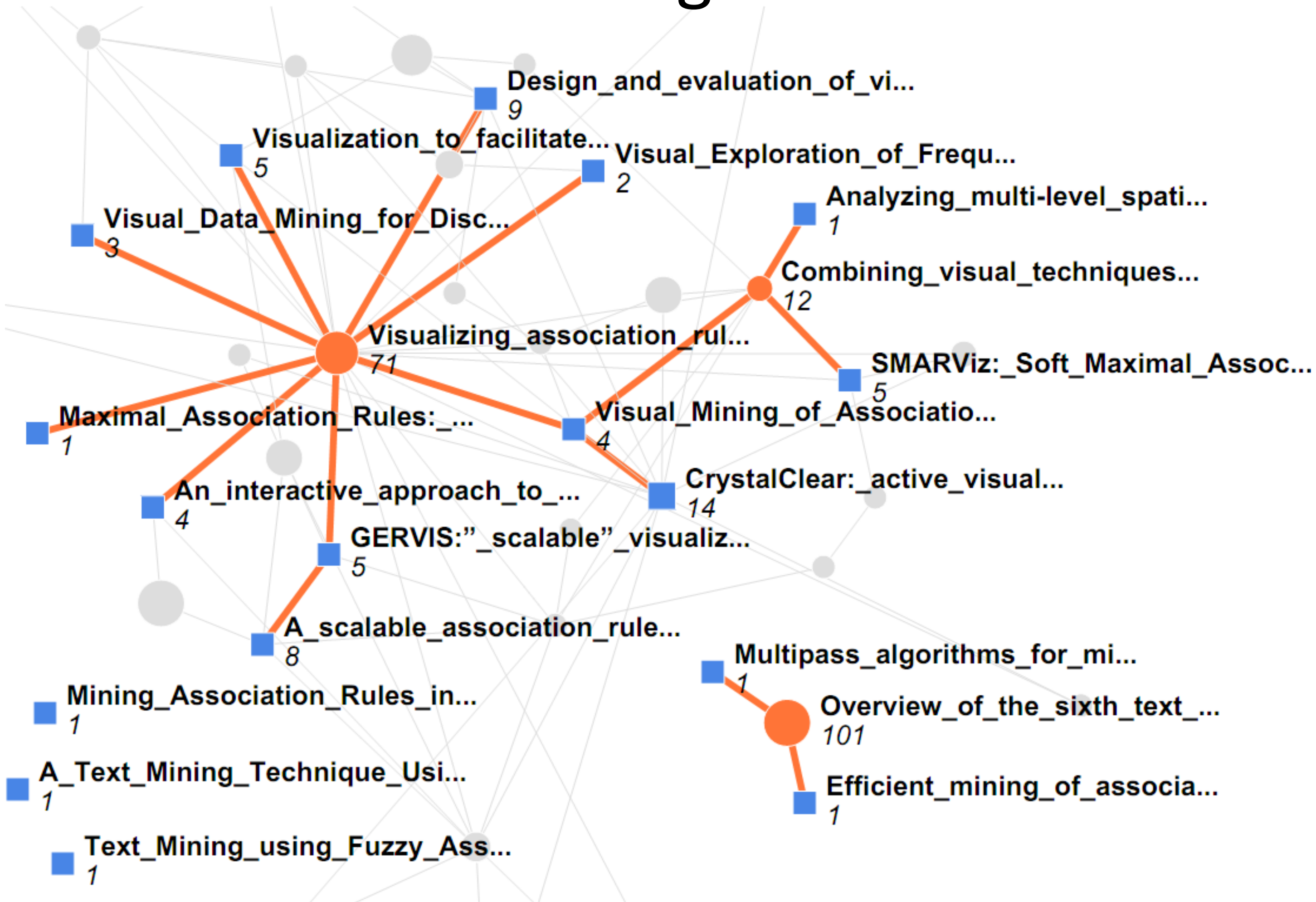
- Case studies on GoogleScholar



(a) *GScholar*: ‘large graphs’, ‘visual’

Experiments

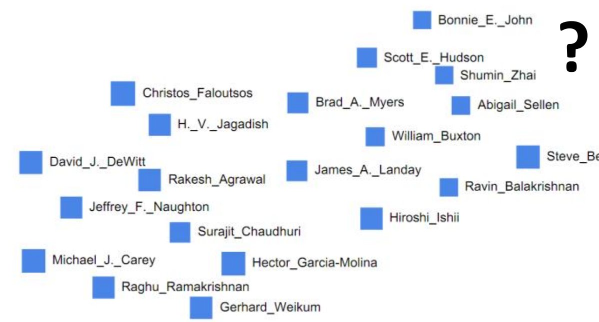
- Case studies on GoogleScholar

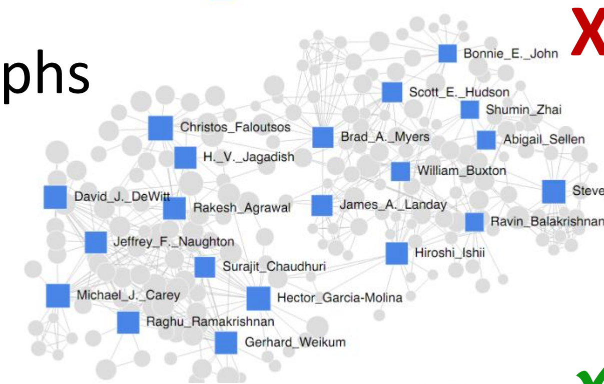


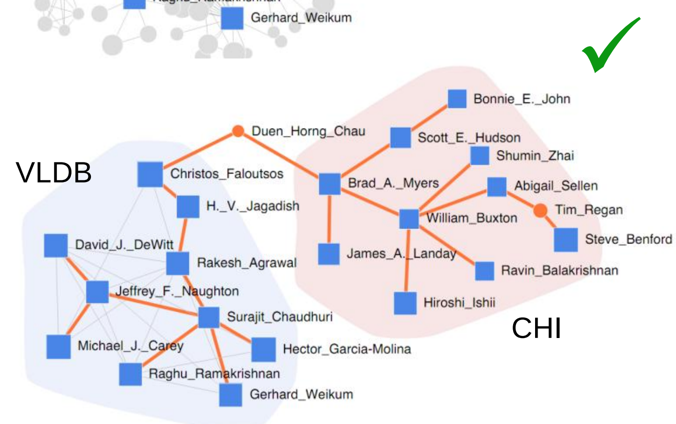
(b) *GScholar*: 'association rule', 'visual', 'text'

Summary

- **Dot2Dot**: A principled framework to "describe" a set of marked nodes in large graphs
- Many applications in the wild
  - Anomaly description/summarization
  - Query summarization
  - Understanding dynamic events on graphs
  - Understanding semantic coherence
  - Segregation studies
  - …
- MDL formulation
- NP-hardness
- Fast algorithms
- Experiments on real graphs







Thank you!

Leman Akoglu

leman@cs.stonybrook.edu

www.cs.stonybrook.edu/~leman