2018 IEEE International Conference on Data Mining

Summarizing Graphs at Multiple Scales: New Trends



Danai Koutra
University of Michigan

Jilles Vreeken
CISPA Helmholtz Center for Information Security

Francesco Bonchi
ISI Foundation

Roadmap

- 1:30-1:45pm Introduction [Jilles]
- 1:45-2:50pm **Network-level Summaries** [Francesco]
- 2:55-3:20pm Multi-network Summaries [Danai]
- 3:20-3:40pm *break*
- 3:40-4:05pm Multi-network Summaries [Danai]
- 4:10-4:40pm Node-level Summaries [Jilles]
- 4:40-4:50pm Conclusion [Jilles]

Part I:
Network-level Summaries

Francesco Bonchi
Jilles Vreeken

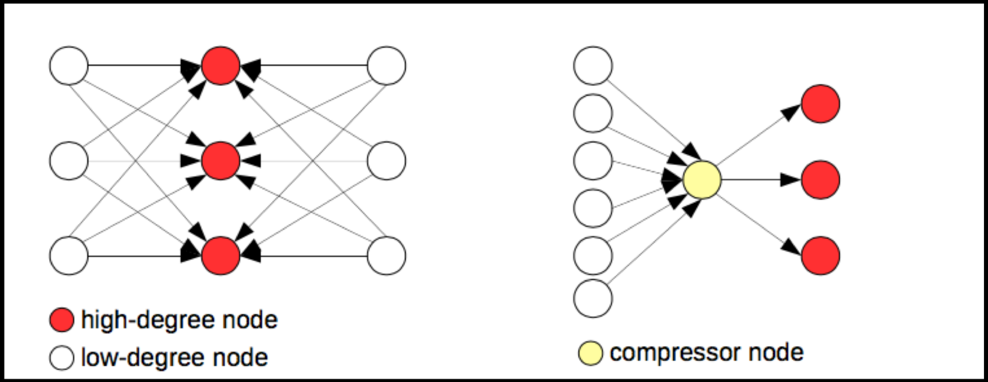
Graph Dedensification

Intuition: redundancy around high-degree nodes

Main Idea: Compress their neighborhoods

- compressor nodes

Used for exact answers to pattern matching queries



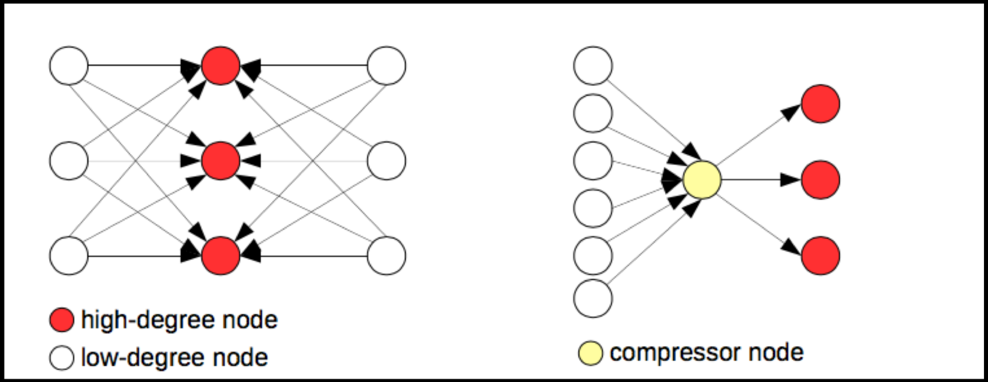
[Maccioni and Abadi, '16]

Graph Dedensification: Beyond MDL

Guarantees on speedup: precondition

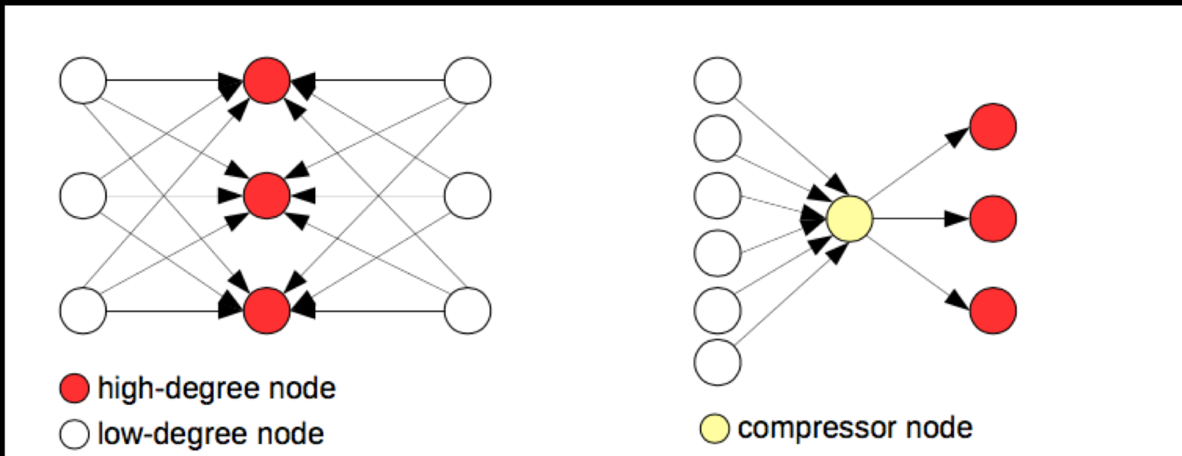
- H: set of high-degree nodes
- M: other nodes
- add compressor node if every node in M has a directed edge to each node in H

$$M \longrightarrow H$$



[Maccioni and Abadi, '16]

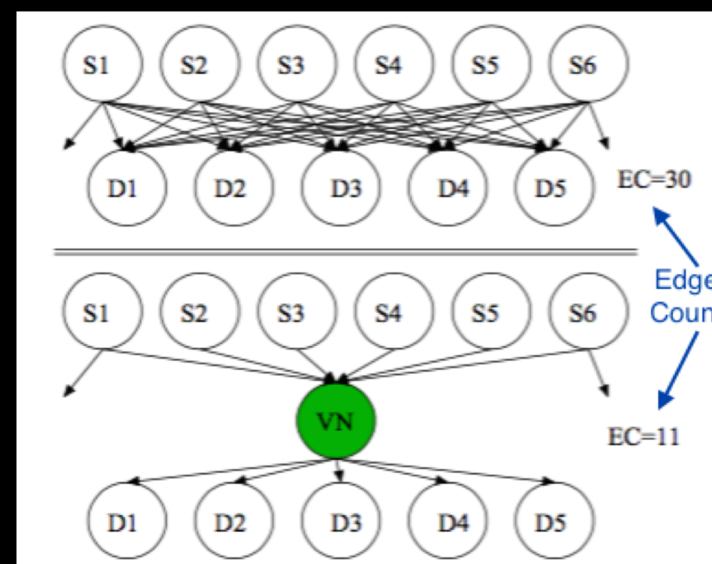
Dedensification vs. Virtual Node Compression



Dedensification

Pattern-matching Queries *with guarantees*

Goal: speedup queries



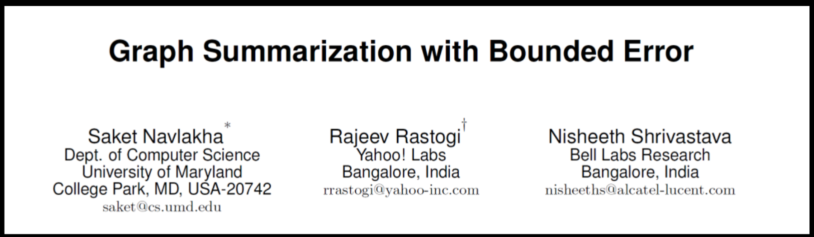
Virtual Node Compression [Buehrer '08]

Community-related queries

Goal: compression

Network-level summarization

SIGMOD 08



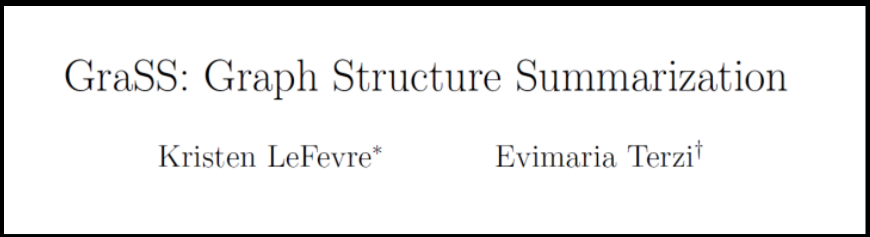
Graph Summarization with Bounded Error

Saket Navlakha[*] Dept. of Computer Science University of Maryland College Park, MD, USA-20742 saket@cs.umd.edu

Rajeev Rastogi[†] Yahoo! Labs Bangalore, India rrastogi@yahoo-inc.com

Nisheeth Shrivastava Bell Labs Research Bangalore, India nisheeths@alcatel-lucent.com

SDM 10



GraSS: Graph Structure Summarization

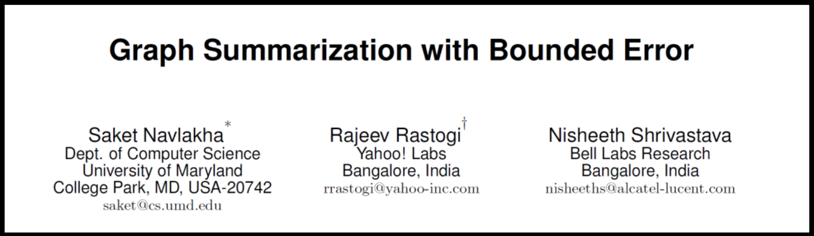
Kristen LeFevre[*] Evimaria Terzi[†]

- summarize in supernodes (set of nodes) and superedges (set of edges)
- follow the MDL principle

- lossless, or lossy with bounded error
- edge corrections

- lossy
- densities
- number of supernodes predefined
- answer queries directly on the summary (expected-value semantics)

[Navlakha et al. ’08, LeFevre & Terzi, ‘10]



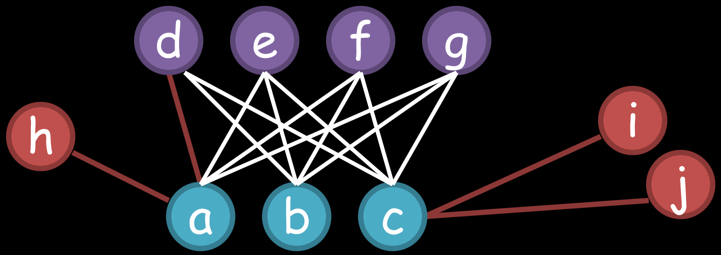
Graph Summarization with Bounded Error

Saket Navlakha[*]
Dept. of Computer Science
University of Maryland
College Park, MD, USA-20742
saket@cs.umd.edu

Rajeev Rastogi[†]
Yahoo! Labs
Bangalore, India
rrastogi@yahoo-inc.com

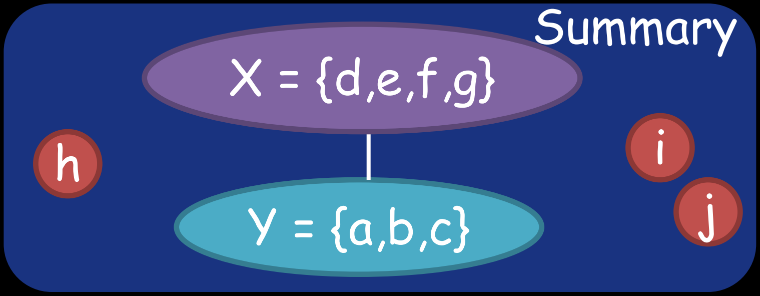
Nisheeth Shrivastava
Bell Labs Research
Bangalore, India
nisheeths@alcatel-lucent.com

Cost = 14 edges



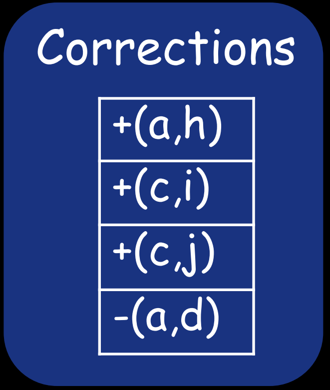
Compression possible (S)

- many nodes with similar neighborhoods
- collapse these into supernodes (clusters) and the edges into superedges
  - ✧ bipartite subgraph of two supernodes and a superedge
  - ✧ clique to supernode with a "self-edge"



Correct mistakes (C)

- most superedges are not complete
  - ✧ nodes don't have exact same neighbors: friends in social networks
- remember corrections
  - ✧ negative edges, not present in superedges
  - ✧ positive edges, not counted in superedges



Corrections

+(a,h)
+(c,i)
+(c,j)
-(a,d)

Cost = 5
(1 superedge + 4 corrections)

Minimize overall cost $= S + C$

[Navlakha et al. '08]

Representation Structure $R = (S, C)$

Summary $S(VS, ES)$

- supernode v represents set of nodes A_v
- superedge (u, v) represents all pairs of edges $\pi_{uv} = A_u \times A_v$

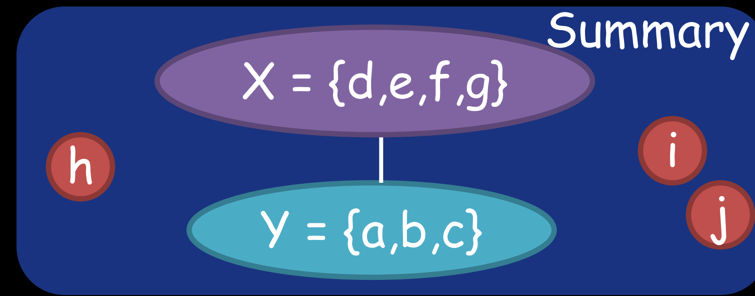
Corrections C: $\{(a, b);\ a\ and\ b\ are\ nodes\ of\ G\}$

Supernodes are key, edges/corrections easy

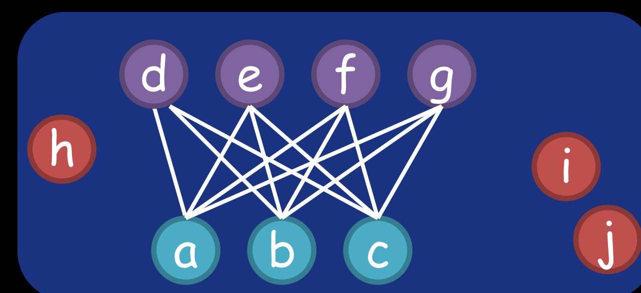
- A_{uv} actual edges of G between A_u and A_v
- cost with $(u, v) = 1 + |\pi_{uv} - A_{uv}|$
- cost without $(u, v) = |A_{uv}|$
- choose minimum, decides whether (u, v) in S

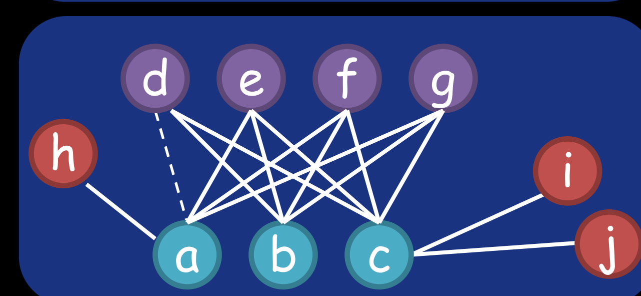
Reconstructing the graph from R

- for all superedges $(u, v) \in S$ insert all pairs π_{uv}
- for all +ve corrections $+(a, b)$, insert (a, b)
- for all -ve corrections $-(a, b)$, delete (a, b)



C = {+(a,h), +(c,i), +(c,j), -(a,d)}





[Navlakha et al. '08]

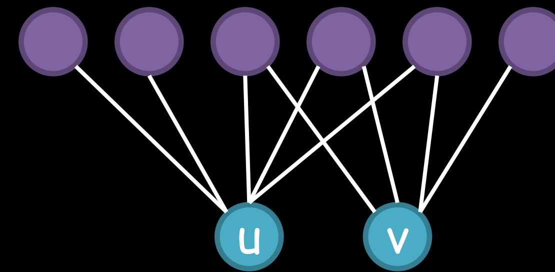
Greedy

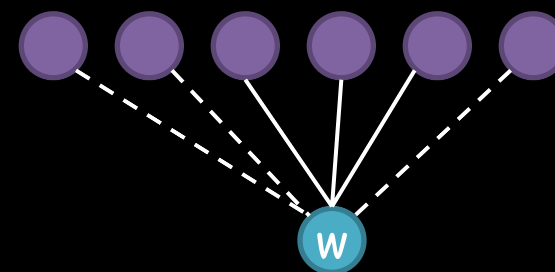
Cost of merging supernodes u and v
into single supernode w

- recall: cost of a superedge (u, x):
  $$c(u, x) = \min\{|\pi_{vx} - A_{vx}| + 1, |A_{vx}|\}$$
- c_u = sum of costs of all its edges = $\Sigma_x c(u, x)$
- $s(u, v) = (c_u + c_v - c_w)/(c_u + c_v)$

Main idea:
recursive bottom-up merging of supernodes

- if $s(u, v) > 0$, merging u and v reduces the cost
- normalize the cost: remove bias towards high degree nodes
- creating supernodes is key: superedges and corrections can be computed later





c_u = 5; c_v =4

c_w = 6 (3 edges, 3 corrections)

s(u,v) = 3/9

[Navlakha et al. '08]

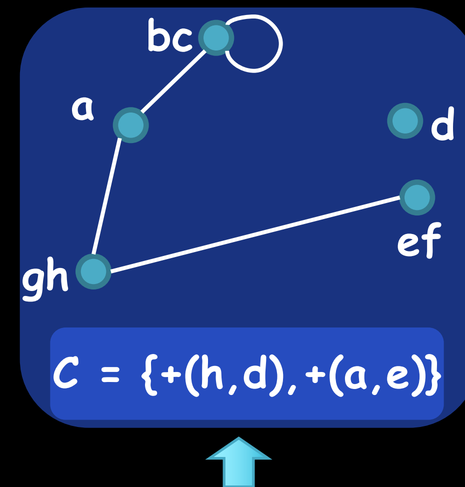
Greedy

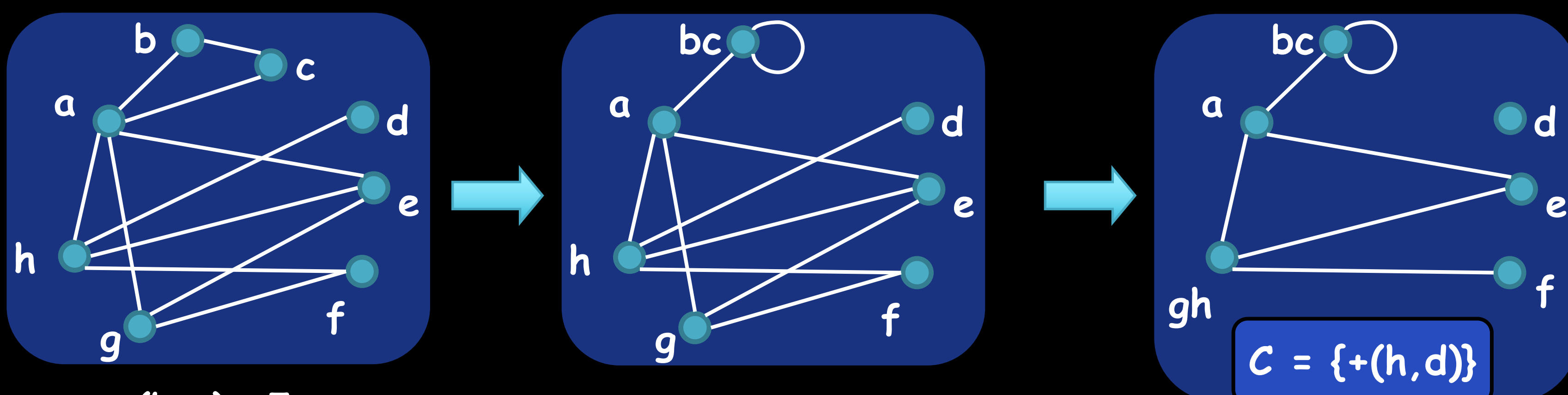
Recall $s(u, v) = (c_u + c_v - c_w)/(c_u + c_v)$

GREEDY algorithm

- start with $S = G$
- at every step, pick pair with max $s(.)$ value, merge
- if no pair has positive $s(.)$ value, stop

Cost reduction: 11 to 6





s(b,c)=.5
[c_b = 2; c_c=2; c_{bc}=2]

s(g,h)=3/7
[c_g = 3; c_h=4; c_{gh}=4]

s(e,f)=1/3
[c_e = 2; c_f=1; c_{ef}=2]

[Navlakha et al. '08]

Randomized

GREEDY is slow

- needs to find the pair with (globally) max $s(.)$ value
- processes all pair of nodes at a distance of 2-hops
- every merge changes costs of all pairs with N_w

Main idea: light-weight randomized procedure

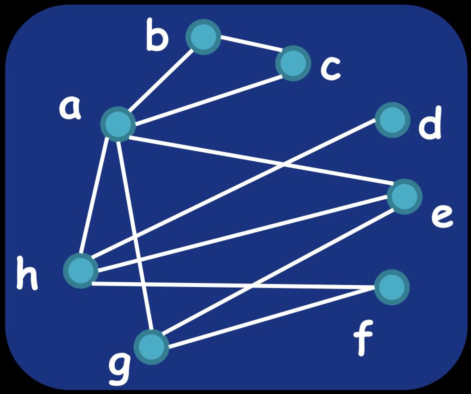
- instead of choosing the globally best pair, (randomly) choose node u
- merge the best pair containing $\boldsymbol{u}$

[Navlakha et al. '08]

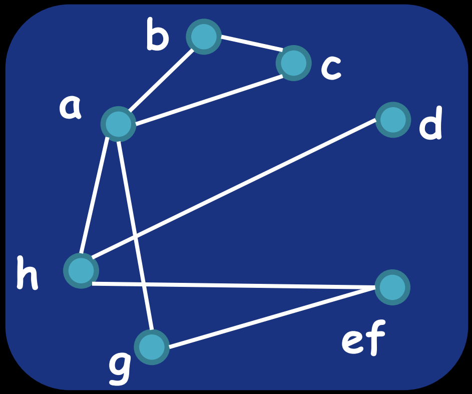
Randomized

RANDOMIZED algorithm

- unfinished set $U = VG$
- at every step, randomly pick a node u from U
- find that node v with $\max s(u,v)$ value
- if $s(u,v) > 0$ then
  - ✧ merge u and v into w
  - ✧ put w in U
- else remove u from U
- repeat till U is not empty

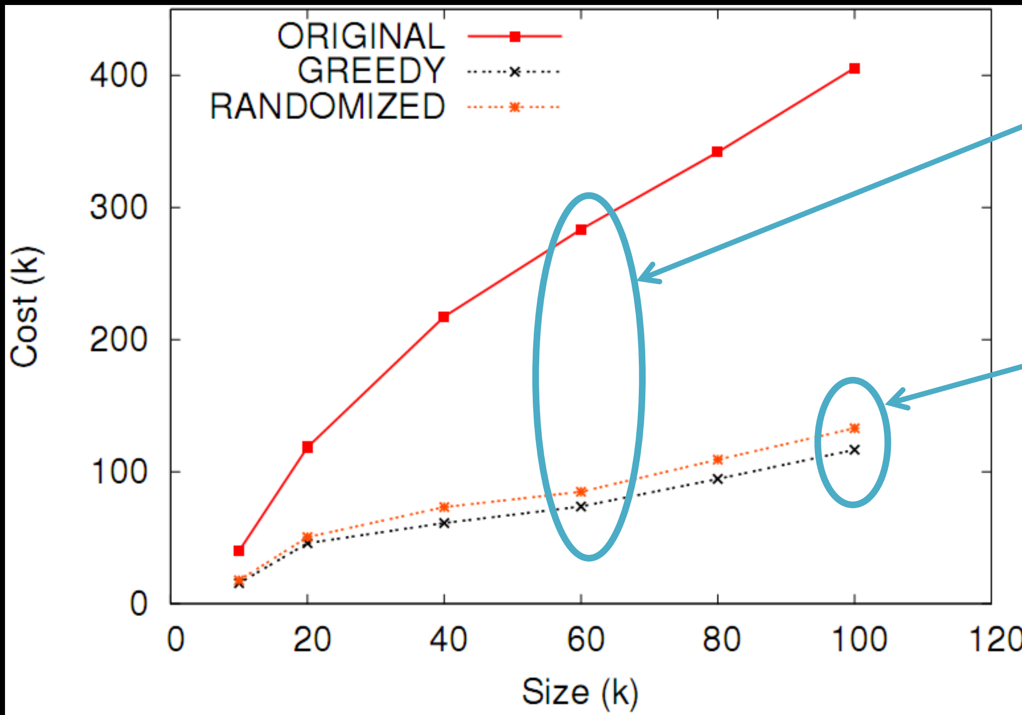


Picked e; s(e,f)=3/5
[c_e = 3; c_f=2; c_{ef}=3]



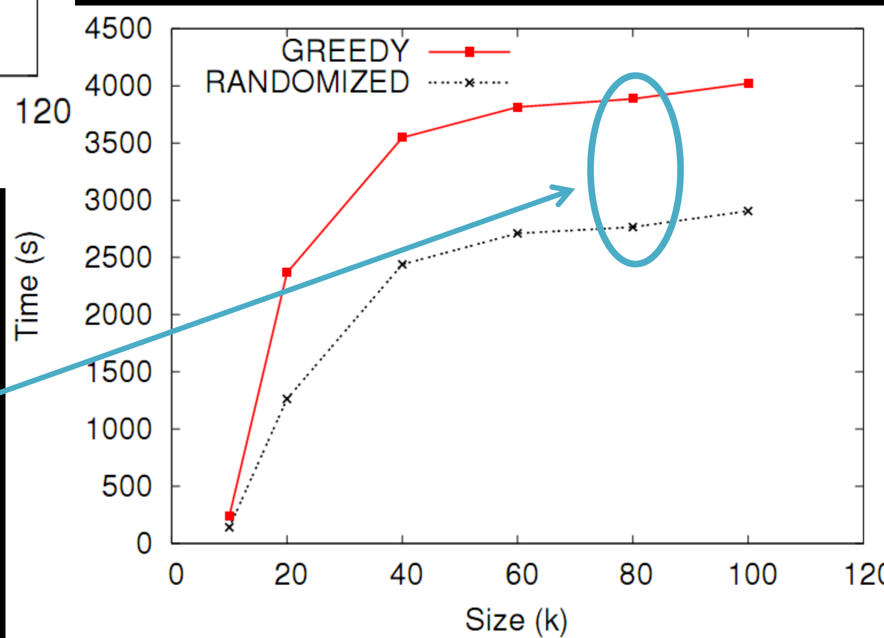
C = {+(a,e)}

[Navlakha et al. '08]



Reduces the cost up to 40%

Cost of GREEDY 20% lower than RANDOMIZED



RANDOMIZED is 60% faster than GREEDY

[Navlakha et al. '08]

Approximate Representation R_ϵ

Approximate representation

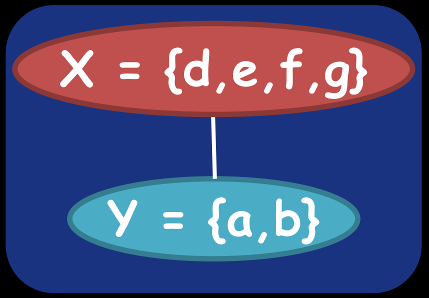
- recreating *exactly* is not always necessary
- reasonable approximation enough to compute communities, anomalous traffic patterns, etc.
- use approximation to get further reduction

Generic Neighbor Query

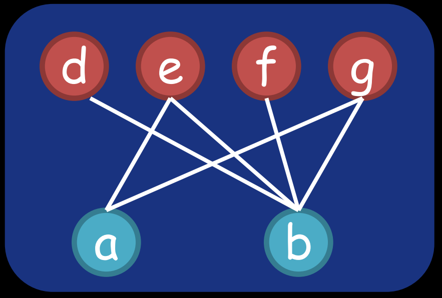
- given node v, find its neighbors $N_v \in G$
- Apx-nbr set N'_v estimates N_v with ϵ-accuracy
- bounded error:
  $error(v) = |N'_v \setminus N_v| + |N_v \setminus N'_v| < \epsilon\,|N_v|$
- number of neighbors added or deleted is at most ϵ-fraction of the true neighbors

Intuition for computing R_ϵ

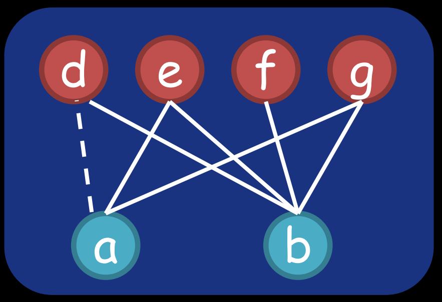
- deleting correction (a, d) adds error for a and d
- from exact representation R, remove (maximum) corrections s.t. ϵ-error guarantees still hold



C = {-(a,d), -(a,f)}

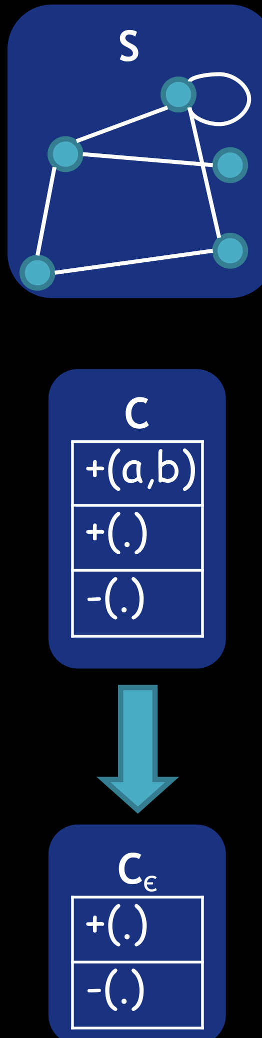


For $\epsilon = .5$, we can remove one correction of a



[Navlakha et al. '08]

Computing approx. representation



Reducing size of corrections

- correction graph H:
  for every correction $(a, b) \in C$, add edge (a, b) to H
- removing (a, b) reduces size of C, but adds error of 1 to a and b
- recall bounded error: $error(v) = |N'_v \setminus N_v| + |N_v \setminus N'_v| < \epsilon |N_v|$
- implies we can remove up to $b_v = \epsilon |N_v|$ edges incident on v
- maximum cost reduction: remove subset M of E_H of max size s. t. M has at most b_v edges incident on v

Same as the b-matching problem

- find matching $M \subset EG$ s.t. at most b_v edges incident on $v \in M$
- for all $b_v = 1$, traditional matching problem
- solvable in time $O(mn^2)$ [Gabow-STOC-83]
  - (for graph with n nodes and m edges)

[Navlakha et al. '08]

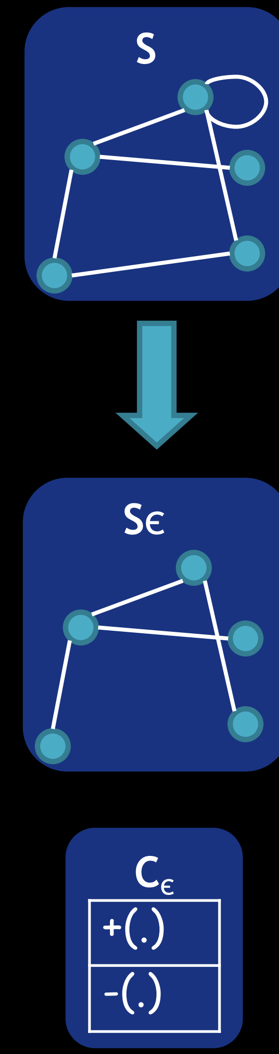
Computing approx. representation

Reducing size of summary

- removing superedge (a, b) is bulk removal of all pair edges π_{uv},
- However, each node in A_u and A_v has **different** b value
- … does not map to clean matching-type problem

A GREEDY approach

- pick superedges by increasing $|\pi_{uv}|$ value
- delete (u, v) if that doesn't violate ϵ-bound for nodes in $A_u U A_v$
- if there is correction (a, b) for π_{uv} in C, we cannot remove (u, v); since removing (u, v) violates error bound for a or b



[Navlakha et al. '08]

APXMDL

Compute the $R(S, C)$ for G

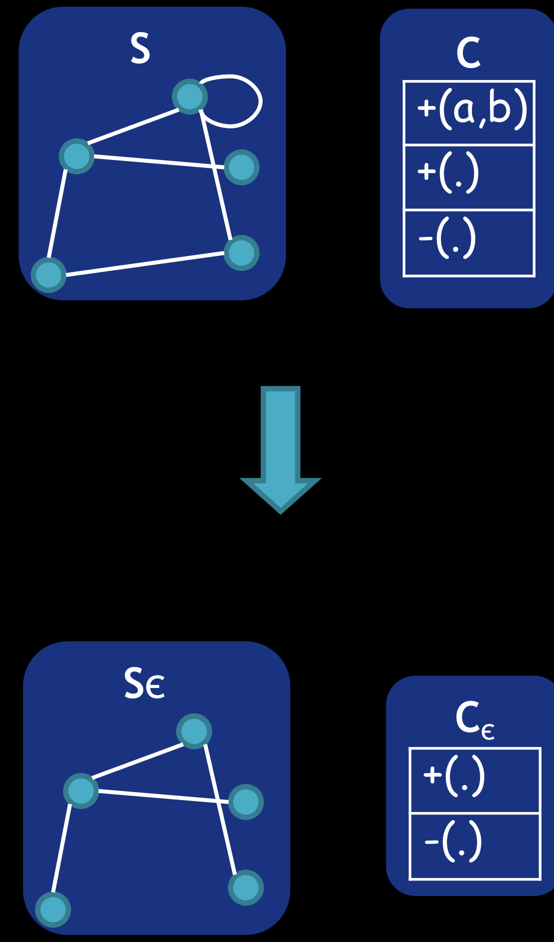
Find C_ϵ

- compute H, with $V_H = C$
- find maximum b-matching M for H; $C_\epsilon = C - M$

Find S_ϵ

- pick superedges (u, v) in S without correction in C_ϵ ascending in $|\pi_{uv}|$
- remove (u, v) if that doesn't violate ϵ-bound for any node in $A_u \cup A_v$

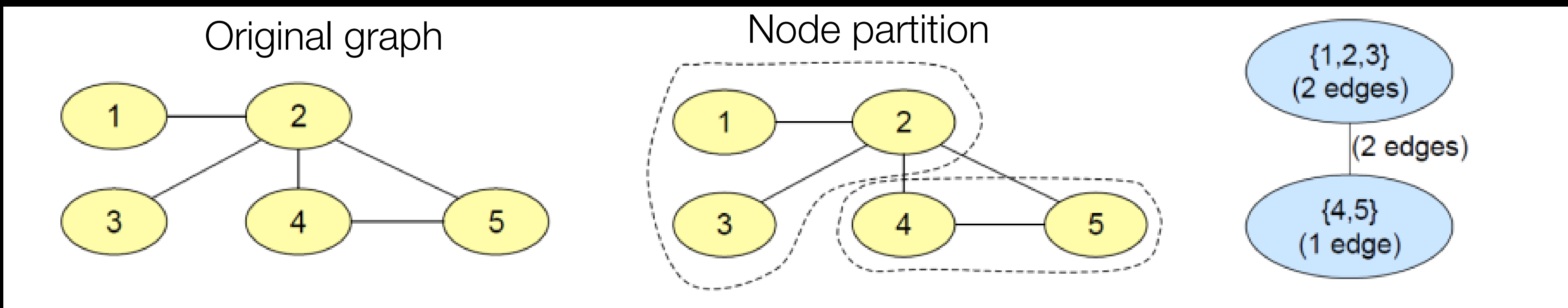
Apx-representation $R_\epsilon = (C_\epsilon, S_\epsilon)$



[Navlakha et al. '08]

GraSS: Graph Structure Summarization

Kristen LeFevre* Evimaria Terzi†



	1	2	3	4	5
1	0	1	0	0	0
2	1	0	1	1	1
3	0	1	0	0	0
4	0	1	0	0	1
5	0	1	0	1	0

	1	2	3	4	5
1	0	2/3	2/3	1/3	1/3
2	2/3	0	2/3	1/3	1/3
3	2/3	2/3	0	1/3	1/3
4	1/3	1/3	1/3	0	1
5	1/3	1/3	1/3	1	0

[Lefevre & Terzi ’10]

	1	2	3	4	5
1	0	2/3	2/3	1/3	1/3
2	2/3	0	2/3	1/3	1/3
3	2/3	2/3	0	1/3	1/3
4	1/3	1/3	1/3	0	1
5	1/3	1/3	1/3	1	0

Example:

Expected degree of node #2:

$$2/3 + 2/3 + 1/3 + 1/3 = 2$$

Other measures

- expected eigenvector centrality
- expected number of triangles*

Query answering

Expected adjacency matrix can be seen as a probabilistic (uncertain) graph

Queries to the original graph can be approximated directly on the summary

expected value semantics

[Lefevre & Terzi, ’10] *[Riondato et al, ‘14]

Minimize the reconstruction error

A summary is good when the expected adjacency matrix is close to original adjacency matrix

Define reconstruction error as difference between the matrices

Problem: given an integer k find a k-partiton of the nodes s.t. the corresponding summary minimizes reconstruction error.

	1	2	3	4	5
1	0	1	0	0	0
2	1	0	1	1	1
3	0	1	0	0	0
4	0	1	0	0	1
5	0	1	0	1	0

	1	2	3	4	5
1	0	2/3	2/3	1/3	1/3
2	2/3	0	2/3	1/3	1/3
3	2/3	2/3	0	1/3	1/3
4	1/3	1/3	1/3	0	1
5	1/3	1/3	1/3	1	0

$$\mathrm{RE}\left(A \mid \overline{A}\right) = \frac{1}{|V|^2} \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} \left|\overline{A}(i,j) - A(i,j)\right|$$

[Lefevre & Terzi, '10]

Greedy algorithm

GREEDY agglomerative hierarchical clustering

1) put each vertex in a separate supernode;
2) until the number of supernodes is k
    1) *merge the two supernodes* whose merging minimizes the reconstruction error;
3) output the resulting k supernodes;

Main limitations

- no quality guarantees
- very slow

[Lefevre & Terzi, '10]

ICDM'14

Graph Summarization with Quality Guarantees

Matteo Riondato
Stanford University
rionda@cs.stanford.edu

David García-Soriano
Yahoo Labs, Barcelona, Spain
davidgs@yahoo-inc.com

Francesco Bonchi
Yahoo Labs, Barcelona, Spain
bonchi@yahoo-inc.com

***Abstract*—We study the problem of graph summarization. Given a large graph we aim at producing a concise lossy representation that can be stored in main memory and used to approximately answer queries about the original graph much faster than by using the exact representation. In this paper we study a very natural type of summary: the original set of vertices is partitioned into a small number of supernodes connected by superedges to form a complete weighted graph. The superedge weights are the edge densities between vertices in the corresponding supernodes. The goal is to produce a summary that minimizes the *reconstruction error* w.r.t. the original graph. By exposing a connection between graph summarization and geometric clustering problems (i.e., k-means and k-median), we develop the *first polynomial-time approximation algorithm* to compute the best possible summary of a given size.**

The GraSS algorithm presented in [1] follows a greedy heuristic resembling an agglomerative hierarchical clustering using Ward's method [3] and as such can not give any guarantee on the quality of the summary. In this paper instead, we propose efficient algorithms to compute summaries of *guaranteed quality* (a constant factor from the optimal). This theoretical property is also verified empirically: our algorithms build more representative summaries and are much more efficient and scalable than GraSS in building those summaries.

II. Problem Definition

We consider an undirected graph $G = (V, E)$ with $|V| = n$. In the rest of the paper, the key concepts are defined from the standpoint of the symmetric adjacency matrix A_G of G. We

Data Mining and Knowledge Discovery manuscript No.
(will be inserted by the editor)

Graph Summarization with Quality Guarantees

Matteo Riondato · David García-Soriano · Francesco Bonchi

DMKD

Received: date / Accepted: date

Abstract We study the problem of graph summarization. Given a large graph we aim at producing a concise lossy representation (a *summary*) that can be stored in main memory and used to approximately answer queries about the original graph much faster than by using the exact representation.

In this work we study a very natural type of summary: the original set of vertices is partitioned into a small number of supernodes connected by superedges to form a complete weighted graph. The superedge weights are the edge densities between vertices in the corresponding supernodes. To quantify the dissimilarity between the original graph and a summary, we adopt the *reconstruction error* and the *cut-norm error*. By exposing a connection between graph summarization and geometric clustering problems (i.e., k-means and k-median), we develop the *first polynomial-time approximation algorithms* to compute the best possible summary of a certain size under both measures.

We discuss how to use our summaries to store a (lossy or lossless) compressed graph representation and to approximately answer a large class of queries about the original graph, including adjacency, degree, eigenvector centrality, and triangle and subgraph counting. Using the summary to answer queries is very efficient as the running time to compute the answer depends on the number of supernodes in the summary, rather than the number of nodes in the original graph.

A preliminary version of this work appeared in the proceedings of IEEE ICDM'14 (Riondato et al, 2014).

M. Riondato
Two Sigma Investments LP, New York, NY, USA (part of the work performed during an internship at Yahoo Labs Barcelona and while affiliated to Brown University)
E-mail: matteo@twosigma.com

D. García-Soriano
Eurecat, Barcelona, Spain
E-mail: david.garcia@eurecat.com

Francesco Bonchi
ISI Foundation, Turin, Italy
E-mail: francesco.bonchi@isi.com

Overcomes GraSS limitations

- fast algorithm with constant-factor approx. guarantee
- generalize reconstruction error to l_p-reconstruction error
- consider cut-norm error
- among contributions: practical use of extreme graph theory, with cut-norm and algorithmic version of Szemerédi's Regularity Lemma.

[Riondato et al, '14, DMKD]

ALGORITHM: just cluster the rows of the adjacency matrix!

- For ℓ_p-reconstruction error, perform *ℓ_p-clustering* of the *rows of A_G* ($p = 1$: k-median, $p = 2$: k-means). If column i is in cluster j, then vertex i is in supernode V_j.

LEMMA: The summary obtained from the optimal ℓ_1 (resp. ℓ_2) clustering is a *8 (resp. 4) approximation* of the optimal summary for the ℓ_1 (resp. ℓ_2) reconstruction error.

Both k-means and k-median are *NP-hard*;

There are *constant factor approximation algorithms*;
BOTTLENECK: computing all pairwise distance for the n rows of A_G is expensive (like matrix multiplication);

SOLUTION: Use a *sketch* of the adjacency matrix with n rows and $\log n$ columns;
Incurs in additional constant error;

Even with the sketch, the approximation algorithms take time $\tilde{O}(n^2)$;
IDEA: *select $O(k)$ rows of the sketch adaptively*, compute a clustering using them;

In the end, the algorithm runs in time $\tilde{O}(m + nk)$ and obtains a constant-factor approximation.

ALGORITHM: just cluster the rows of the adjacency matrix!

Algorithm 1: Graph summarization with ℓ_p-reconstruction error

---

Input : $G = (V, E)$ with $|V| = n$, $k \in \mathbb{N}$, $p \in \{1, 2\}$
Output: A $O(1)$-approximation to the best k-summary for G under the ℓ_p-reconstruction error

```
// Create the n × O(log n) sketch matrix (Indyk, 2006)
S ← createSketch(A_G, O(log n), p)
// Select O(k) rows from the sketch (Aggarwal et al, 2009)
R ← reduceClustInstance(A_G, S, k)
// Run the approximation algorithm by Mettu and Plaxton (2003) to
obtain a partition.
P ← getApproxClustPartition(p, k, R, S)
// Compute the densities for the summary
D ← computeDensities(P, A_G)
```

return $(\mathcal{P}, D)$

[Riondato et al, '14, DMKD]

Influence-based Summarization

Influence-based summarization methods aim to discover a short representation of the influence flow in large-scale graphs.

Sparsification-based method: SPINE

Idea: keep only edges that explain the information propagation ("backbone" of influence network)

- i.e. that **maximize** the likelihood of observed data

$$\log L(G) = \sum_{\alpha \in \mathcal{A}} \log L_\alpha(G)$$
$$= \sum_{\alpha \in \mathcal{A}} \sum_{v \in V} (\log P_\alpha^+(v) + \log P_\alpha^-(v))$$

P_α^+: at least one node succeeds to influence $\boldsymbol{v}$
$\boldsymbol{P_a^-}$: all nodes fail

- assuming the Independent Cascade model
- no grouping

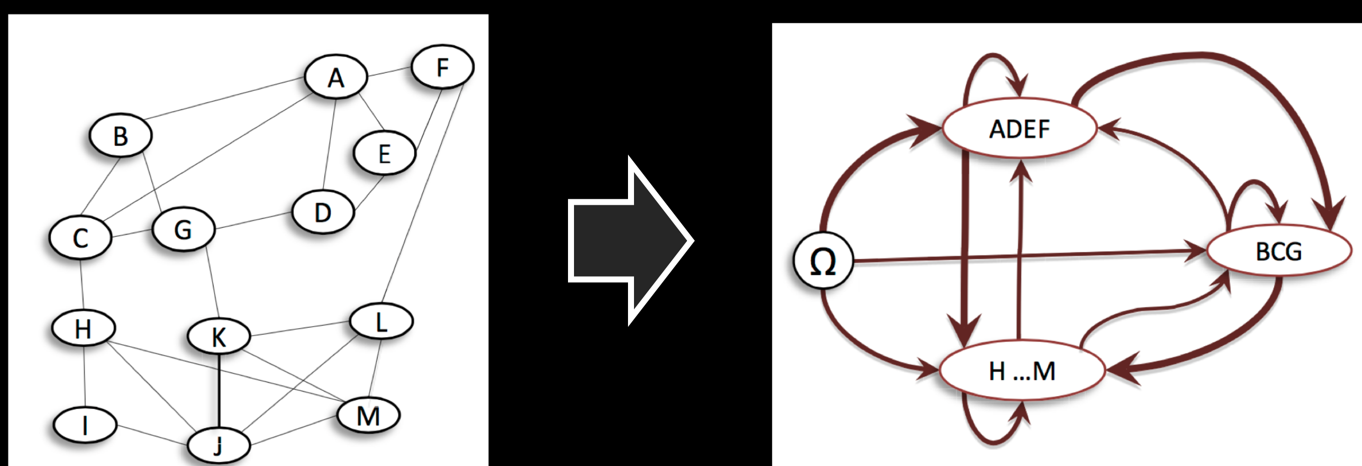
[Mathioudakis et al., '11.]

Community-level Social Influence (CSI)

Goal: summarize information propagation and social influence

- Independent Cascade model to find influence between communities (extension from nodes)

Output: Community = set of nodes that share a similar influence tendency over nodes in other communities

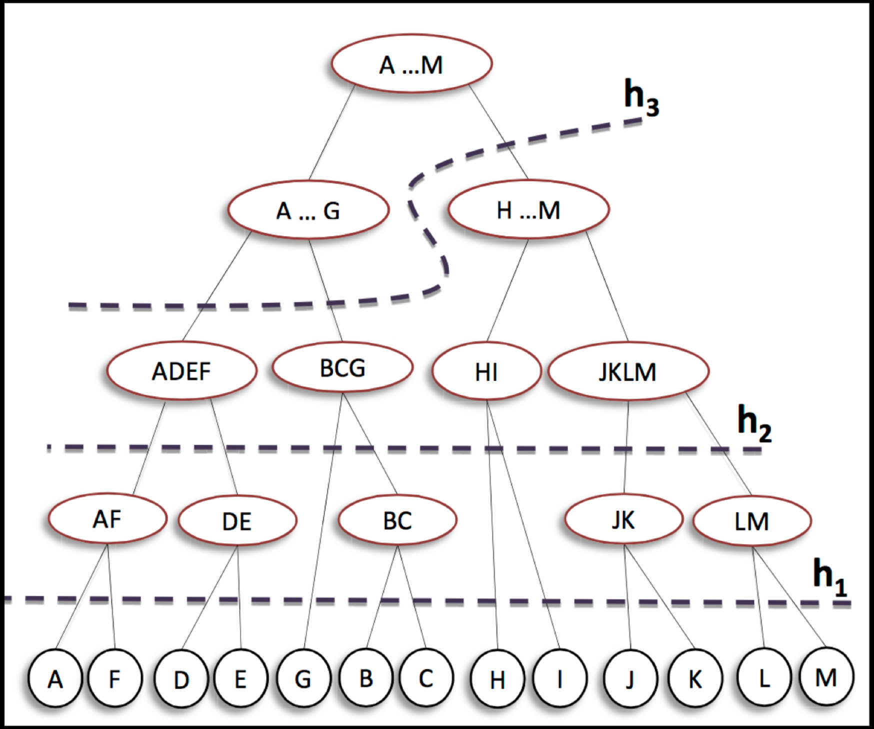


[Mehmood et al., '*13*.]

Community-level Social Influence (CSI)

Algorithm:

- recursive application of METIS for hierarchical communities
- EM algorithm to learn pairwise influence relationships
- merge two communities (with same parent)
  - ✧ MDL or BIC to select the "best" cut

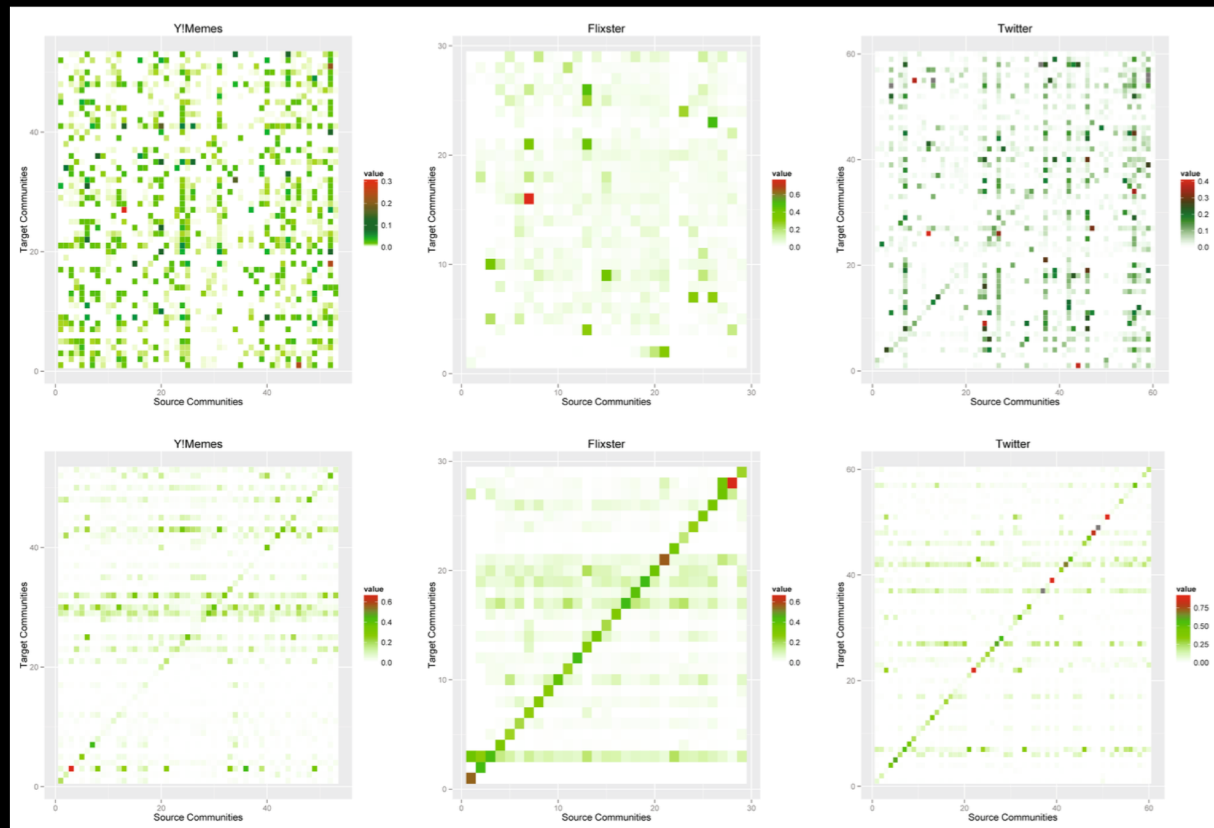


[Mehmood et al., '13.]

CSI: Y! Memes, Flixster, Twitter

Community-to-community Influence Probabilities

Social Links



- No correlation between influence and link probabilities.
- Even dense communities do not exhibit strong internal influence.

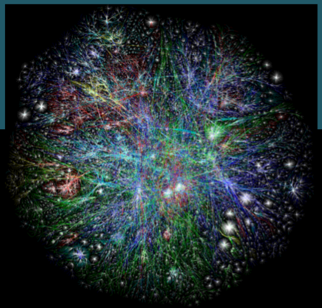
[Mehmood et al., '*13*.]

Pattern mining-based Summarization

Pattern mining techniques aim to summarize an input network via structural patterns.

(can also be combined with grouping techniques and compression)

Using Frequent Patterns



Target:

- compress web gaphs
- support community discovery

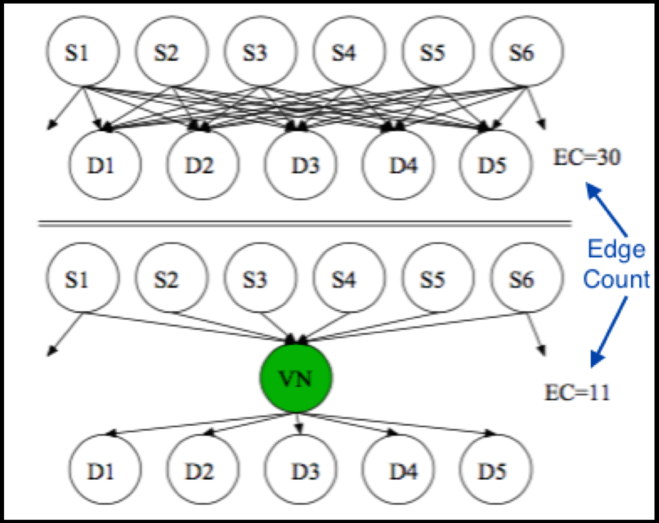
Main idea:

- Frequent pattern mining: patterns are replaced with a virtual node

Algorithm:

- Phase 1: Clustering of similar nodes (probabilistic sampling)
- Phase 2: Frequent pattern mining by casting outlinks as an itemset

Vertex Id	Outlink List
23	1,2,3,5,6,10,12,15
55	1,2,3,5
102	1,2,3,20
204	1,7,8,9
13	1,2,3,8
64	1,2,3,5,6,10,12,15
43	1,2,3,5,6,10,22,31
431	1,2,3,5,6,10,21,31,67

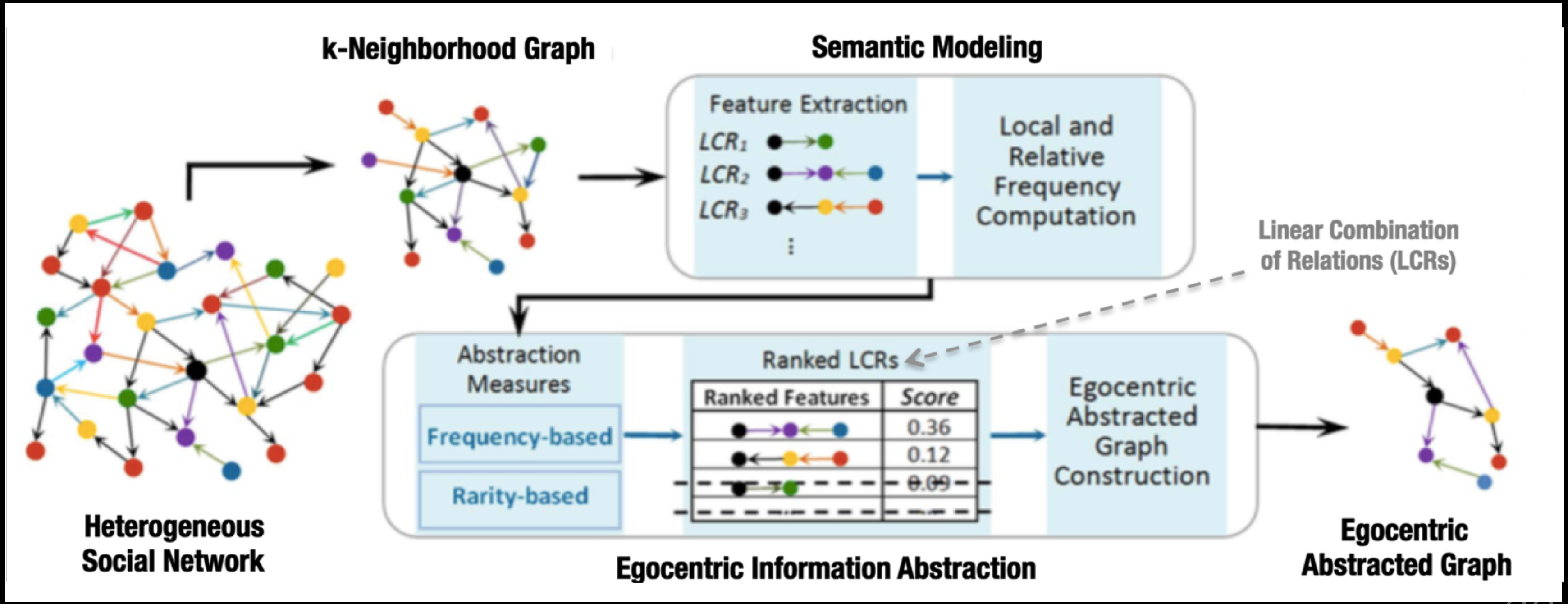


[Buehrer and Chellapilla, '08]

Egocentric Abstraction

Main Ideas:

- unsupervised approach that creates an abstract representation of an ego-network
- edge filtering: based on frequent or rare patterns

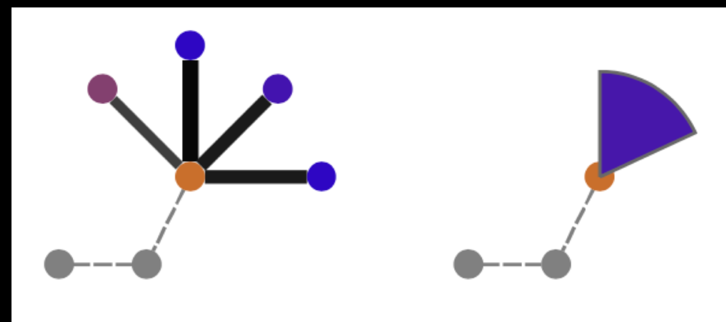


[Li et al., '09]

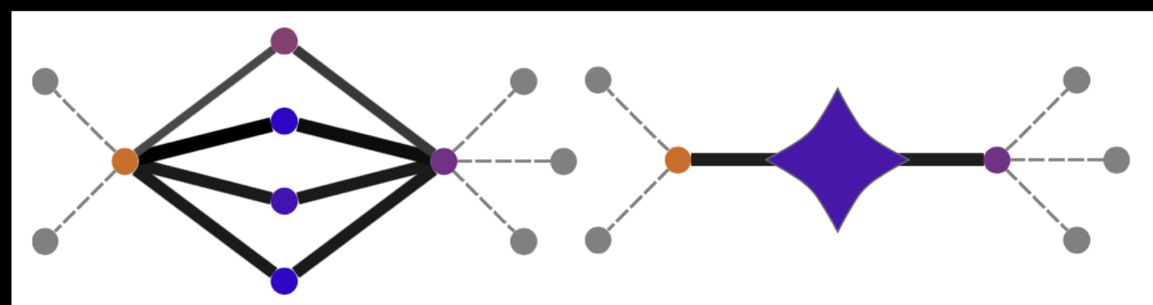
Motif Simplification

Tailored detection algorithms for three motifs:

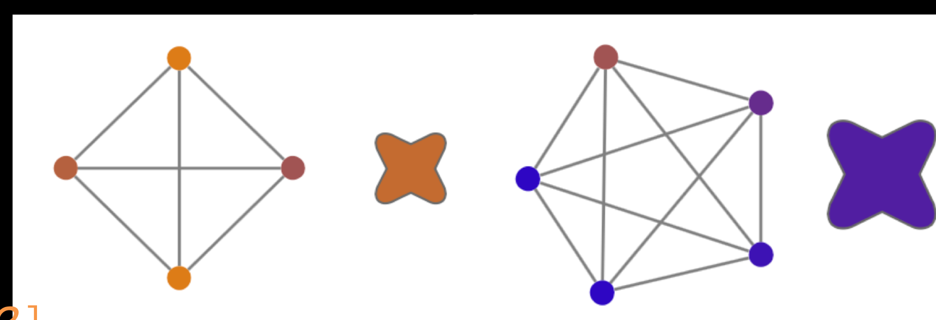
- Fans



- Connectors



- Cliques

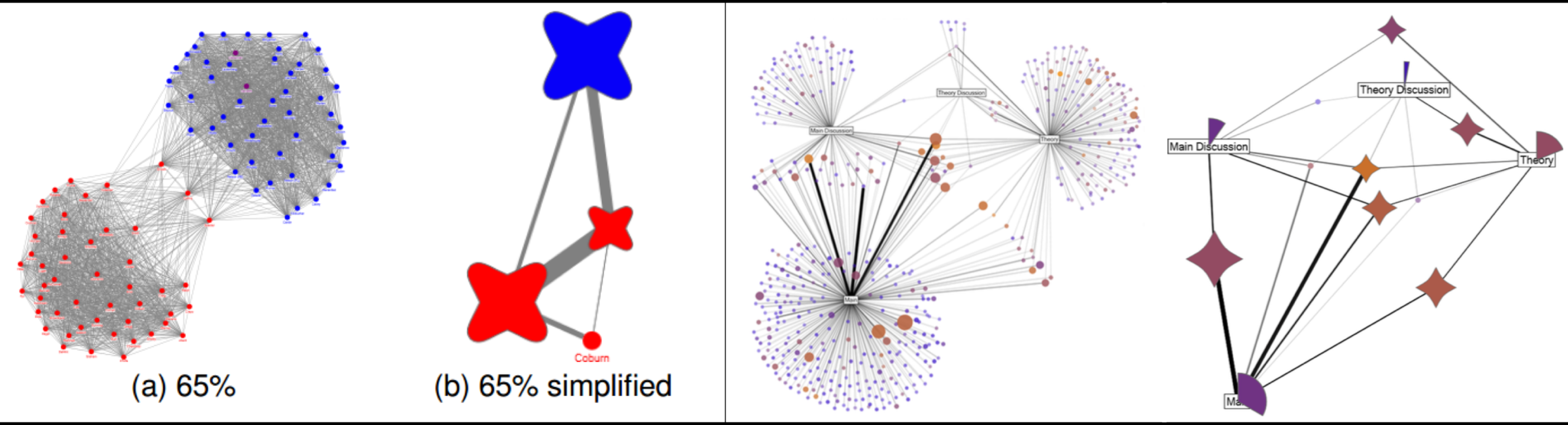


[Dunne and Shneiderman, '*13*]

Motif Simplification

Target: Visualization

- less screen space and layout effort
- better understanding



US Senate 2007
co-voting network

Lostpedia wiki edits
(bipartite network)

[Dunne and Shneiderman, ‘13.]

Main Underlying Idea: Minimum Description Length

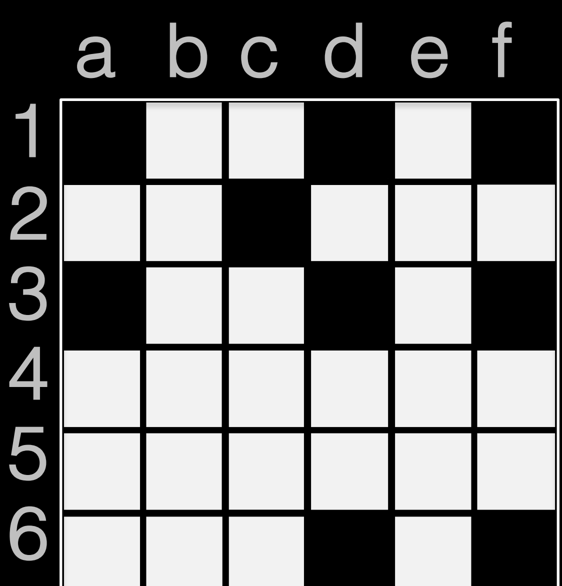
$$\min \; L(M) + L(D|M)$$

~Occam's razor

simple and **good** explanations

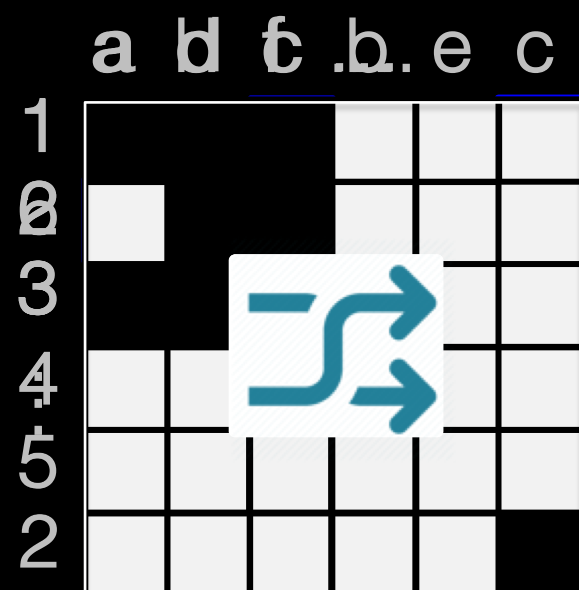
the
ng M

Option 1



[M]
Ø

[D|M](1,
a)
(1,d)
(1,f)
(2,c)
(3,a)
(3,d)
(3,f)
(6,d)
(6,f)



Option 2

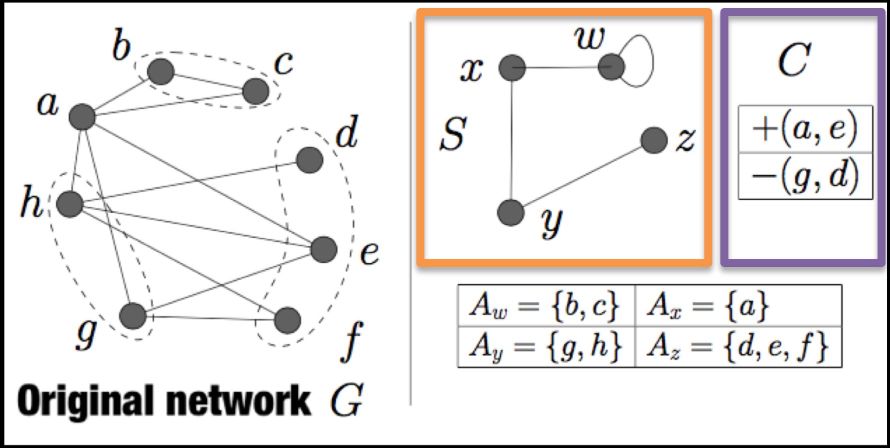
[M]
(1,6,3:
a,d,f)

[D|M]
-(6,a)
+(2,c)

Using Grouping and Compression

Two-part representation

- Aggregated graph S:
  - S_{Node}: collection of original nodes
  - S_{Edge}: edges between all node pairs in S_{Node}
- Edge corrections C:
  - to recreate the original nodes

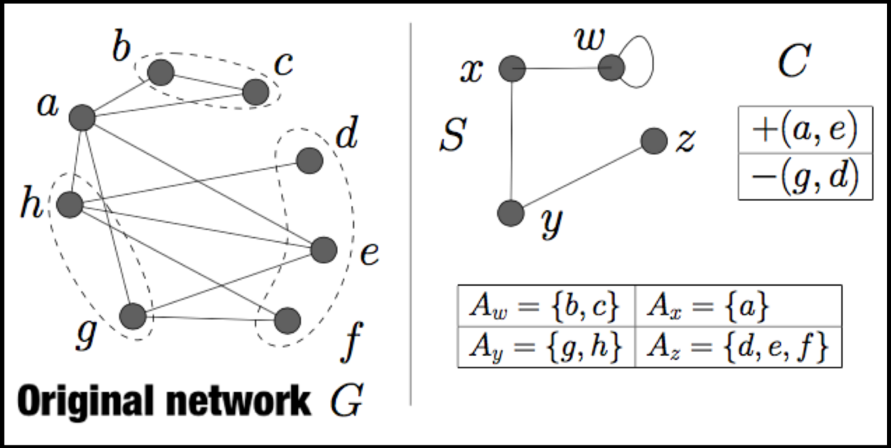


[Navlakha et al., '09]

Using Grouping and Compression

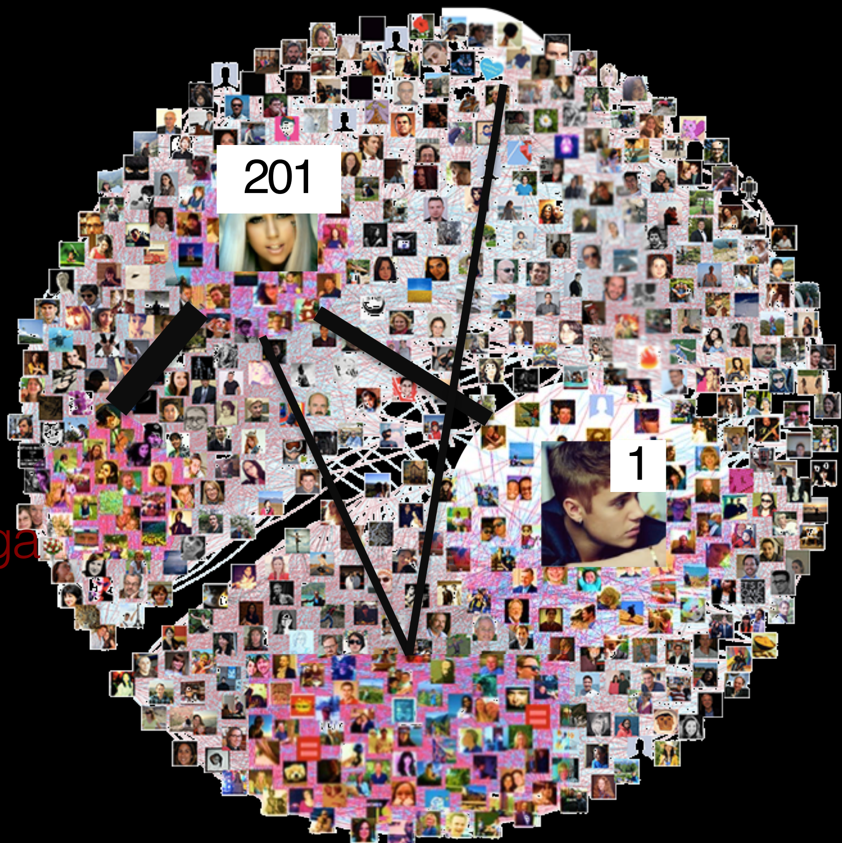
Algorithmic Ideas:

- merge node groups when the MDL cost decreases
- Greedy: iteratively merge nodes with highest MDL cost reduction
  - ✧ only considers pairs of nodes within 2-hops from each other
- Randomized:
  randomly picks nodes and merges it with its best neighbor in 2-hop neighborhood



[Navlakha et al., '09]

VoG: Vocabulary-based Summarization



Lady Gaga
Fan Club

Given: a large unlabeled graph

Find: a **succinct** summary efficiently

Find important graph structures

[Koutra et al., *SIAM SDM* '14.]

[Koutra et al., *Stat. Anal. Data Min. J.* '14.]

Challenges

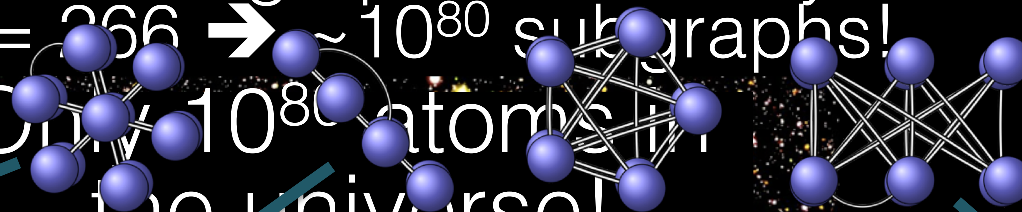
Challenge 1: What subgraphs to consider?

For n nodes ➔ 2^n possible subgraphs

Idea 1: Use a graph vocabulary.

e.g., n = 266 ➔ ~10^{80} subgraphs!

Only 10^{80} atoms in the universe!



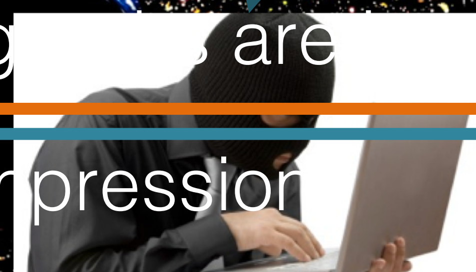
Challenge 2: Which subg... are ...

...mpression

popularity

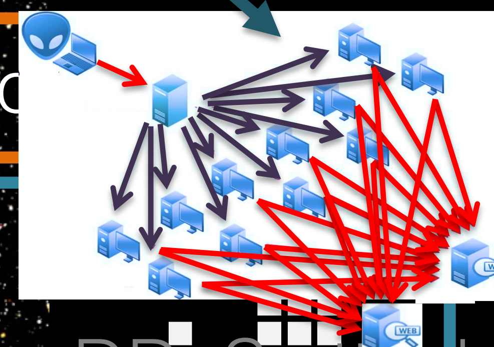
best graph summary

influence == propagation



shortest lossless description

sockpuppets



DDoS attack

[**Koutra** et al., *SIAM SDM* '14.]

[**Koutra** et al., *Stat. Anal. Data Min. J.* '14.]

Minimum Graph Description

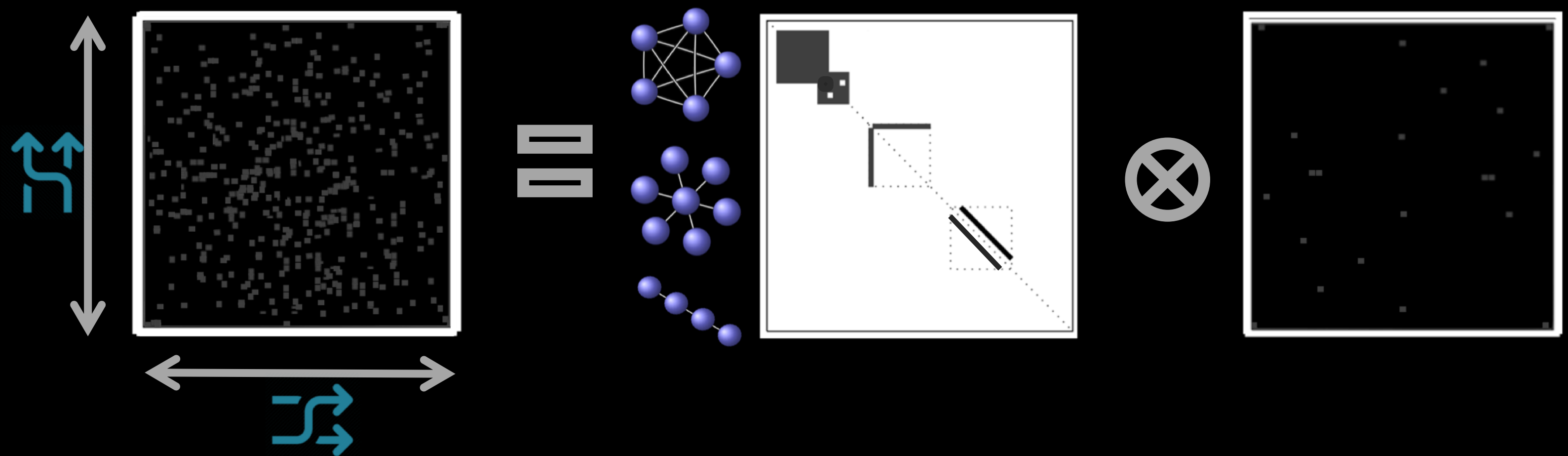
plain

Given: a graph G with adjacency matrix A

Find: model M s.t.

$$\min L(G,M) = \min \{ L(M) + L(E) \}$$

Adjacency A   Model M   Error E



[Koutra et al., *SIAM SDM* '14.]   [Koutra et al., *Stat. Anal. Data Min. J.* '14.]

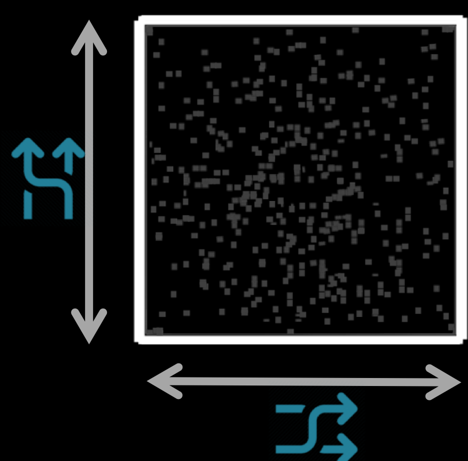
Minimum Graph Description

Given: a graph G with adjacency matrix A

Find: model M s.t.

$$\min L(G,M) = \min\{ L(M) + L(E) \}$$

Adjacency A

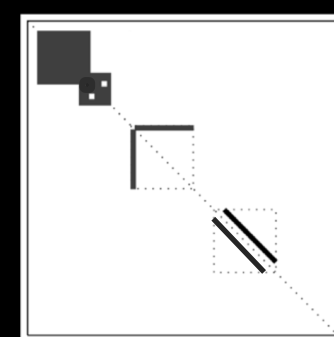


How?

1) Minimize description

2) Scalable

Model M

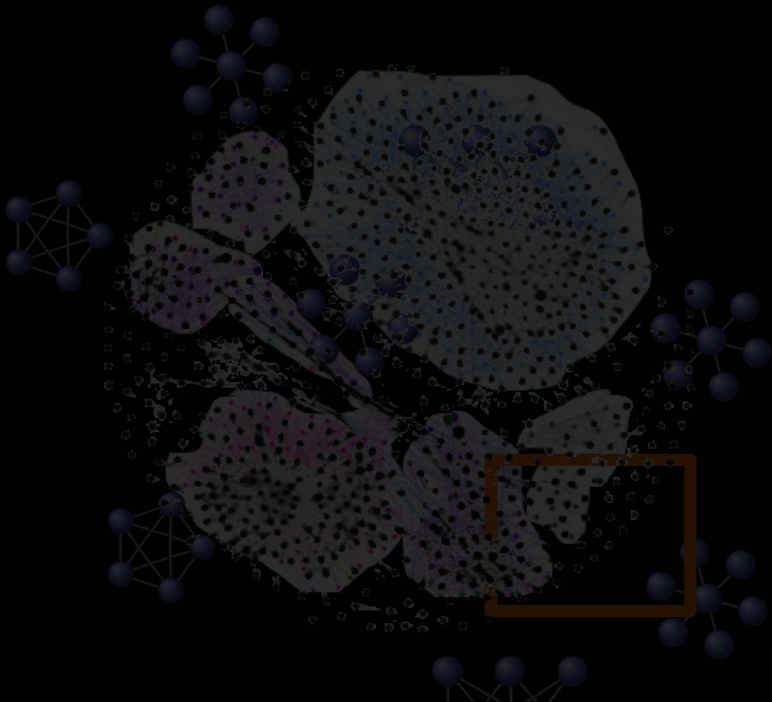


[Koutra et al., *SIAM SDM* '14.]

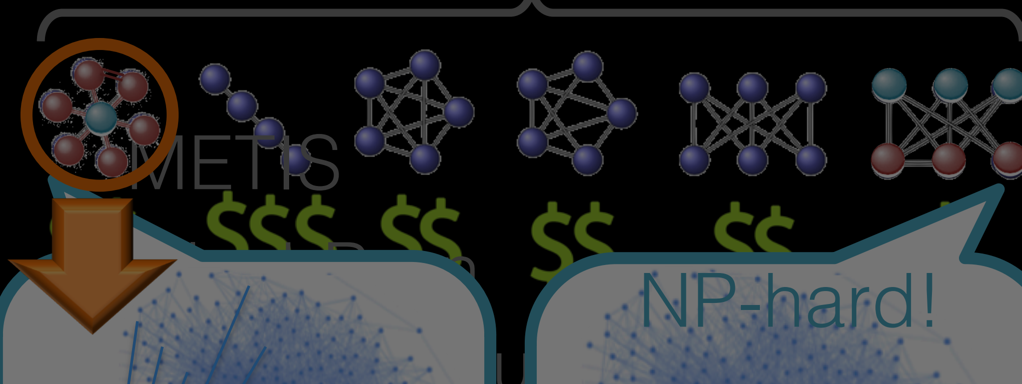
[Koutra et al., *Stat. Anal. Data Min. J.* '14.]

VoG: Subgraph Extraction

VoG: Vocabulary-based Graph Summarization

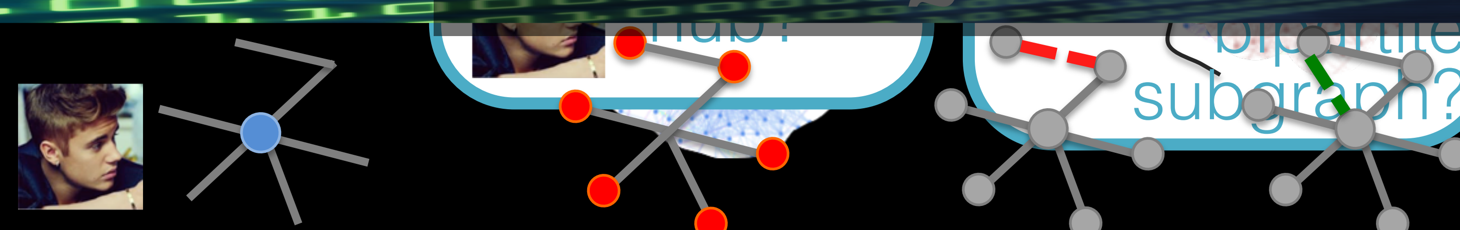


Could use:
ANY (overlapping) subgraph extraction method

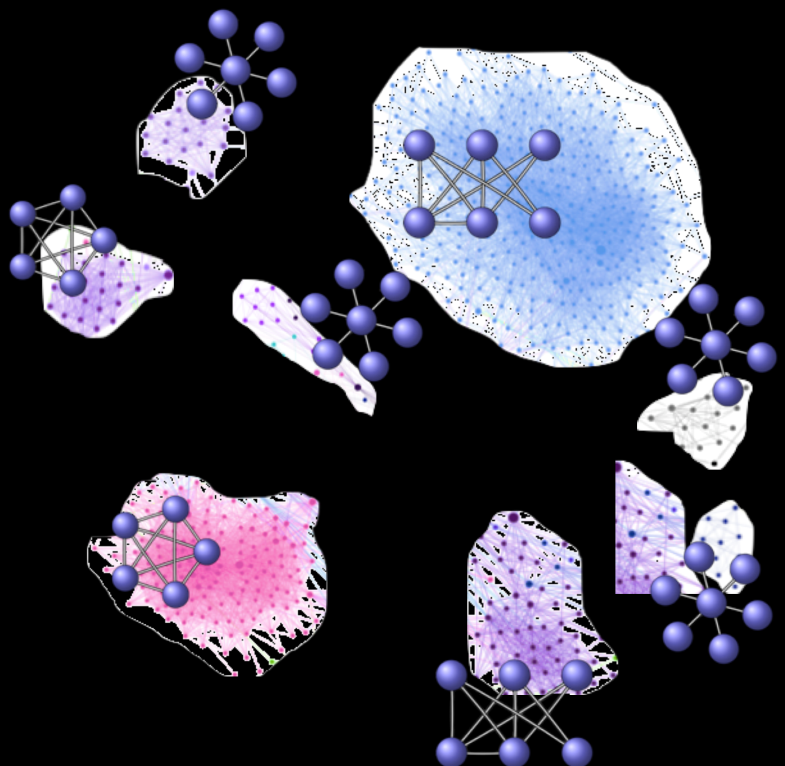


argmin

Star structure	+	Errors



VoG: Summary Assembly



Should we show all structures?

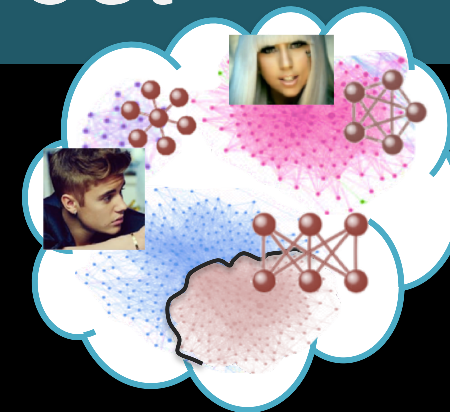
No, MDL will decide!

Choose the structures that:

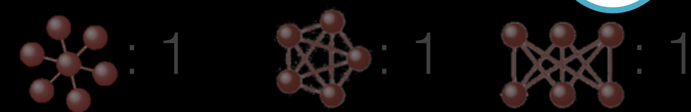
$$\text{minimize}\left\{\begin{array}{c}\text{encoding cost}\\\text{of the whole}\\\text{graph}\end{array}\right\}$$

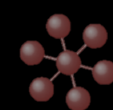
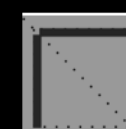
[Koutra et al., *SIAM SDM* '14.] [Koutra et al., *Stat. Anal. Data Min. J.* '14.]

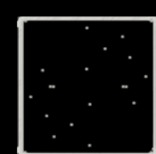
Summary Encoding Cost



$L(G, M) =$ # of structures + 3

# of structures per type + 

for each structure its encoding length + its  shape its  connectivity

errors 

min L(G,M) over $2^{\#\ \text{structures}}$ possible summaries

hard! # struct: 500-30,000

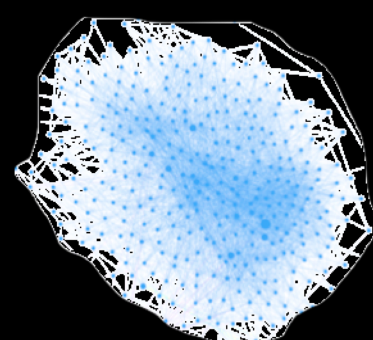
VoG Summary Assembly

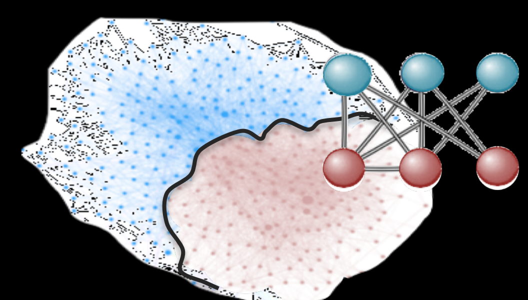
(i) Plain ✗ Too many

(ii) Top-k ✗ Overlaps?

Savings = # bits as noise - # bits as structure

compression gain

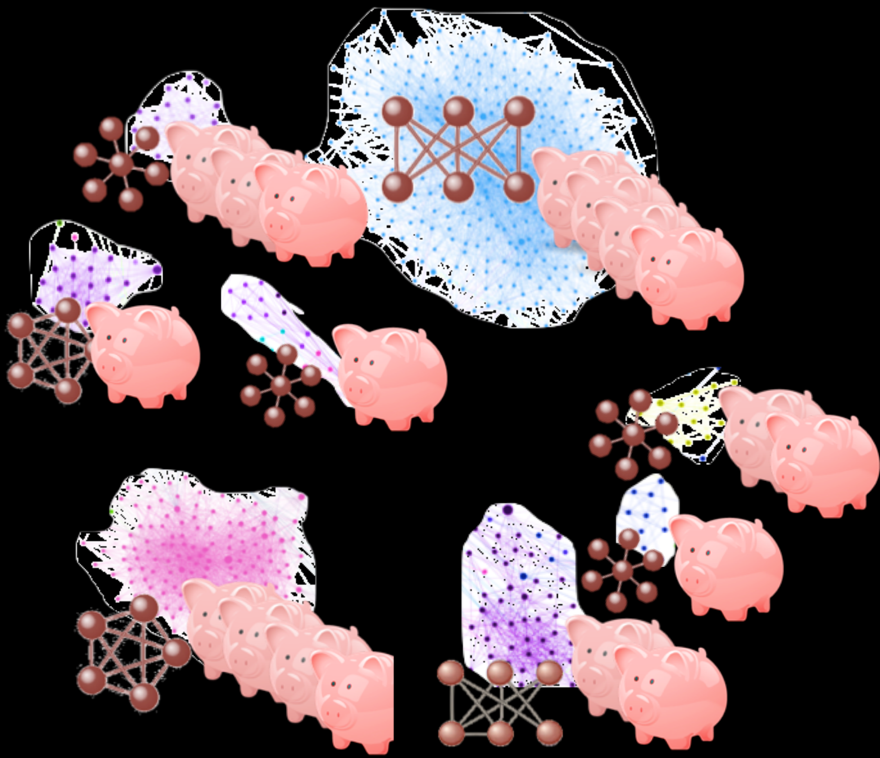


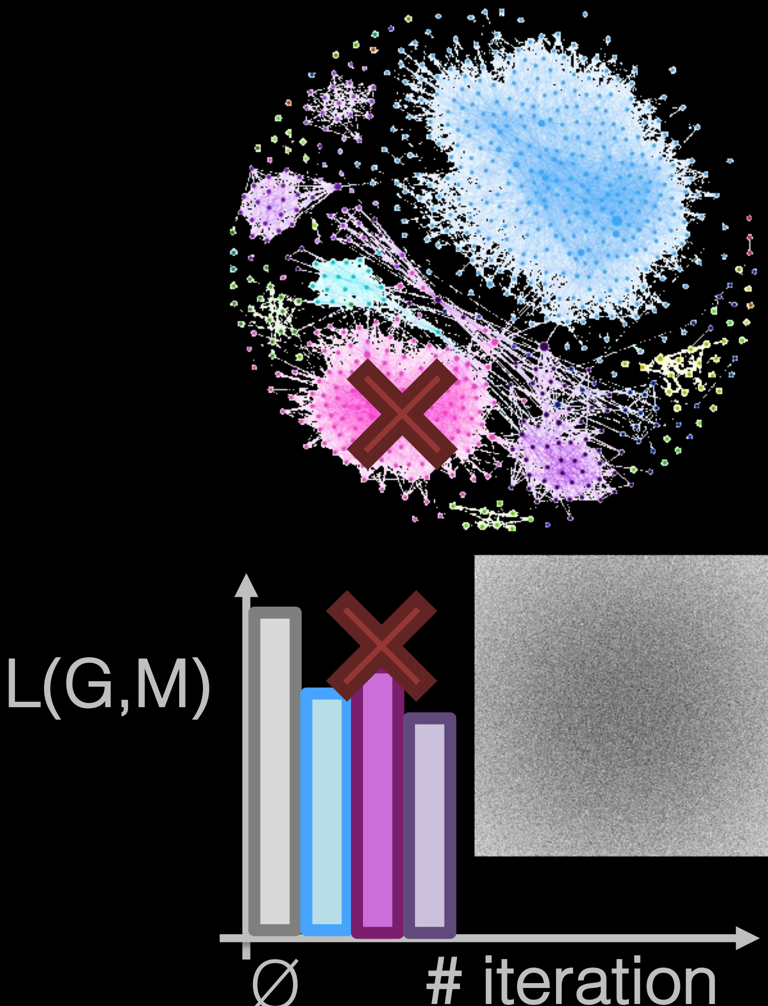


[Koutra et al., *SIAM SDM* '14.]

[Koutra et al., *Stat. Anal. Data Min. J.* '14.]

VoG Summary

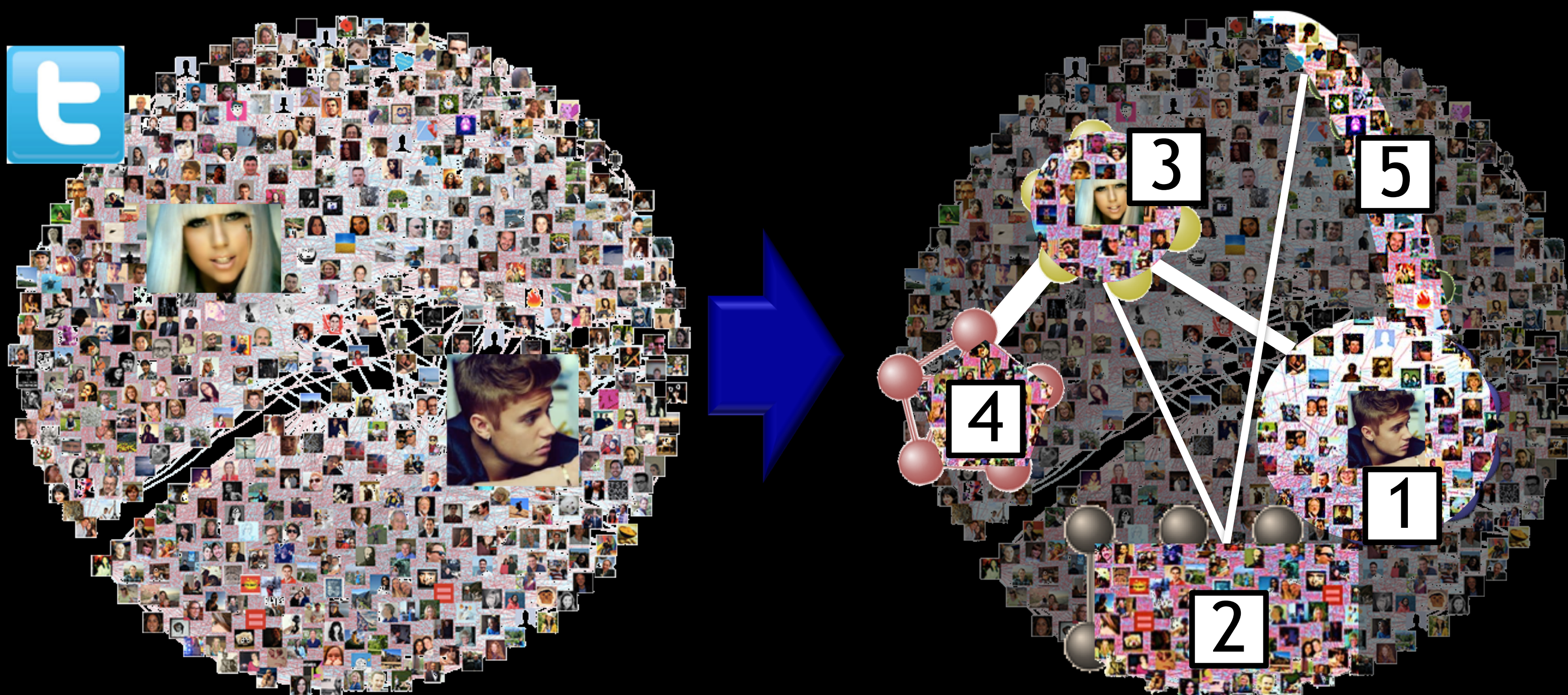




[Koutra et al., *SIAM SDM* '14.]

[Koutra et al., *Stat. Anal. Data Min. J.* '14.]

VoG Summary

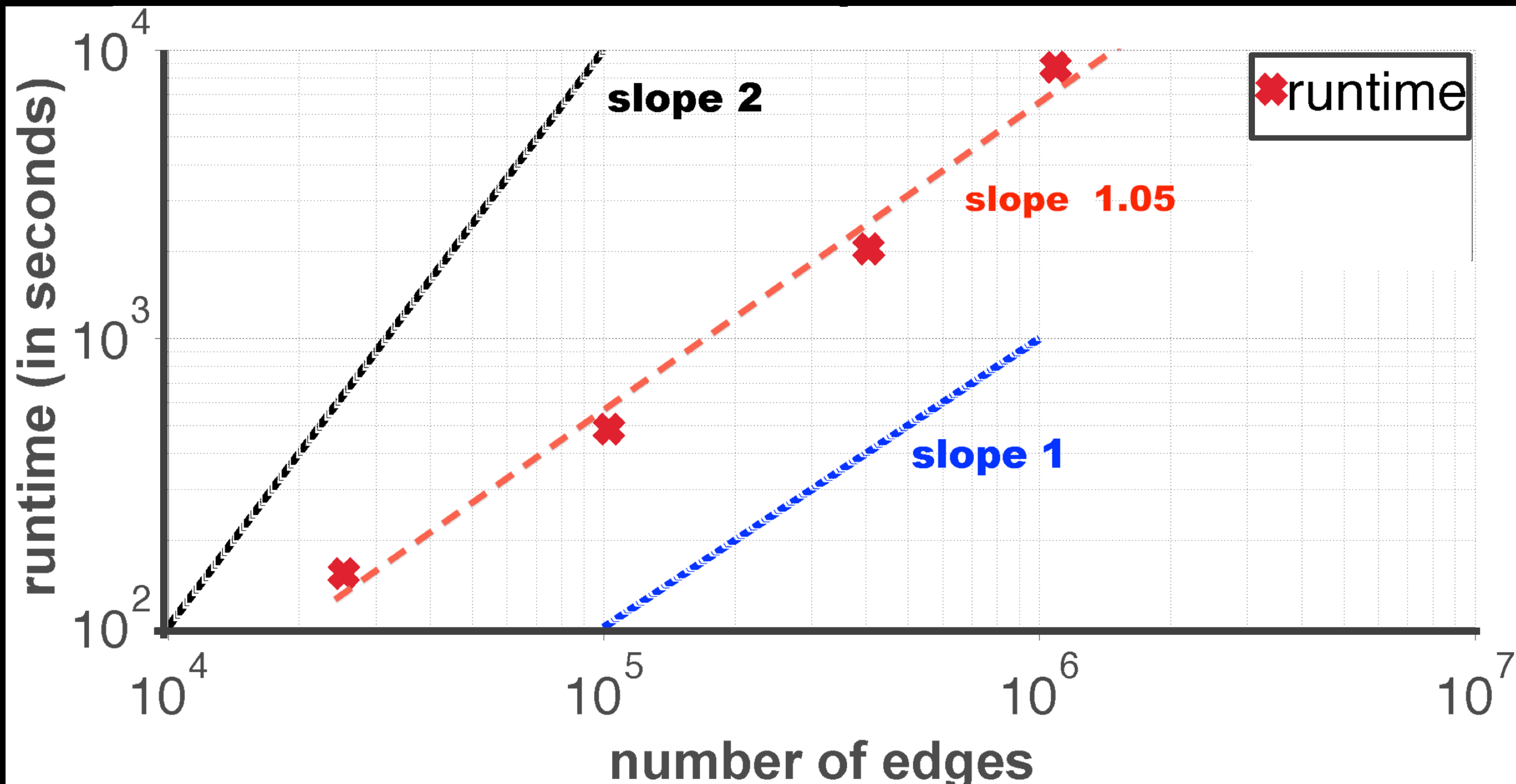


ranked on importance
"Attention routing"

[Koutra et al., *SIAM SDM* '14.]

[Koutra et al., *Stat. Anal. Data Min. J.* '14.]

VoG Runtime

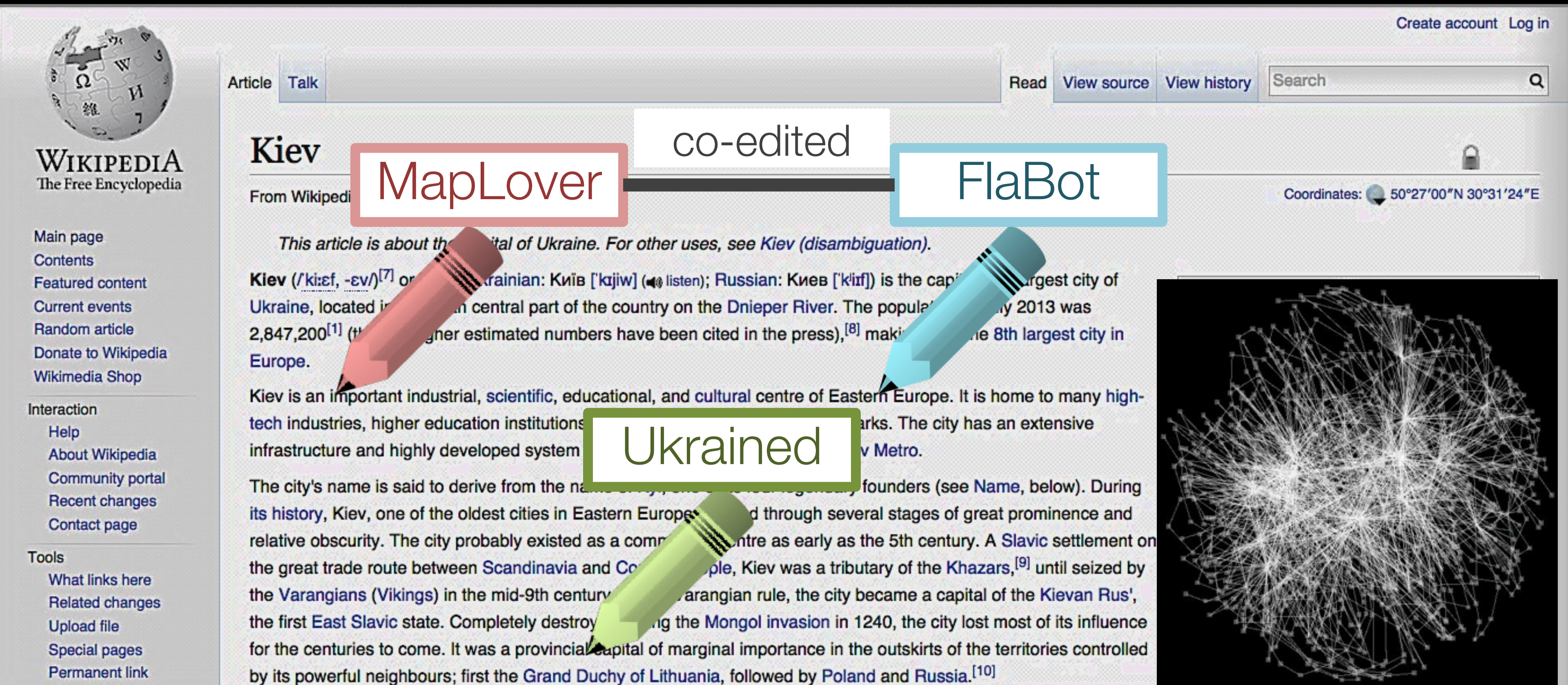


VoG is *near-linear* on # edges of the input graph.

[Koutra et al., *SIAM SDM* '14.]

[Koutra et al., *Stat. Anal. Data Min. J.* '14.]

VoG: Understanding Wiki



[Koutra et al., *SIAM SDM* '14.]

[Koutra et al., *Stat. Anal. Data Min. J.* '14.]

VoG: Understanding Wiki

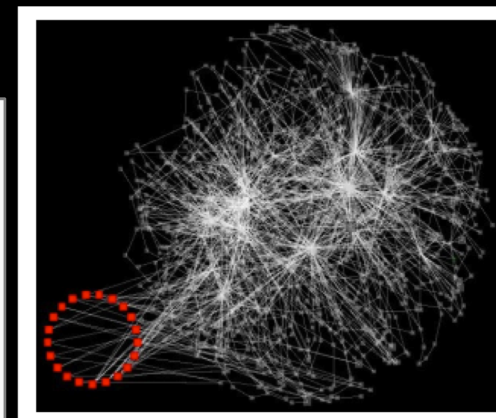

History

There leaders name was King Tong

Kiev is one of the oldest and most important cities of Eastern Europe and has played a pivotal role in the development of the medieval East Slavic civilization as well as in the modern Ukrainian nation.

Slavic settlement at the site of the present day city may have occurred as early as the sixth century AD (fifth century according to some researchers).[4] There are no known historical records as to the founding dates of the city. The Kiev article in Encyclopedia Britannica states: "The village that became the modern city may have been founded as early as the 6th century AD." The Columbia Encyclopedia in Kiev states: "It probably existed as a commercial centre as early as the 5th cent."</ref> With the exact time of city foundation being hard to determine, May 1982 was chosen to celebrate the city's 1,500th anniversary.

During the eighth and ninth centuries, Kiev was an outpost of the Khazar empire. Starting in the late ninth century or early tenth century Kiev was ruled by the Varangian nobility and became the nucleus of the Rus' polity, whose Golden Age (eleventh to early twelfth centuries) has from the nineteenth century become referred to as Kievan Rus'. In 968, the nomadic Pechenegs attacked and then besieged the city.[5] In 1203 Kiev was captured and burned by Prince Rurik Rostislavich and his Kipchak allies. In the 1230s the city was sieged and ravaged by different Russian princes several times. In 1240 the Mongol invasion of Rus led by Batu Khan completely destroyed Kiev,[6] an event that had a profound effect on the future of the city and the East Slavic civilization. At the time of the Mongol destruction, Kiev was reputed as one of the largest cities in the world, with a population exceeding one hundred thousand.



Bipartite core 2: edit war

vandals vs. admins

[Koutra et al., *SIAM SDM* '14.]

[Koutra et al., *Stat. Anal. Data Min. J.* '14.]

Summarization as an Evaluation Metric for Clustering

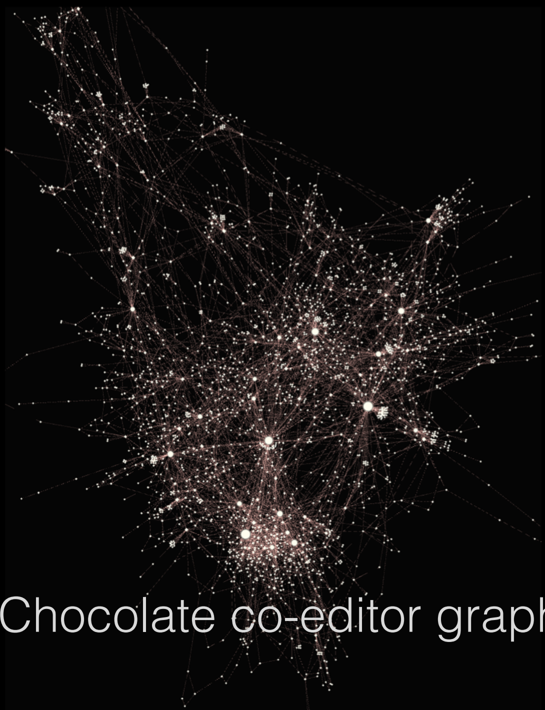
- Extension of VoG [Liu et al.'16] to handle:
  - overlapping edges (extra penalty) and
  - multiple clustering methods
  - clustering as a summarization tool

	SlashBurn [16]	Louvain [4]	Spectral [15]	METIS [17]	HyCoM [2]	BigClam [30]	KCBC [24]
Overlapping Clusters	✔	✘	✘	✘	✔	✔	✔
Cliques	Many	Many	Many	Many	Some	Many	Many
Stars	Many	Some	Some	Some	Many	Some	Some
Bipartite Cores	Some	Few	Many	Some	Some	Few	Few
Chains	Few	Few	Few	Few	Few	Few	Few
Hyperbolic Structures	Few	Few	Few	Few	Many	Few	Few
Complexity	$O(t(m + n\log n))$	$O(n\log n)$	$O(n^3)$	$O(m \cdot k)$	$O(k(m + h\log h^2 + hm_h))$	$O(d \cdot n \cdot t)$	$O(t(m+n))$
Summarization Power	Excellent	Very Good	Good	Good	Poor	Good	Poor

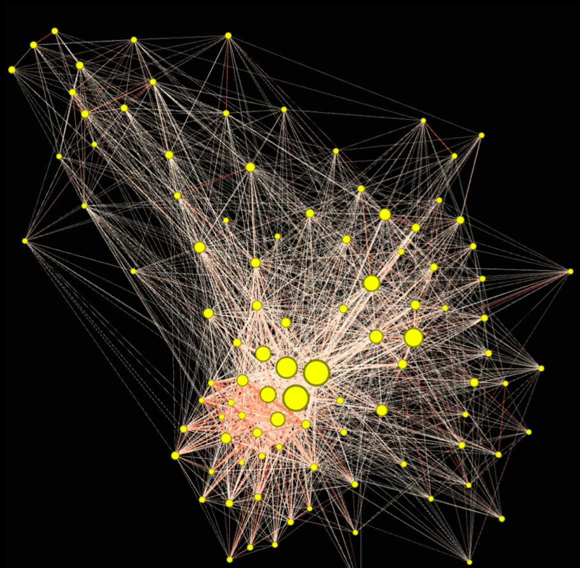
Dataset	Clustering Methods						
	SlashBurn	Louvain	Spectral	METIS	HyCoM	BigClam	KCBC
Choc	88%	99%	99%	100%	100%	87%	**78%**
AS-Oregon	76%	94%	82%	85%	98%	83%	**65%**
AS-Caida	**70%**	100%	100%	98%	98%	91%	74%

[Liu et al., '16]

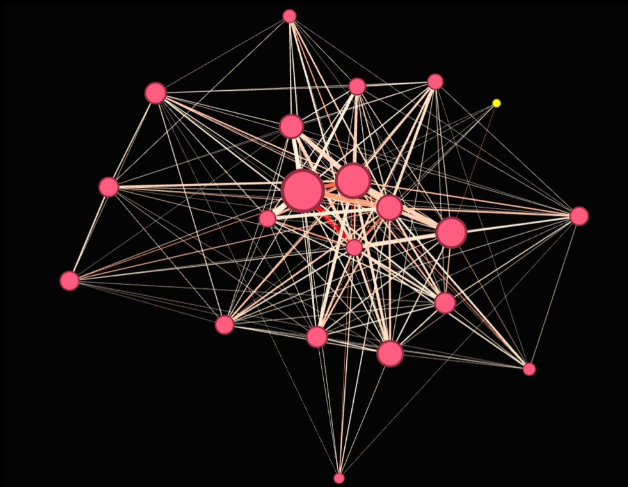
Summarization for Visualization



Chocolate co-editor graph



VoG [Koutra et al.'14]



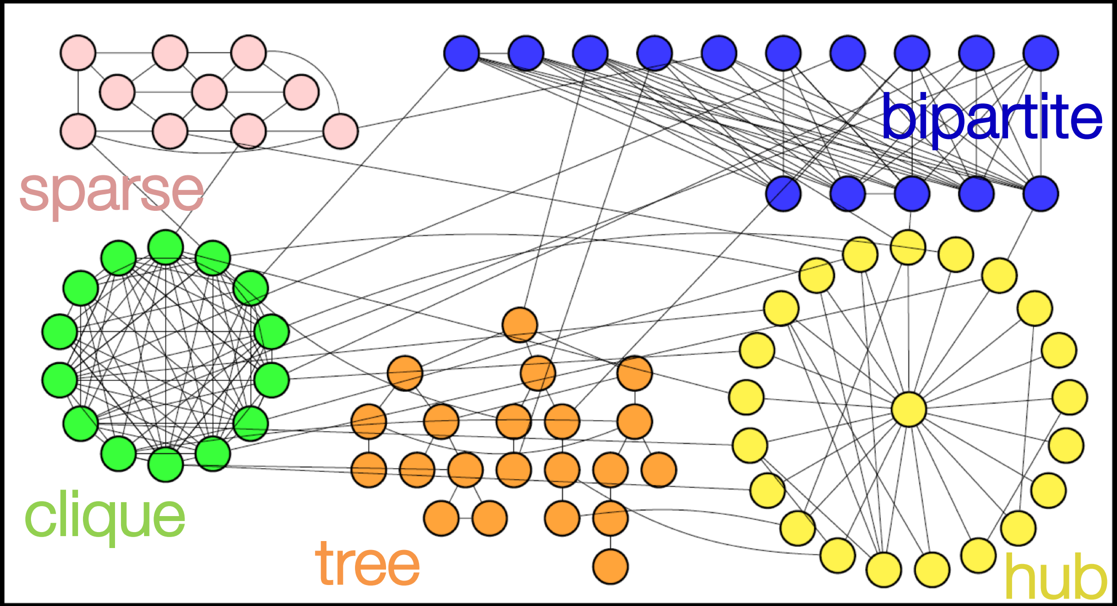
Condense [Liu et al.'16]

[Liu et al., '16]

MEGS

Similar in vein to VoG

- based on MDL, assumes node order
- summarizes the whole graph, instead of only parts with well-identified structure
- does not allow overlapping supernodes

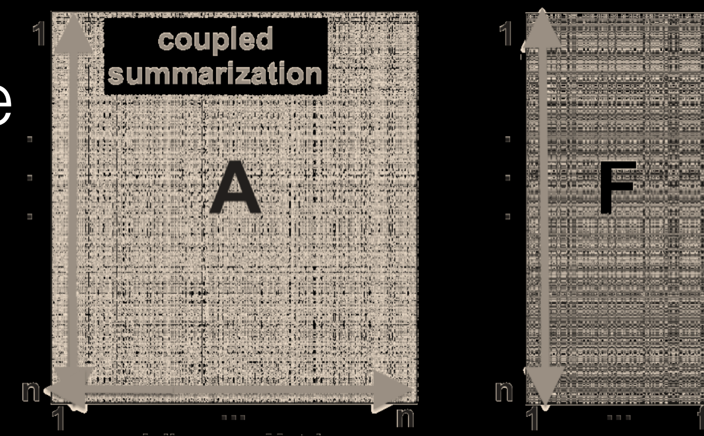


[Goebl et al., '16]

Attributed Graph Summarization

The vast majority of methods are based on grouping

- nodes that are structurally similar + share attributes



		Input Graph				Algorithmic Properties				
	Method	**Weighted**	**Undirect.**	**Directed**	**Heterog.**	**Prm-free**	**Linear**	**Technique**	**Output**	**Objective**
Static Labeled Graphs	S-Node [Raghavan and Garcia-Molina 2003]	✗	✗	✓	✗	✓	✗	grouping	supergraph	query efficiency
	SNAP/k-SNAP [Tian et al. 2008]	✗	✓	✓*	✗	✓	✓	grouping	supergraph	query efficiency
	CANAL [Zhang et al. 2010]	✗	✓	✓*	✗	✓	✗	grouping	supergraph	patterns
	Probabilistic [Hassanlou et al. 2013]	✓	✗	✓	✗	✓	✓	grouping	supergraph	compression
	Query-Pres. [Fan et al. 2012]	✗	✗	✓	✗	✗	✓	grouping	supergraph	query efficiency
	ZKP [Shoaran et al. 2013]	✗	✗	✓	✗	✓	✓	grouping	supergraph	privacy
	Randomized [Chen et al. 2009]	✗	✓	✗	✓	✓	✗	grouping	supergraph	patterns
	d-summaries [Song et al. 2016]	✗	✗	✓	✓	✗	✗	grouping	supergraph	query efficiency
	SUBDUE [Cook and Holder 1994]	✗	✓	✓	✓	✓	✗	compression	supergraph	patterns
	AGSUMMARY [Wu et al. 2014]	✗	✗	✓	✗	✓	✓	compression	supergraph	compression
	LSH-based [Khan et al. 2014]	✗	✗	✓	✗	✗	✓	compression	supergraph	compression
	VEGAS [Shi et al. 2015]	✓*	✗	✓	✗	✗	✓*	influence	supergraph	influence

For more details




- Based on survey

https://dl.acm.org/citation.cfm?id=3186727

Graph Summarization Methods and Applications: A Survey

YIKE LIU, TARA SAFAVI, ABHILASH DIGHE, and DANAI KOUTRA, University of Michigan, Ann Arbor

While advances in computing resources have made processing enormous amounts of data possible, human ability to identify patterns in such data has not scaled accordingly. Efficient computational methods for condensing and simplifying data are thus becoming vital for extracting actionable insights. In particular, while data summarization techniques have been studied extensively, only recently has summarizing interconnected data, or *graphs*, become popular. This survey is a structured, comprehensive overview of the state-of-the-art methods for summarizing graph data. We first broach the motivation behind and the challenges of graph summarization. We then categorize summarization approaches by the type of graphs taken as input and further organize each category by core methodology. Finally, we discuss applications of summarization on real-world graphs and conclude by describing some open problems in the field.

62

CCS Concepts: • **Mathematics of computing** → **Graph algorithms**; • **Information systems** → **Data mining**; **Summarization**; • **Human-centered computing** → *Social network analysis*; • **Theory of computation** → *Unsupervised learning and clustering*; • **Computing methodologies** → *Network science*;

Additional Key Words and Phrases: Graph mining, graph summarization

ACM Reference format:
Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph Summarization Methods and Applications: A Survey. *ACM Comput. Surv.* 51, 3, Article 62 (June 2018), 34 pages.
https://doi.org/10.1145/3186727

1 INTRODUCTION

References

Graph Summarization with Bounded Error. Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. In SIGMOD, 2008.

CSI: Community-level Social Influence Analysis. Mehmood, Y.; Barbieri, N.; Bonchi, F.; and Ukkonen, A. In Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), pages 48–63. Springer, 2013.

GraSS: Graph Structure Summarization. LeFevre, K.; and Terzi, E. In Tenth SIAM International Conference on Data Mining (SDM), pages 454-465, 2010. SIAM

Graph Summarization with Quality Guarantees. Riondato, M.; García-Soriano, D.; and Bonchi, F. In Proceedings of the 14th IEEE International Conference on Data Mining (ICDM), 2014.

Summarizing and Understanding Large Graphs. Koutra, D.; Kang, U; Vreeken, J.; and Faloutsos, C. In Statistical Analysis and Data Mining, 2015. John Wiley & Sons, Inc.

Reducing large graphs to small supergraphs: a unified approach. Liu, Y.; Safavi, T.; Shah, N.; and Koutra, D. Social Netw. Analys. Mining, 8(1): 17. 2018.

References

PERSEUS-HUB: Interactive and Collective Exploration of Large-scale Graphs. Jin, D.; Leventidis, A.; Shen, H.; Zhang, R.; Wu, J.; and Koutra, D. Informatics (Special Issue ``Scalable Interactive Visualization''), 4(3). 2017.

VEGAS: Visual influEnce GrAph Summarization on Citation Networks. Shi, L.; Tong, H.; Tang, J.; and Lin, C. IEEE Transactions on Knowledge and Data Engineering, 27(12): 3417–3431. 2015.

Substructure Discovery Using Minimum Description Length and Background Knowledge. Cook, D. J.; and Holder, L. B. Journal of Artificial Intelligence Research, 1: 231-255. 1994.

Improving Network Visualization Readability with Fan, Connector, and Clique Glyphs. Cody Dunne and Ben Shneiderman. Motif Simplification. In Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI), 2013.

Sparsification of Influence Networks. Michael Mathioudakis, Francesco Bonchi, Carlos Castillo, Aristides Gionis, and Antti Ukkonen. In KDD. 2011.

References

A Scalable Pattern Mining Approach to Web Graph Compression With Communities. Gregory Buehrer and Kumar Chellapilla. International Conference on Web Search and Data Mining, ACM, 2008.

Set-based approximate approach for lossless graph summarization. Khan, K.; Nawaz, W.; and Lee, Y. Computing, 97(12): 1185–1207. 2015.

Egocentric Information Abstraction for heterogeneous social networks. Cheng-Te Li and Shou-De Lin. 2009. International Conference on Advances in Social Network Analysis and Mining (ASONAM). IEEE, 255–260.

Graph Summarization for Attributed Graphs. Wu, Y.; Zhong, Z.; Xiong, W.; and Jing, N. In 2014 International Conference on Information Science, Electronics and Electrical Engineering (ISEEE), pages 503–507, 2014.

Mining Summaries for Knowledge Graph Search. Qi Song, Yinhui Wu, and Xin Luna Dong. In IEEE 16th International Conference on Data Mining (ICDM), 2016.

References

Discovery-driven Graph Summarization. Ning Zhang, Yuanyuan Tian, and Jignesh M. Patel. In ICDE, 2010.

Representing Web Graphs. Sriram Raghavan and Hector Garcia-Molina. In IEEE ICDE, 2003.

Efficient Aggregation for Graph Summarization. Yuanyuan Tian, Richard A Hankins, and Jignesh M Patel. In ACM SIGMOD, 2008.

Compression of Weighted Graphs. Hannu Toivonen, Fang Zhou, Aleksi Hartikainen, and Atte Hinkka. In KDD. 2011.