OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | INF | FAKULTÄT FÜR INFORMATIK

Graph Sparsification & Embeddings:

A study of their applications in graph databases

MSc Thesis Defense

Sindhuja Madabushi

Advisors:
Prof. Gunter Saake
Gabriel Campero Durand

Agenda

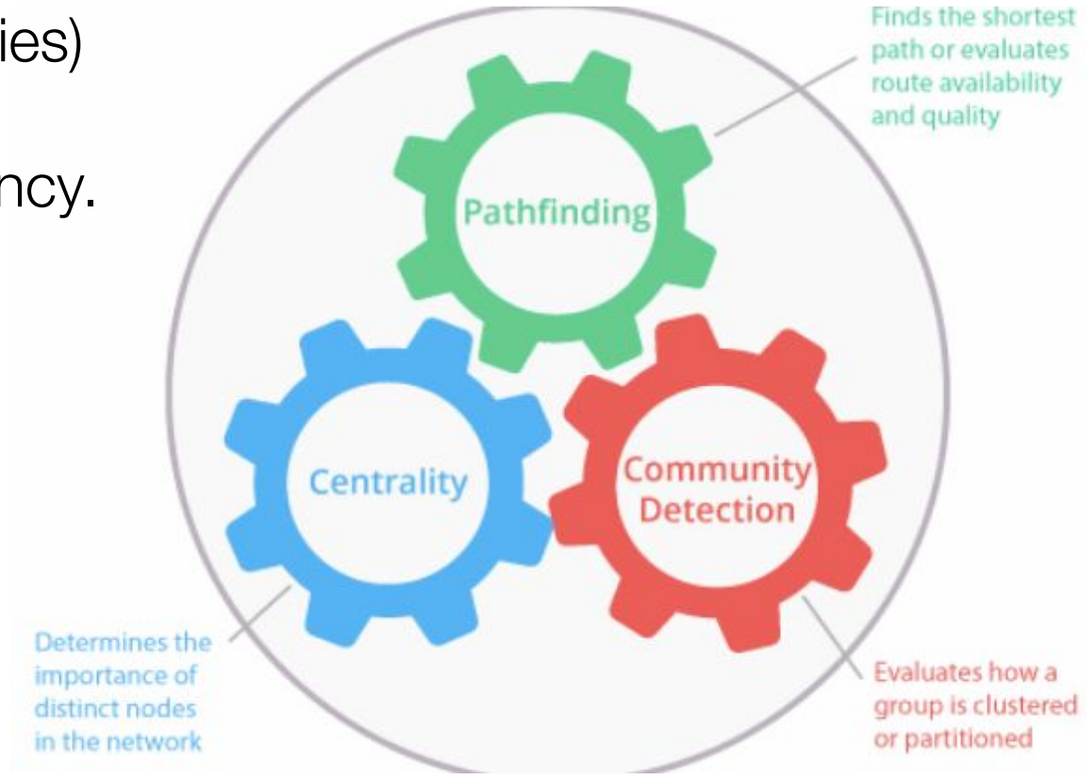
1. Motivation
2. Research Questions
3. Background
4. Prototypical implementation
5. Results & Discussion
6. Conclusion & Future work

1. Motivation

Motivation

Networks are everywhere

- In our digital world, **very large interconnected data** is generated rapidly every second
  - **Network analysis** (for pathfinding, and other tasks) is important
  - **Efficient tools** are required to store and analyze this data
    - Specialized graph systems (databases, libraries) help by providing join-free, index-free adjacency.
  - **Declarative query languages** (like Cypher) nowadays provide an SQL-like interface for optimizations to be done without involving users



https://neo4j.com/docs/graph-algorithms/3.5/

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | INF | FAKULTÄT FÜR INFORMATIK

Large scale graph analytics

Scalability and visualization are the most imperative challenges faced by researchers and practitioners while dealing with large graph data[1].

Challenge	Total	R	P
Scalability (i.e., software that can process larger graphs)	45	20	25
Visualization	39	17	22
Query Languages / Programming APIs	39	18	21
Faster graph or machine learning algorithms	35	19	16
Usability (i.e., easier to deploy, configure, and use)	25	10	15
Benchmarks	22	12	10
Extract & Transform	20	6	14
More general purpose graph software (e.g., that can process offline, online, and streaming computations)	20	9	11
Graph Cleaning	17	7	10
Debugging & Testing	10	2	8

[1]Sahu, Siddhartha, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M. Tamer Özsu. "The ubiquity of large graphs and surprising challenges of graph processing." Proceedings of the VLDB Endowment 11, no. 4 (2017): 420-431.

Approximations might improve scalability and visualization...

SELECT avg(sessionTime)
FROM Table
WHERE city='San Francisco'
WITHIN 2 SECONDS

Queries with Time Bounds

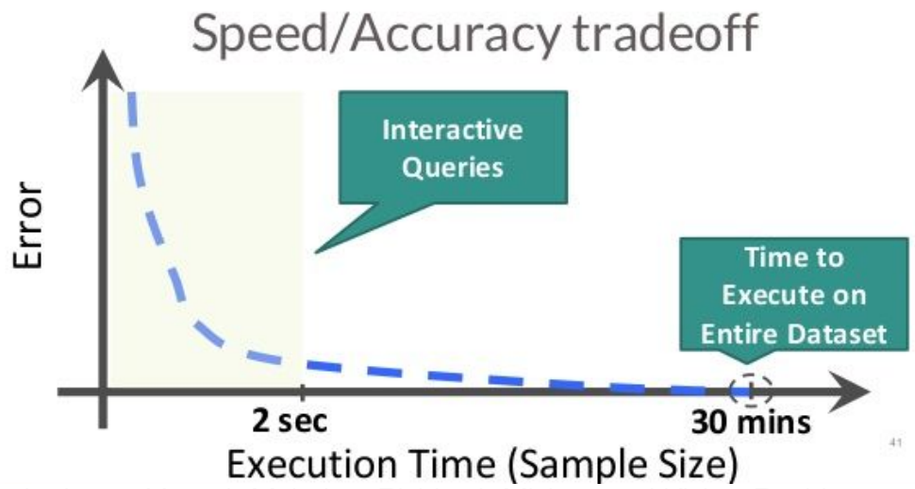
SELECT avg(sessionTime)
FROM Table
WHERE city='San Francisco'
ERROR 0.1 **CONFIDENCE** 95.0%

Queries with Error Bounds

For other kinds of data, **Approximate Query Processing** is studied to improve response time with large data.

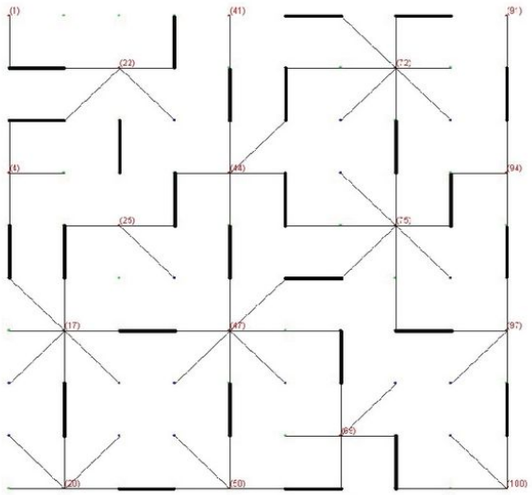
- Samples
- Summaries
- Special structures: Bloom filters

Let users decide on speed/accuracy tradeoff[1]



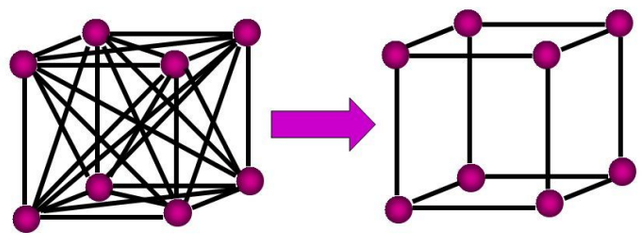
1Ramnarayan, Jags, Barzan Mozafari, Sumedh Wale, Sudhir Menon, Neeraj Kumar, Hemant Bhanawat, Soubhik Chakraborty, Yogesh Mahajan, Rishitesh Mishra, and Kishor Bachhav. "Snappydata: A hybrid transactional analytical store built on spark." In Proceedings of the 2016 International Conference on Management of Data, pp. 2153-2156. ACM, 2016.

But, how can graph approximation techniques be used?



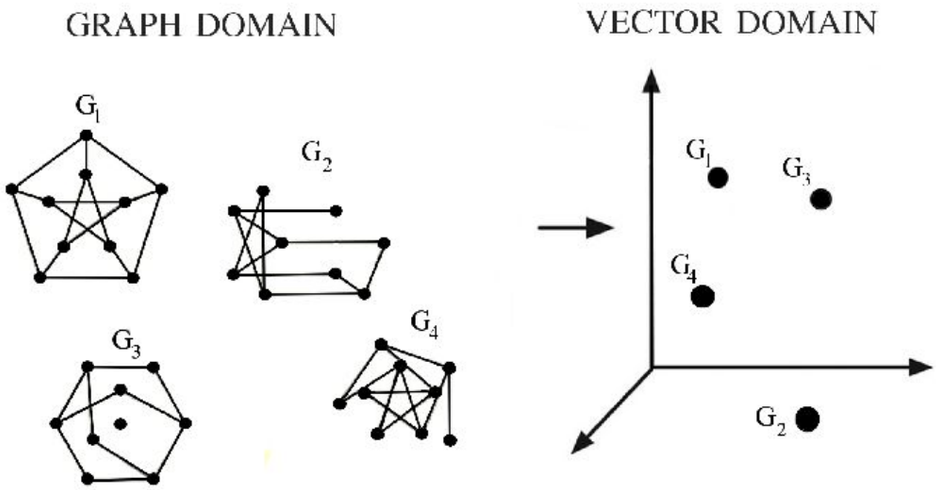
Graph Spanners

Preserving paths



Graph Sparsification

Preserving structural features



Graph Embedding

Domain change, with other views on data

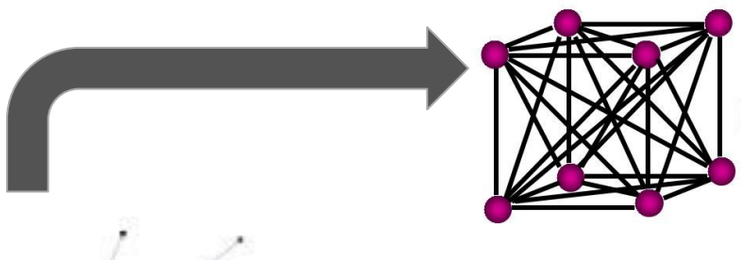
How much do these techniques improve graph processing?

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | INF | FAKULTÄT FÜR INFORMATIK

How much do these techniques improve graph processing?

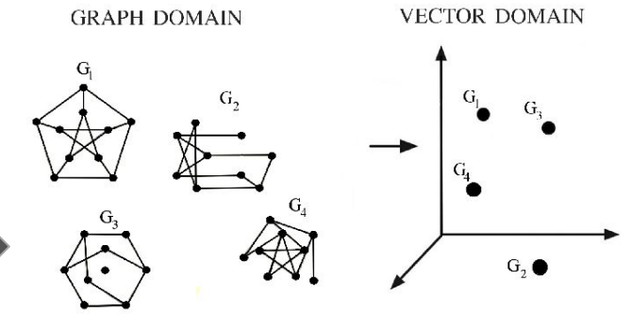
Performance/ Accuracy comparison, Scalability

Overhead of summarization process?



Original Graph

Graph Sparsification



Graph Embedding

Structural Queries

Query over raw data

Structural Node Similarity Queries

2. Research Questions

Sparsification

1. How does **time required to sparsify graph data** change with each sparsification technique?

2. How does the number of connected components and **community count** get affected if the graph data is sparsified using different techniques?

3. How does **query execution time** change when different queries are run on data sparsified using different techniques?

4. Are these techniques scalable, that is, do the above findings change for data with **higher scale factors**?

Embeddings

4. How does the **embedding time** change for different types of embeddings?

5. How does time taken for calculating **pair-wise cosine similarity** change for both embedded and non embedded data in the graph database, compared to calculating it manually?

3. Background

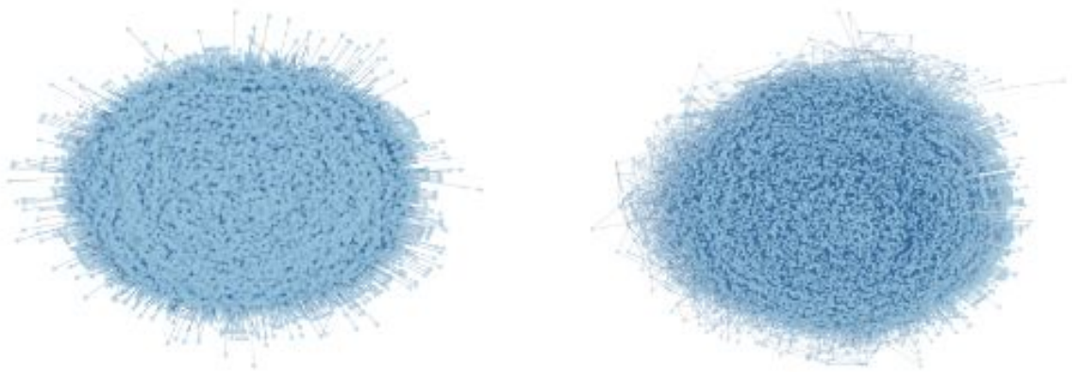
Summarization of large graph data

- Grouping/aggregation based summaries
- Simplification based summaries (spanners, sparsification)
- Bit-compression based summaries
- Domain specific summaries
- Latent representations (graph embeddings)

Liu, Yike, Tara Safavi, Abhilash Dighe, and Danai Koutra. "Graph Summarization Methods and Applications: A Survey." ACM Computing Surveys (CSUR) 51, no. 3 (2018): 62.

Summarization techniques for graphs

- **Benefits**:
  - Reduced memory footprint
  - Speed-up of graph algorithms and queries
  - Easy to visualize avoiding the “hairball” visualization problem for graphs
  - Noise elimination



Hairballs

Summarization techniques for graphs

- **Challenges**:
  - Complexity of data
  - Some summarization techniques are not scalable
  - Usefulness of summaries depends on workloads
  - Accuracy and error bounds are required for query processing over summaries
  - Updates on the origuinal data might require inefficient recomputation of summaries

Summarization of large graph data

- Grouping/aggregation based summaries
- Simplification based summaries (spanners, **sparsification**)
- Bit-compression based summaries
- Domain specific summaries
- Latent representations (**graph embeddings**)

We have selected these two families for our study due to the recent availability of open source standard implementations.

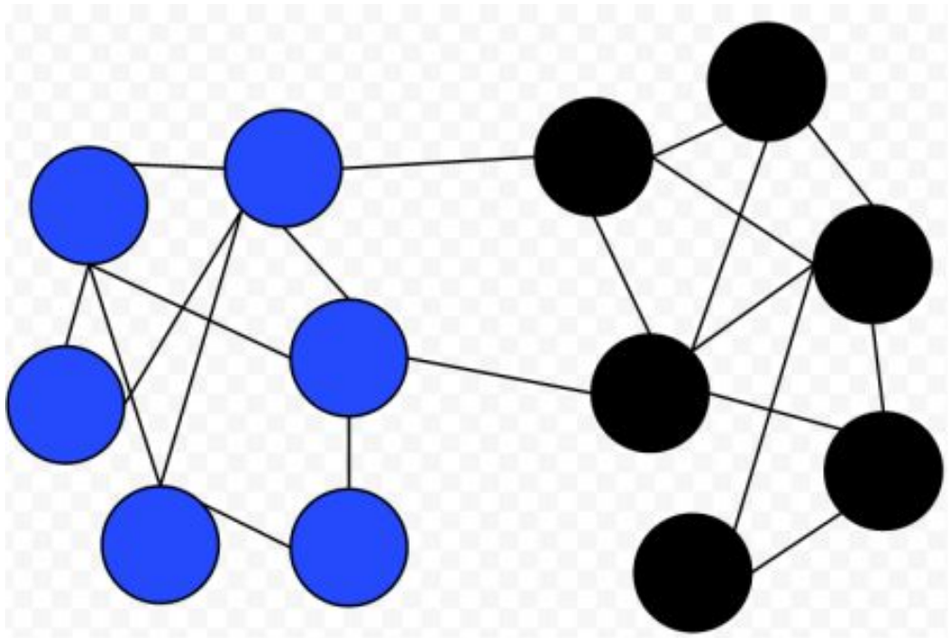
Liu, Yike, Tara Safavi, Abhilash Dighe, and Danai Koutra. "Graph Summarization Methods and Applications: A Survey." ACM Computing Surveys (CSUR) 51, no. 3 (2018): 62.

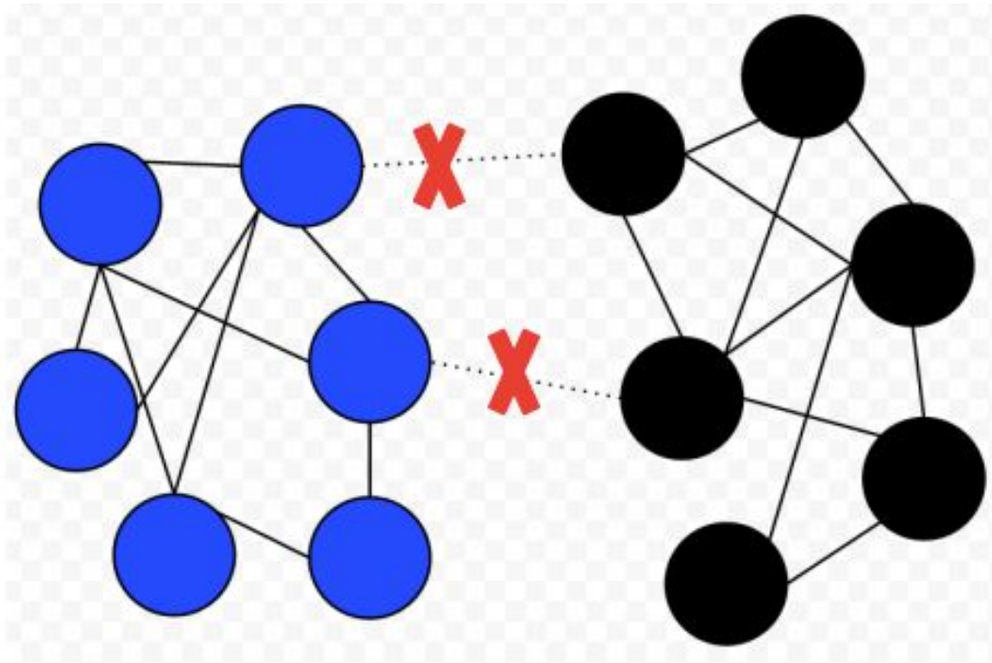
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | INF | FAKULTÄT FÜR INFORMATIK

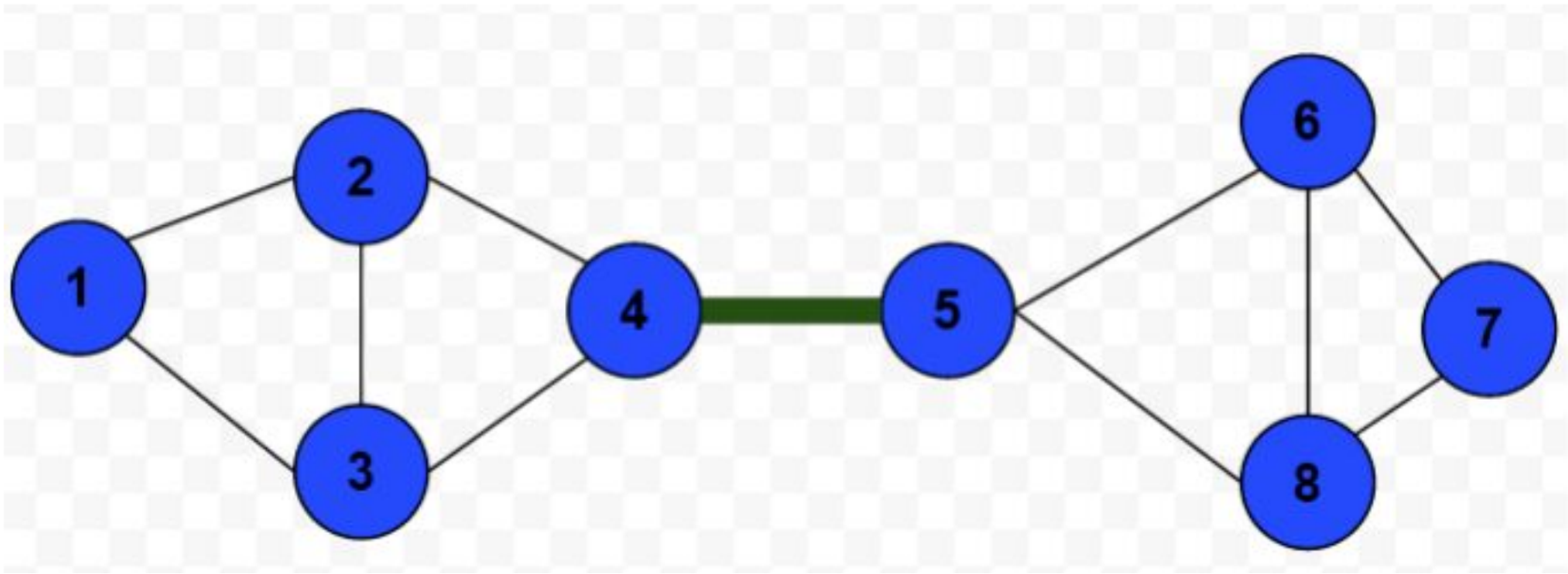
Simplification based summaries

- Aim at lessening the number of edges in a graph using a specific metric
- Size of network is reduced
- Structural and statistical properties are preserved
- Different types include: Sparsification, Spanners, Sketches, Graph sampling

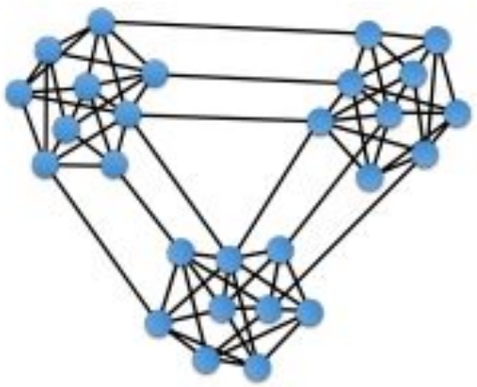
Sampling strategies are the core of sparsification



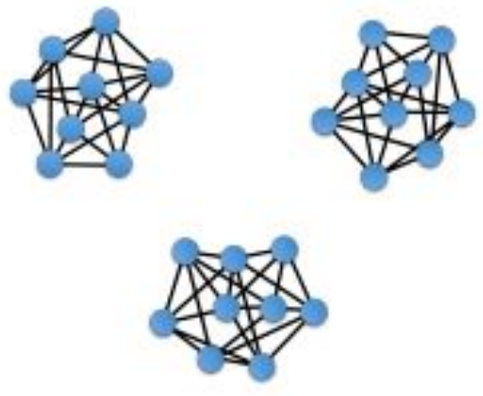




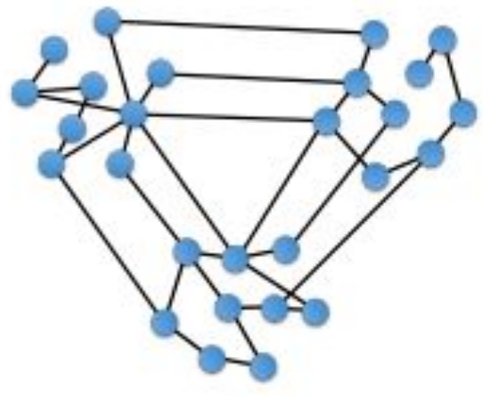
Satuluri, Venu, Srinivasan Parthasarathy, and Yiye Ruan. "Local graph sparsification for scalable clustering." In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pp. 721-732. ACM, 2011.



(a) Original network



(b) Sparsified network by eliminating δ-weak connections



(c) Sparsified network by eliminating δ-strong connections

Satuluri, Venu, Srinivasan Parthasarathy, and Yiye Ruan. "Local graph sparsification for scalable clustering." In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pp. 721-732. ACM, 2011.

Types of sparsification

- Random edge
- Algebraic distance
- Local degree
- Local similarity
- Triangle

Due to time constraints we only introduce 3.

Lindner, Gerd, Christian L. Staudt, Michael Hamann, Henning Meyerhenke, and Dorothea Wagner. "Structure-preserving sparsification of social networks." In Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on, pp. 448-454. IEEE, 2015.

Random edge sparsification

- Selects edges uniformly at random until a sparsification ratio is achieved

Local degree sparsification

- Based on the concept of “hub nodes”
- Edges to top $\lfloor d(u)^{\alpha} \rfloor$ nodes are kept for each node where d(u) is the degree of node u and $\alpha \in [0, 1]$ arranged in descending order

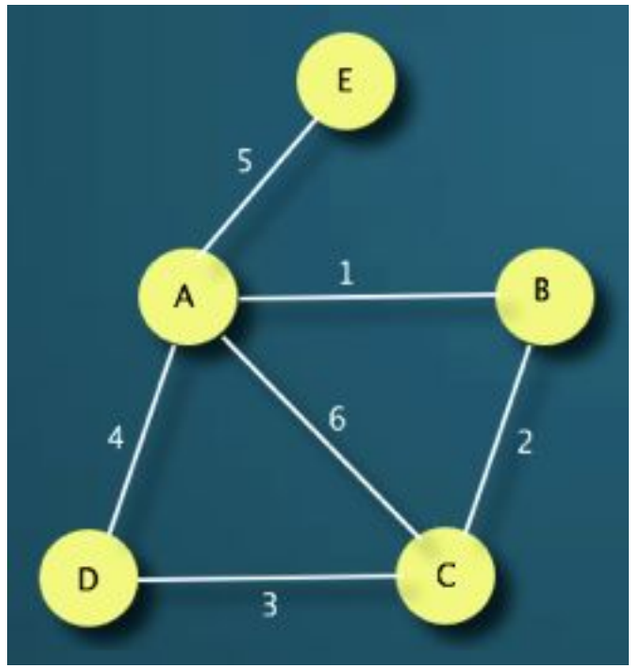
this prunes the local, keeps the global

Lindner, Gerd, Christian L. Staudt, Michael Hamann, Henning Meyerhenke, and Dorothea Wagner. "Structure-preserving sparsification of social networks." In Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on, pp. 448-454. IEEE, 2015.

Example

- For node A: d(C) = 3, d(B) = 2, d(D) = 2, d(E)=1, for convenience, let α=1.

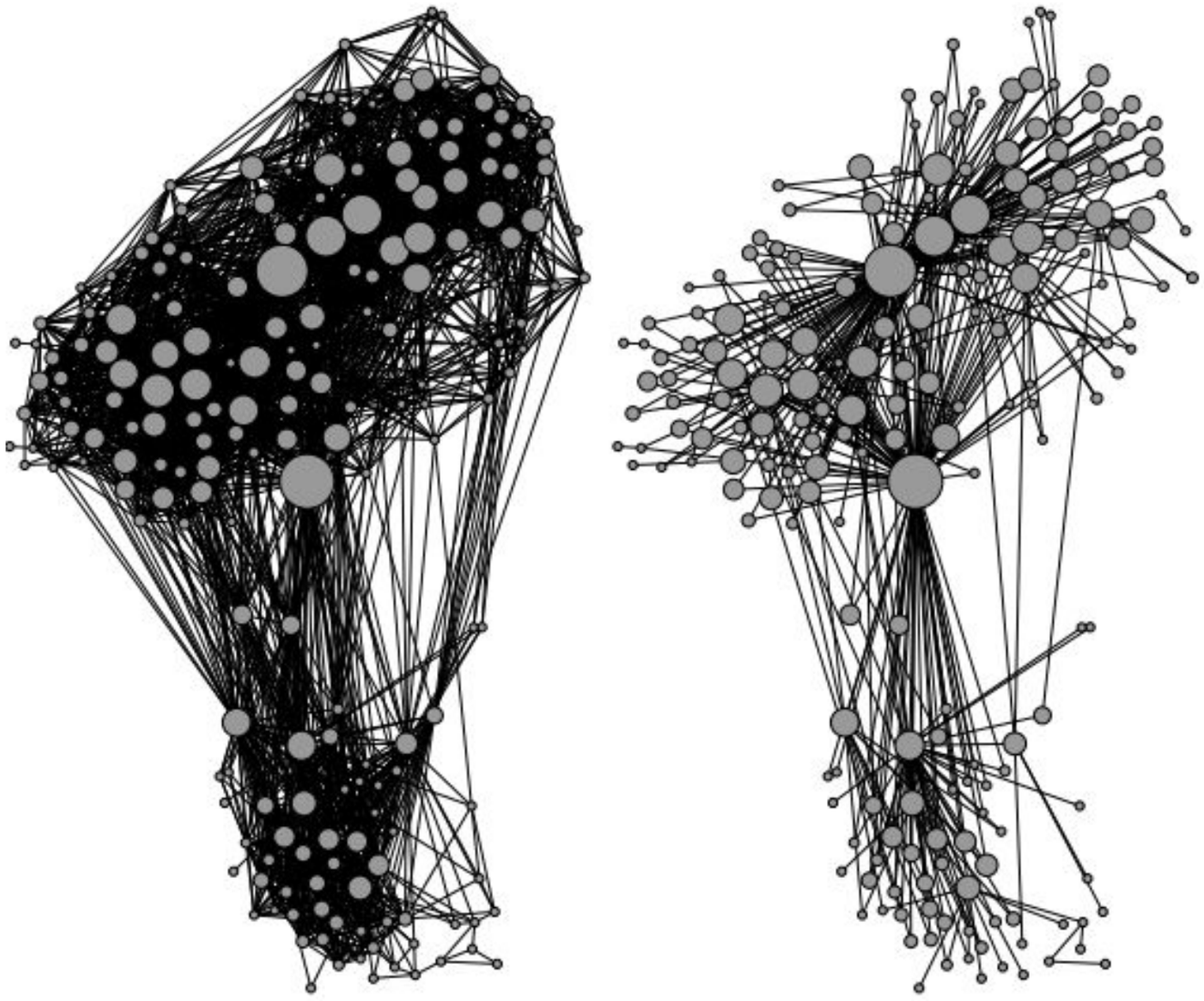
- If we stand in A and set our threshold to be 3, only to the edge to C is preserved.



  - C here is the neighboring node with the highest degree.

  - Hence, through this sparsification technique, neighboring nodes with highest degrees are preserved.

Lindner, Gerd, Christian L. Staudt, Michael Hamann, Henning Meyerhenke, and Dorothea Wagner. "Structure-preserving sparsification of social networks." In Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on, pp. 448-454. IEEE, 2015.



Drawing of the Jazz musicians collaboration network and the Local Degree sparsified version containing 15% of edges.

Local is pruned, global is kept

Lindner, Gerd, Christian L. Staudt, Michael Hamann, Henning Meyerhenke, and Dorothea Wagner. "Structure-preserving sparsification of social networks." In Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on, pp. 448-454. IEEE, 2015.

Algebraic distance sparsification

- It aims to achieve a balance between local and global structure preservation
- It ranks edges based on algebraic distances, and filters them on a parametric threshold.
- The process begins by initializing each node with a vector.

John, Emmanuel, and Ilya Safro. "Single-and multi-level network sparsification by algebraic distance." Journal of Complex Networks 5, no. 3 (2016): 352-388.

Lindner, Gerd, Christian L. Staudt, Michael Hamann, Henning Meyerhenke, and Dorothea Wagner. "Structure-preserving sparsification of social networks." In Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on, pp. 448-454. IEEE, 2015.

Algebraic distance

Algorithm 1: Computing algebraic distances for graphs

Input: Parameter ω, initial vector $x^{(0)}$

1 **for** $k = 1,2,\ldots$ **do**

2 $\quad \tilde{x}^{(k)} \leftarrow \sum_j w_{ij} x_j^{(k-1)} / \sum_j w_{ij} \ \forall i$

3 $\quad x^{(k)} \leftarrow (1\text{-}\omega)x^{(k-1)} + \omega\tilde{x}^{(k)}$

4 **end**

- Vectors are updated in a given number of iterations using a weighted adjacency matrix. At each position they average the corresponding value of the vectors of the adjacent edges.

- This is called a Jacobian overrelaxation process.

- After repeated iterations the values of neighboring vectors converge with each other and updates become small.

- Algebraic distances are calculated for the final values obtained for each vector after a set of given iterations.

$$s_{ij}^{(k)} = | x_i^{(k)} - x_j^{(k)} |$$

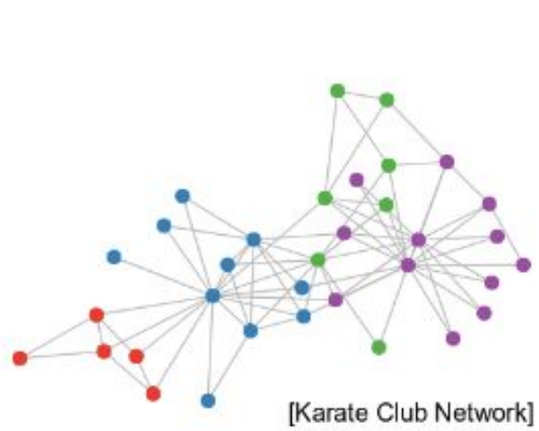
Chen, Jie, and Ilya Safro. "Algebraic distance on graphs." SIAM Journal on Scientific Computing 33, no. 6 (2011): 3468-3490.

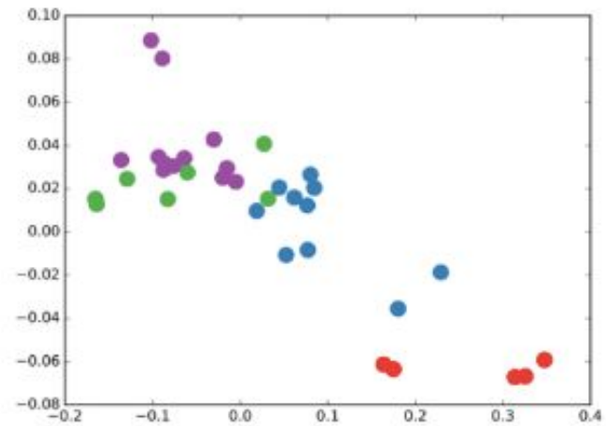
Graph Embeddings

- A technique to represent nodes of a graph as vectors in an Euclidian vector space

- Structure and other inherent properties of the graph are preserved
  - Most notably: node structural similarity

- Graph is accessible to vector-based machine learning methods





Graph embedding of Zakary Karate graph

Embedding/Latent representation Techniques

- Matrix factorization methods
- Random walk methods
- Deep learning methods

Goyal, Palash, and Emilio Ferrara. "Graph embedding techniques, applications, and performance: A survey." Knowledge-Based Systems 151 (2018): 78-94.

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | INF | FAKULTÄT FÜR INFORMATIK

Types of embeddings in our study

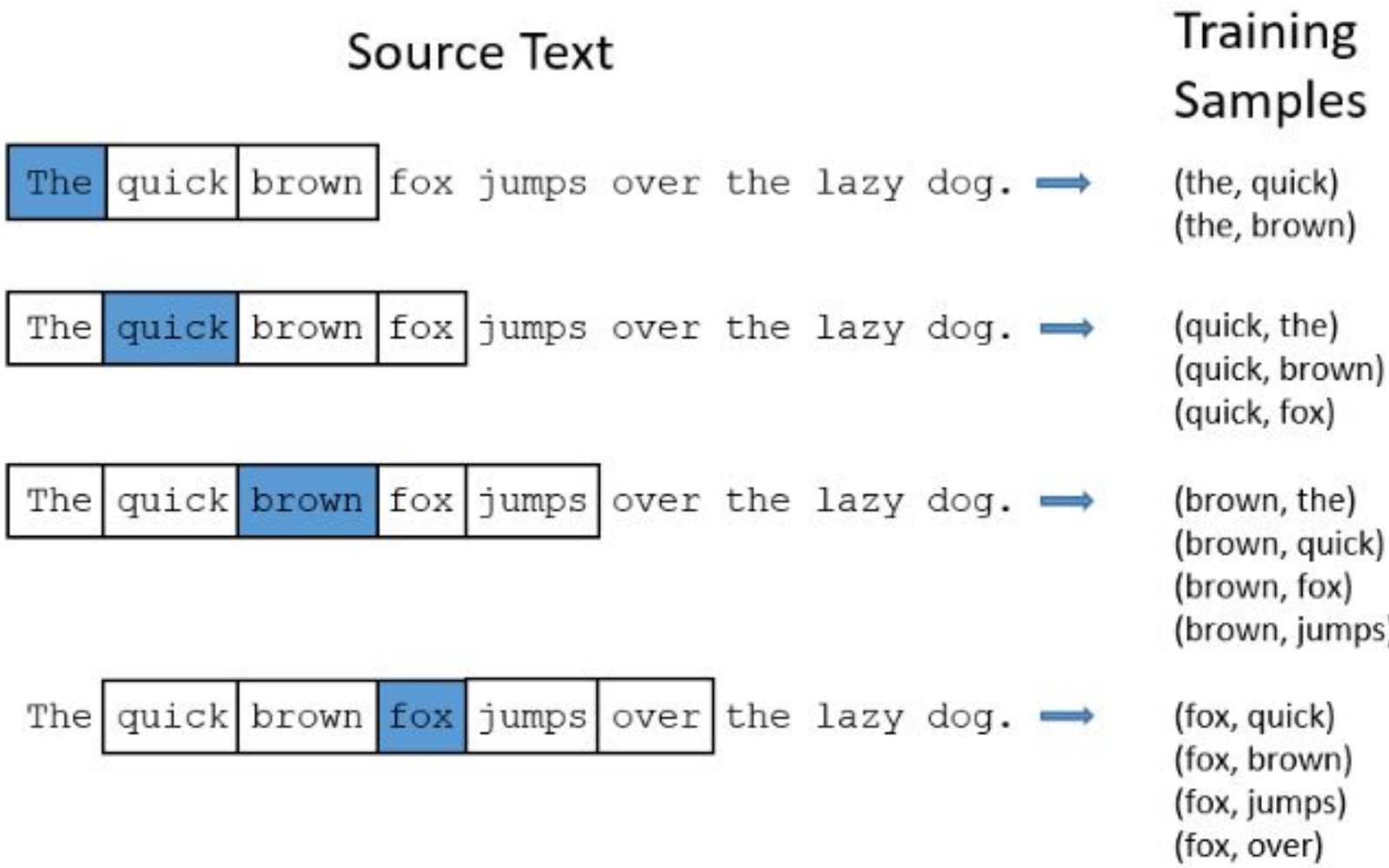
- Laplacian eigenmaps
- Locally linear embedding
- Higher order proximity preserving graph embedding (HOPE)
- Node2Vec

Node2vec

- A way of representing nodes of a graph as vectors in vector spaces
- Uses a skip gram model used by Word2Vec to obtain embeddings
- To obtain training data, node2vec uses biased random walk.

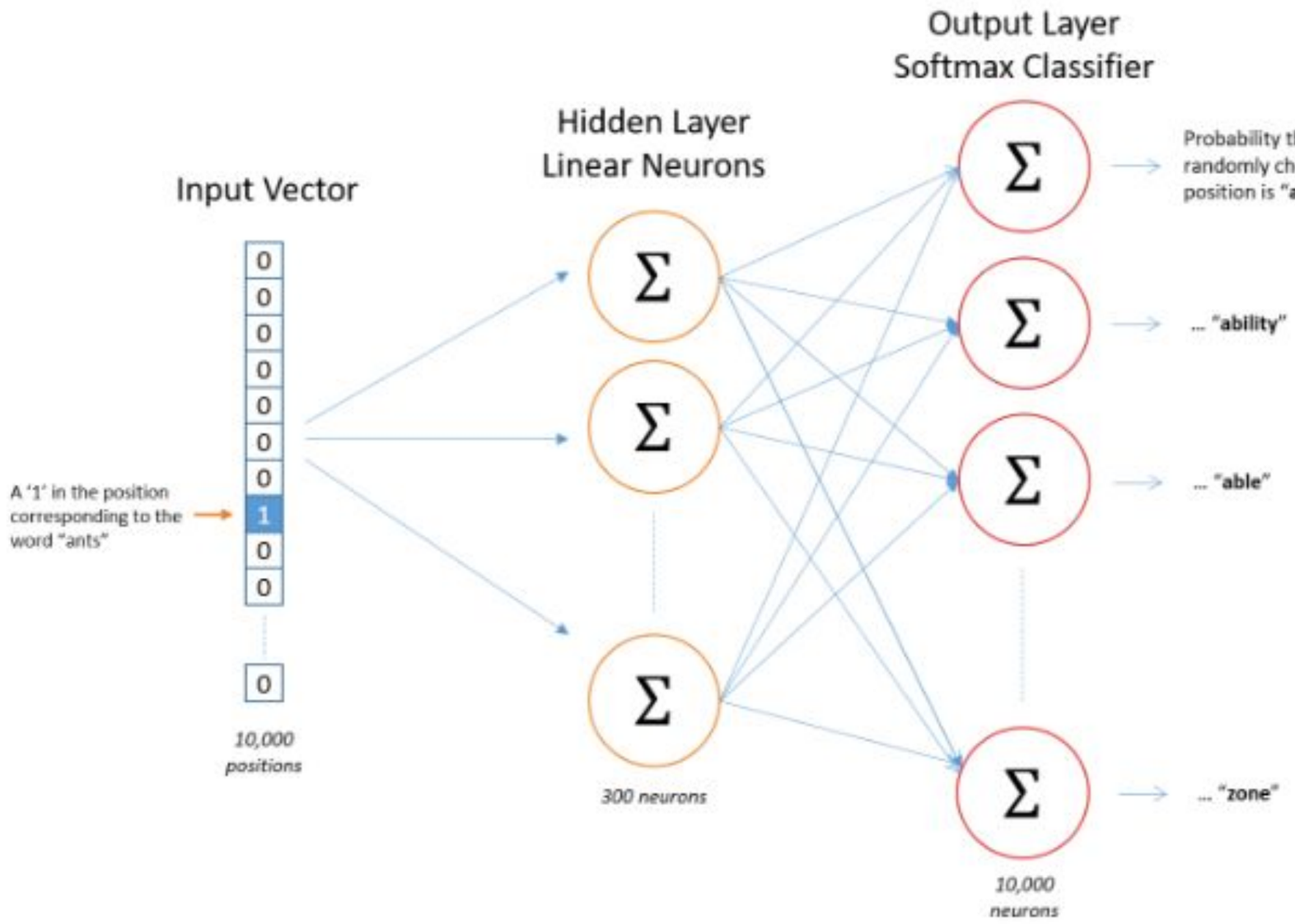
Grover, Aditya, and Jure Leskovec. "node2vec: Scalable feature learning for networks." In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 855-864. ACM, 2016.

An insight into Word2Vec



https://www.knime.com/blog/word-embedding-word2vec-explained

Architecture of neural network for word2vec



A neural network is trained, taking as input each one-hot-encoded word, and as expected output, the probability of words co-occurring in the training data.

The NN creates weights that minimize the loss between the learned prediction and the given probabilities.

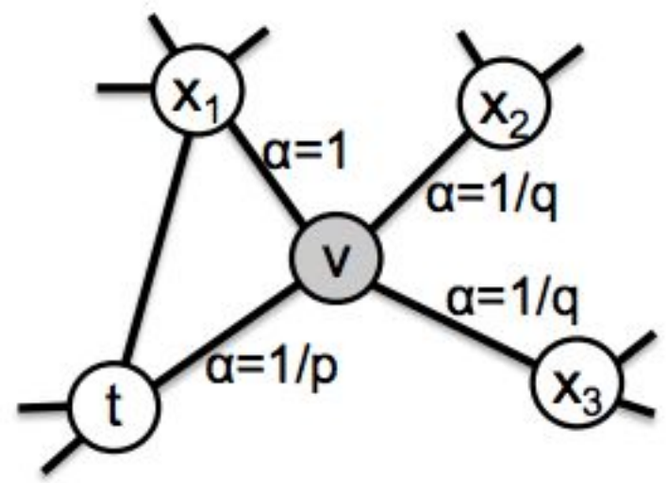
https://www.knime.com/blog/word-embedding-word2vec-explained

Second order random walks for training data

- Consider a random walk that just traverse edge (t, v) and now resides at node v.

- The walk now needs to decide on the next step so it evaluates the transition probabilities πvx on edges (v, x) leading from v

- unnormalized transition probability is given by:



$$\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$$

Where,

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

Grover, Aditya, and Jure Leskovec. "node2vec: Scalable feature learning for networks." In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 855-864. ACM, 2016.

Second order random walks

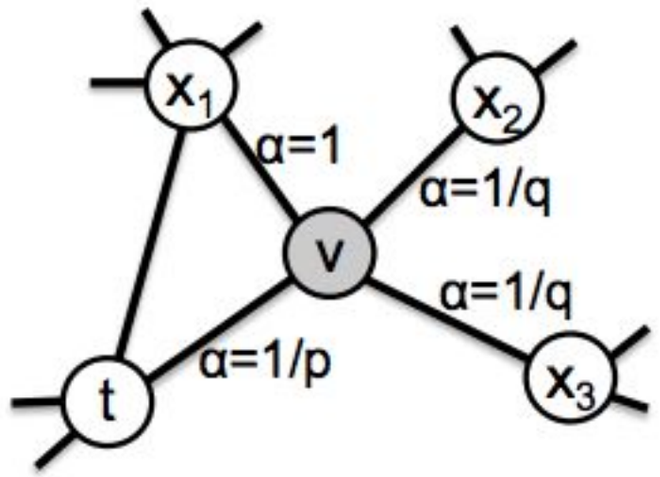
Intuitively,

- Parameter p controls the likelihood of immediately revisiting a node in the walk

- If parameter q is relatively low, walk is more inclined to visit nodes which are further away from the node t

- If q is relatively high, the random walk is biased towards nodes close to node t

- With this approach the training data is collected, and then it is embedded similarly to Word2Vec.



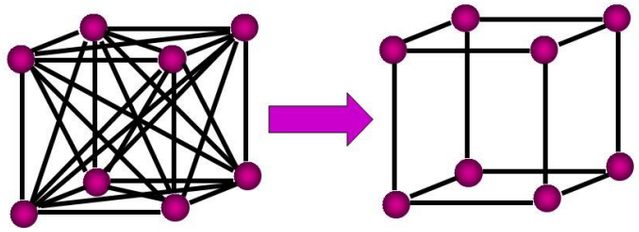
Grover, Aditya, and Jure Leskovec. "node2vec: Scalable feature learning for networks." In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 855-864. ACM, 2016.

4. Implementation

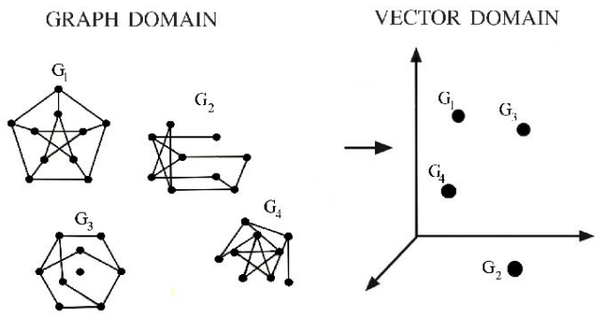
NetworKit
Large-Scale Network Analysis

LDBC

Original Graph



Graph Sparsification



Graph Embedding

palash1992 / GEM

SNAP

neo4j

openCypher

Structural Queries

openCypher

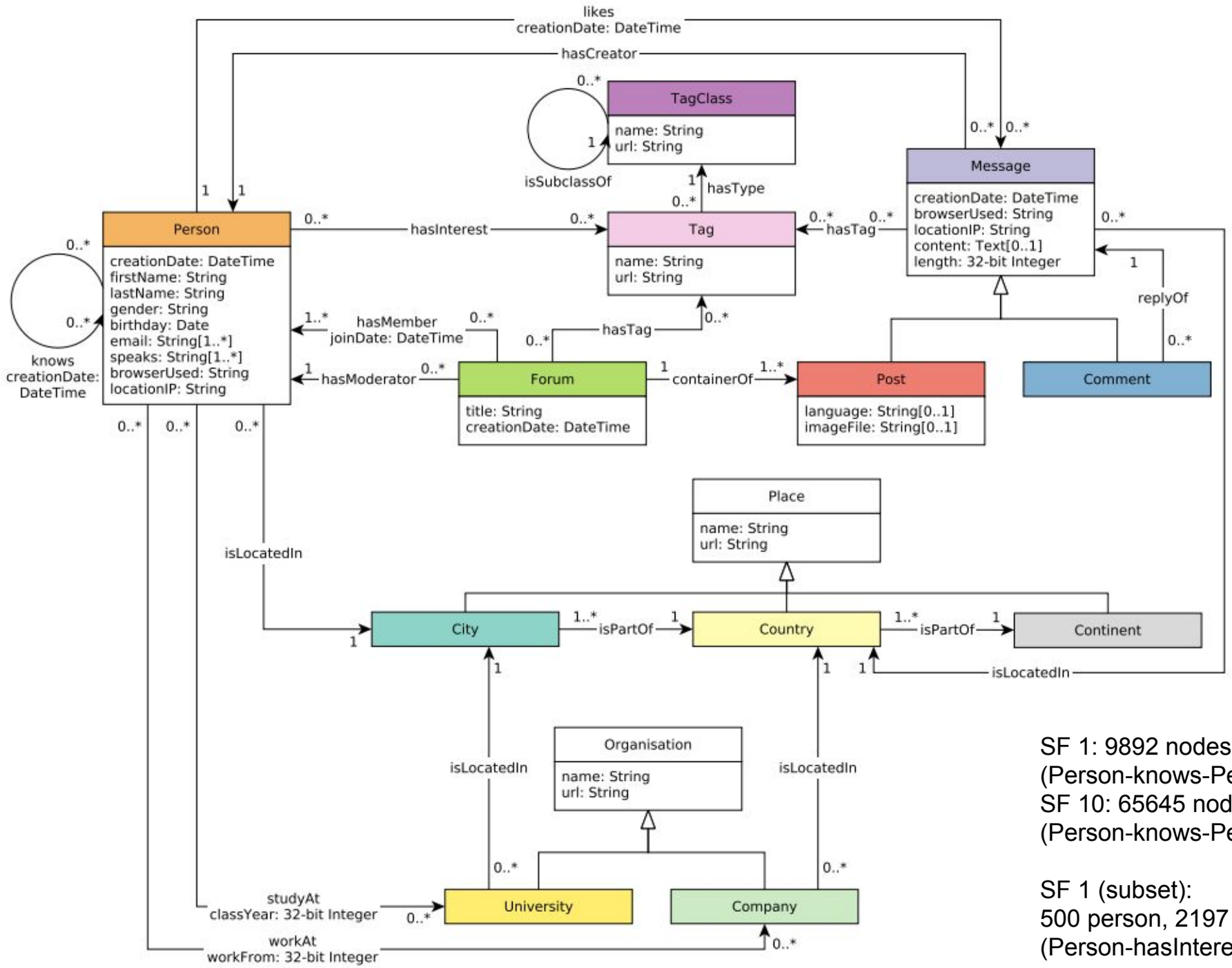
Query over raw data

Structural Node Similarity Queries

Queries with stored procedures

neo4j

Dataset - LDBC SNB data schema



SF 1: 9892 nodes, 180623 edges
(Person-knows-Person)
SF 10: 65645 nodes, 1947294 edges
(Person-knows-Person)

SF 1 (subset):
500 person, 2197 tags, 10157 edges
(Person-hasInterest-Tag)

5. Results and Discussion

Results and discussion

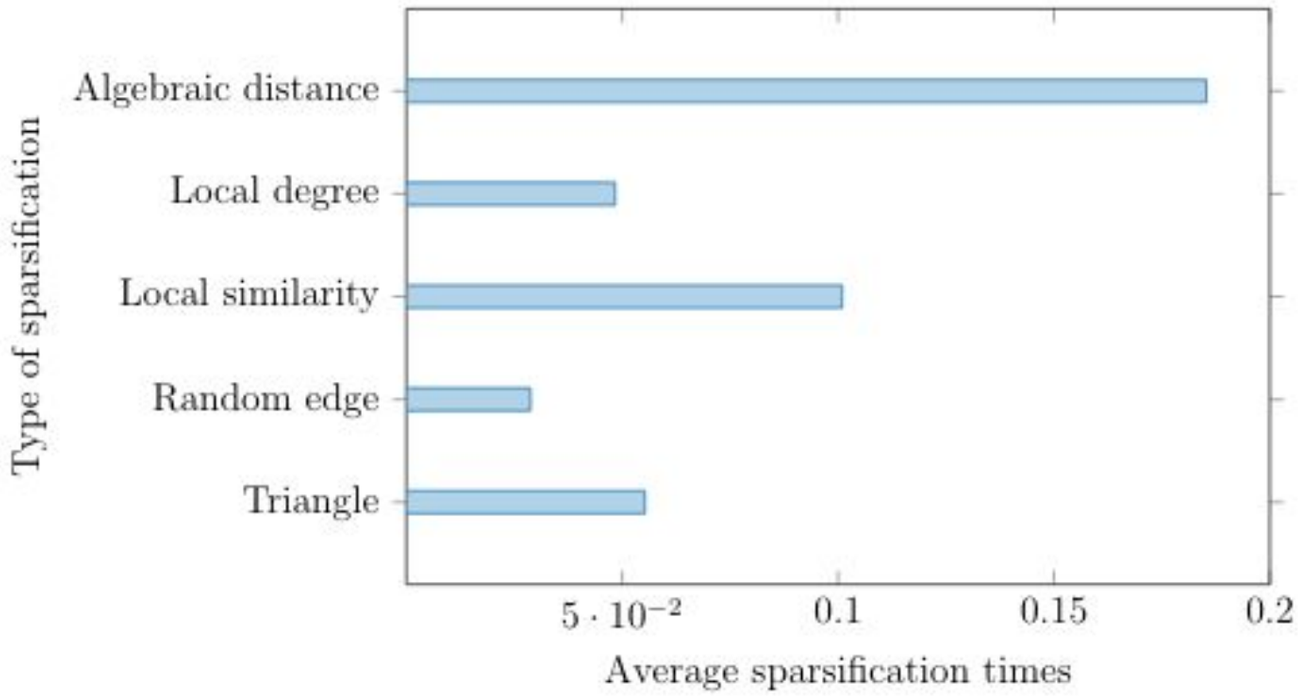
Sparsification technique	# of edges	# of nodes	sparsification ratio
Baseline	180623	9893	
Algebraic distance	178277	9893	0.98
Local degree	43255	9893	0.23
Local similarity	**35022**	**9893**	**0.19**
Triangle	170049	9893	0.94
Random edge	90541	9893	0.5

Sparsification ratios for various sparsification techniques on LDBC SF1 data

Sparsification ratios for various sparsification techniques on LDBC SF10 data

Sparsification technique	# of edges	# of nodes	sparsification ratio
Baseline	1947294	65645	
Algebraic distance	1928513	65645	0.99
Local degree	376240	65645	0.19
Local similarity	**308113**	**65645**	**0.15**
Triangle	1800432	65645	0.92
Random edge	973645	65645	0.49

Sparsification times SF1, SF10

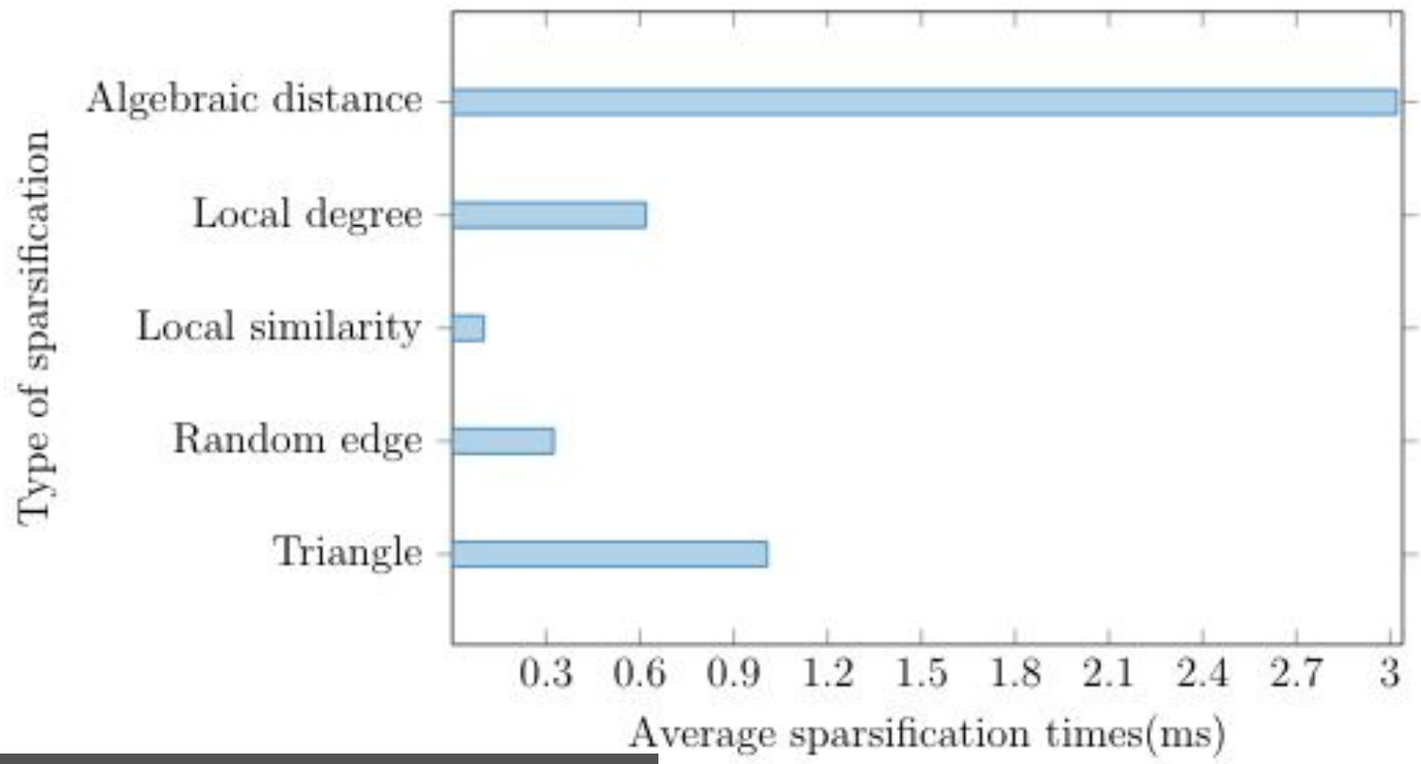


AD is the most time consuming, as expected.

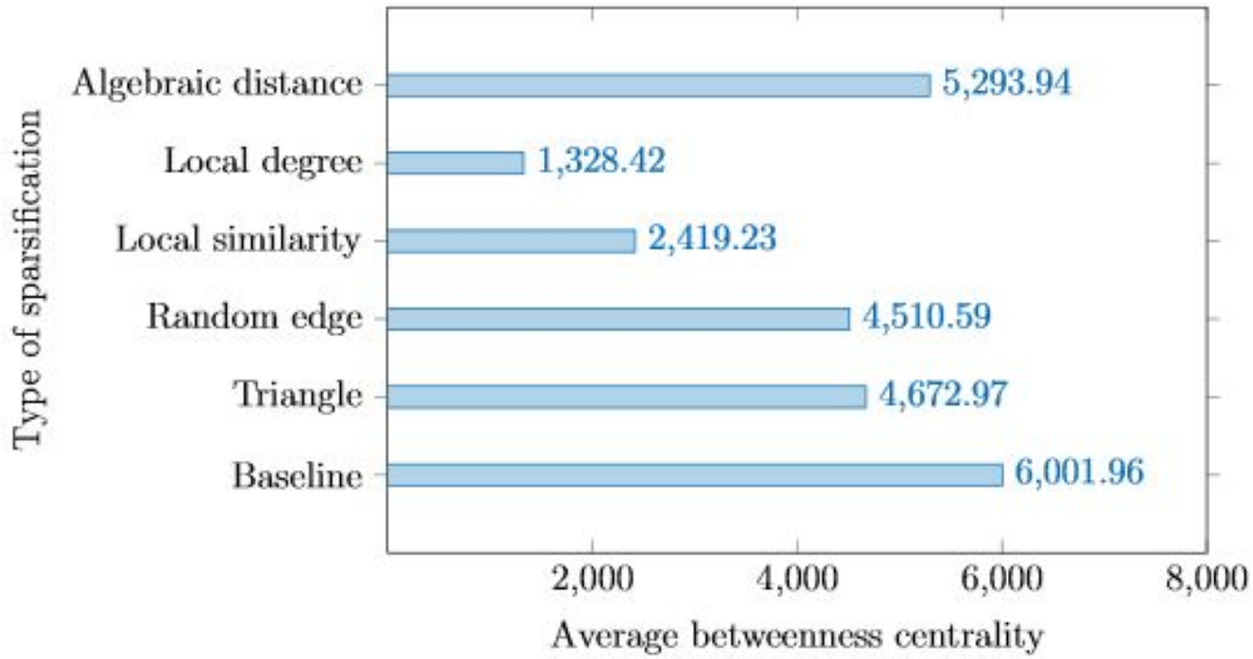
Sparsification times SF1

Sparsification times SF10

RE is outperformed, surprisingly.



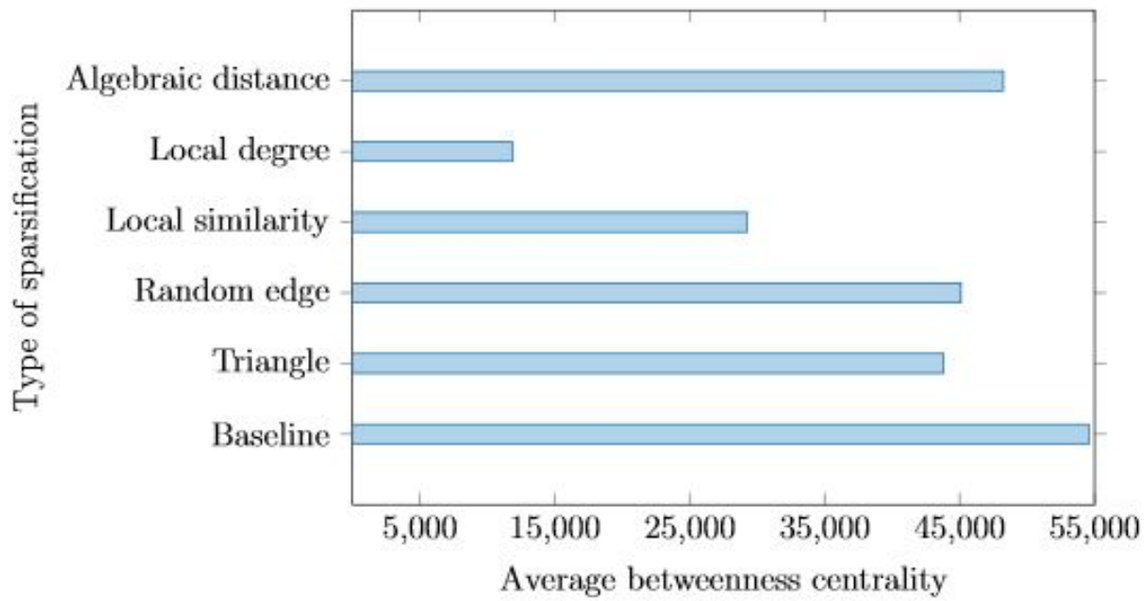
Accuracy: Average betweenness centrality



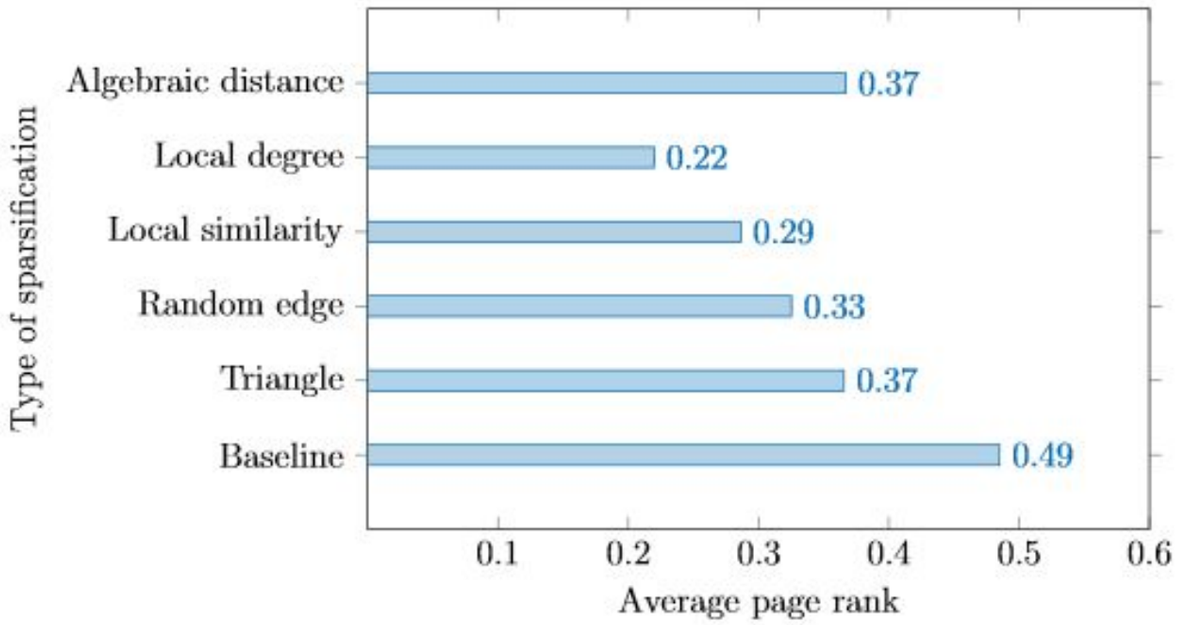
Average betweenness centrality of SF1 data

Local methods perform worst, AD performs best.

Average betweenness centrality of SF10 data



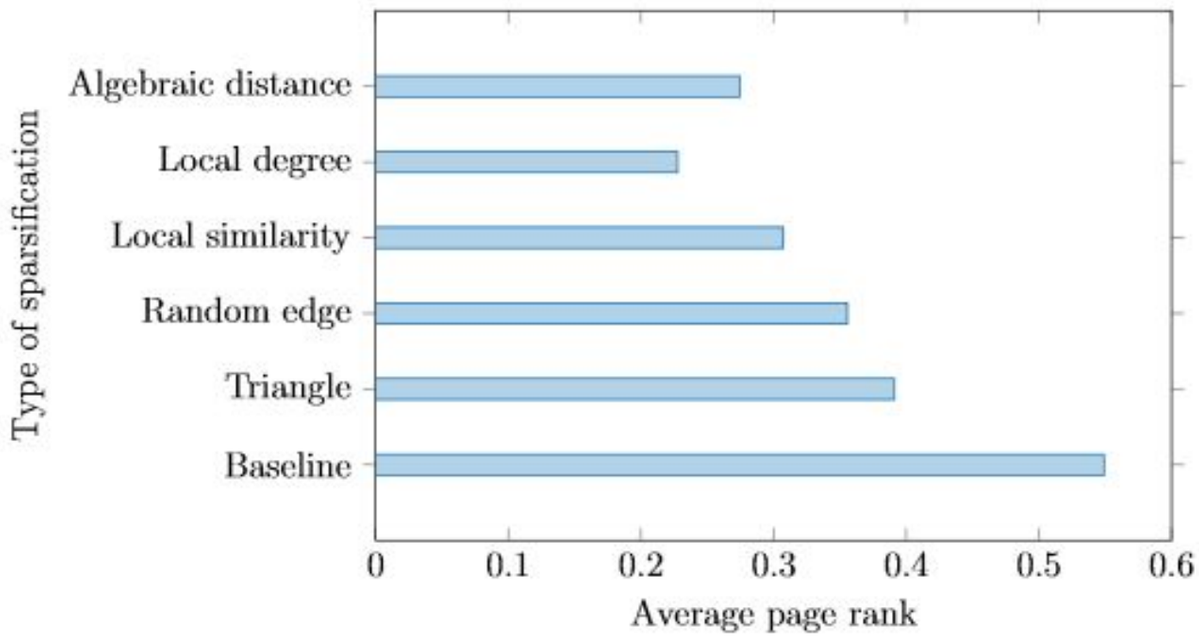
Average pagerank



Average pagerank for SF1 data

Local methods perform worst,
AD performs best.

Average pagerank for SF10 data



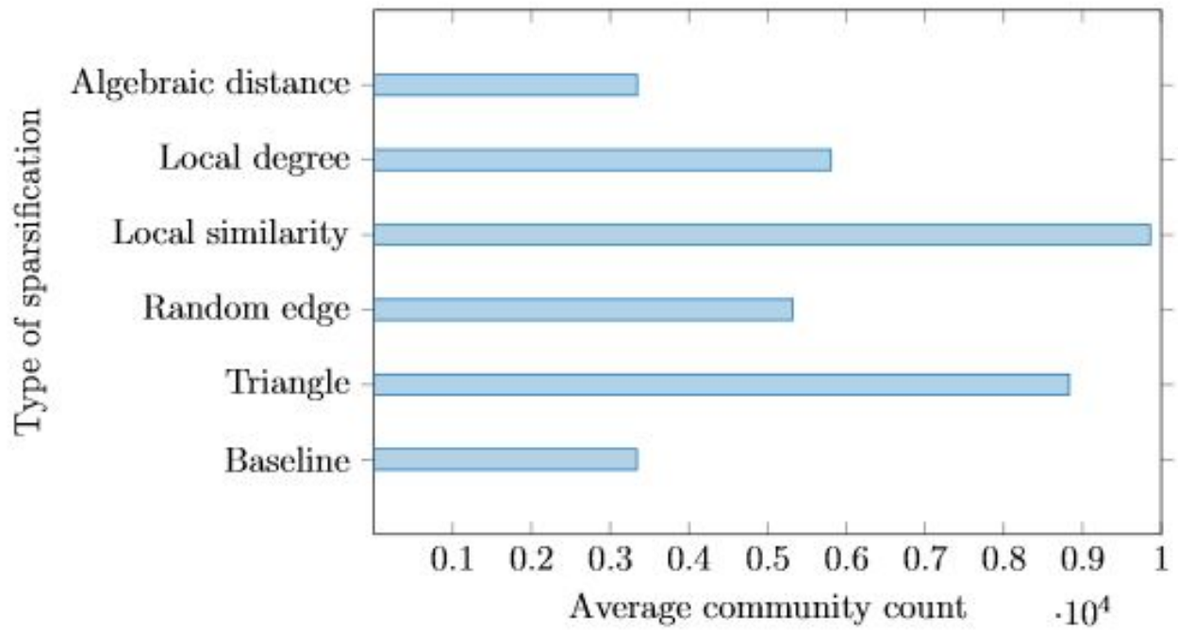
Average community count

Sparsification technique	approximate sparsification ratio	Community count
Baseline	N.A	809.16
Algebraic distance	0.98	809.26
Local degree	0.23	1220.16
local similarity	0.19	2041.38
random edge	0.5	1605.56
Triangle sparsifier	0.94	1637.16

Average community counts for SF1 data

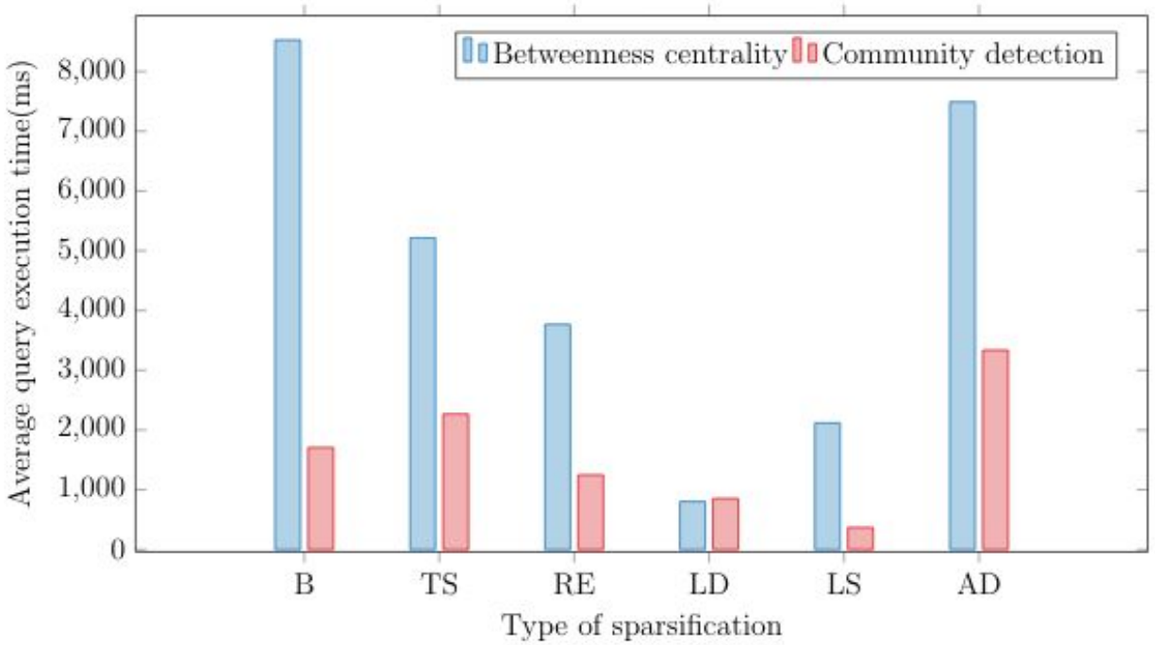
Local similarity performs worst, AD performs best, LD is competitive.

Average community counts for SF10 data



Connected components and partition sizes are preserved for all except RE and TS

Execution times of queries SF1

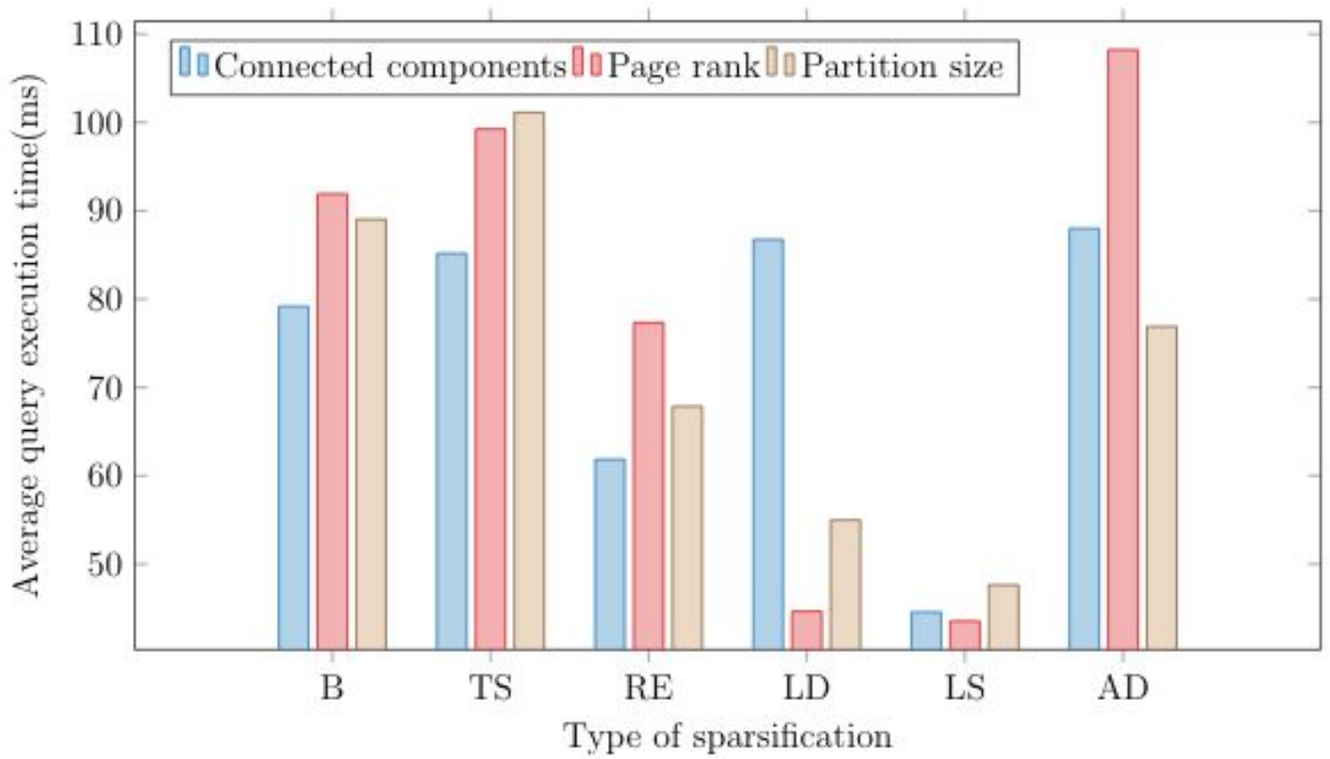


All methods improve betweenness centrality

AD, TS deteriorate execution time of community detection

Benefits only for
connected components using LS
(with RE results are not accurate)

RE provides reasonable runtime for page rank.



Execution times of SF10

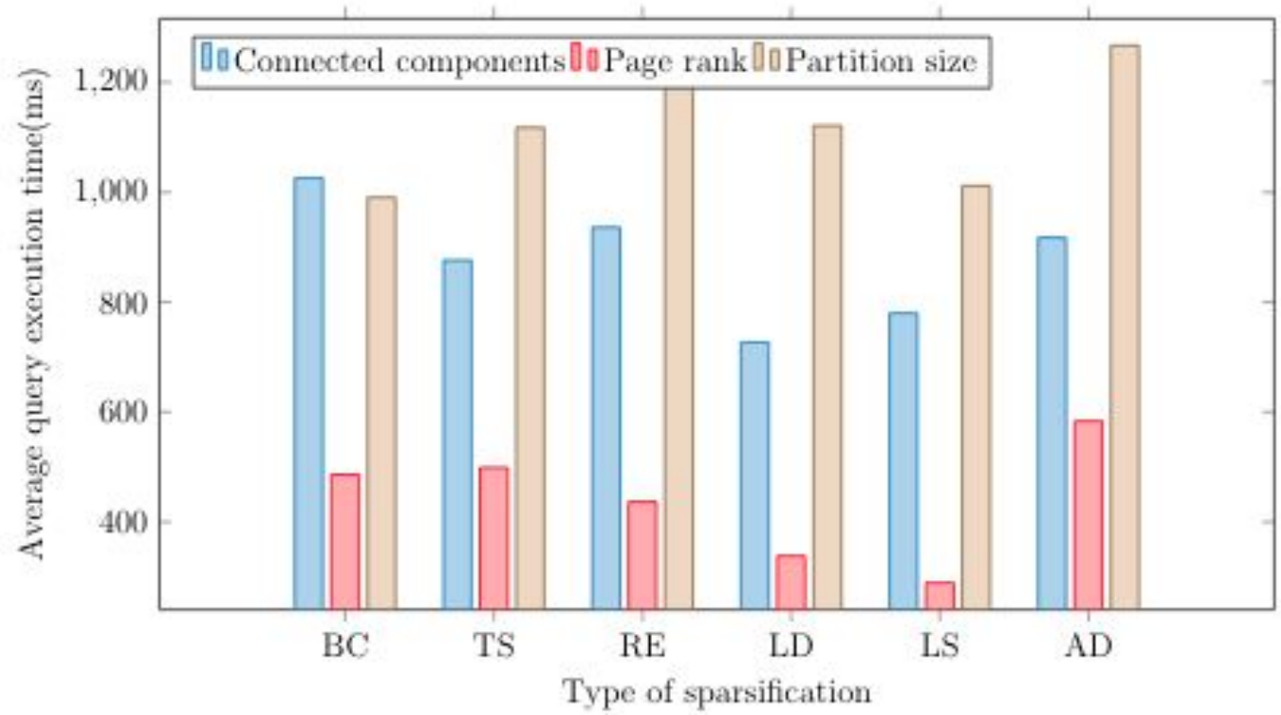


Figure 4.10: Average execution times for different types of sparsification

Speed ups for page rank query on local sparsified versions

Connected components works better for all techniques than baseline.

Notable speed ups for betweenness centrality on all types of sparsification

Community detection: All methods bring some small improvements.

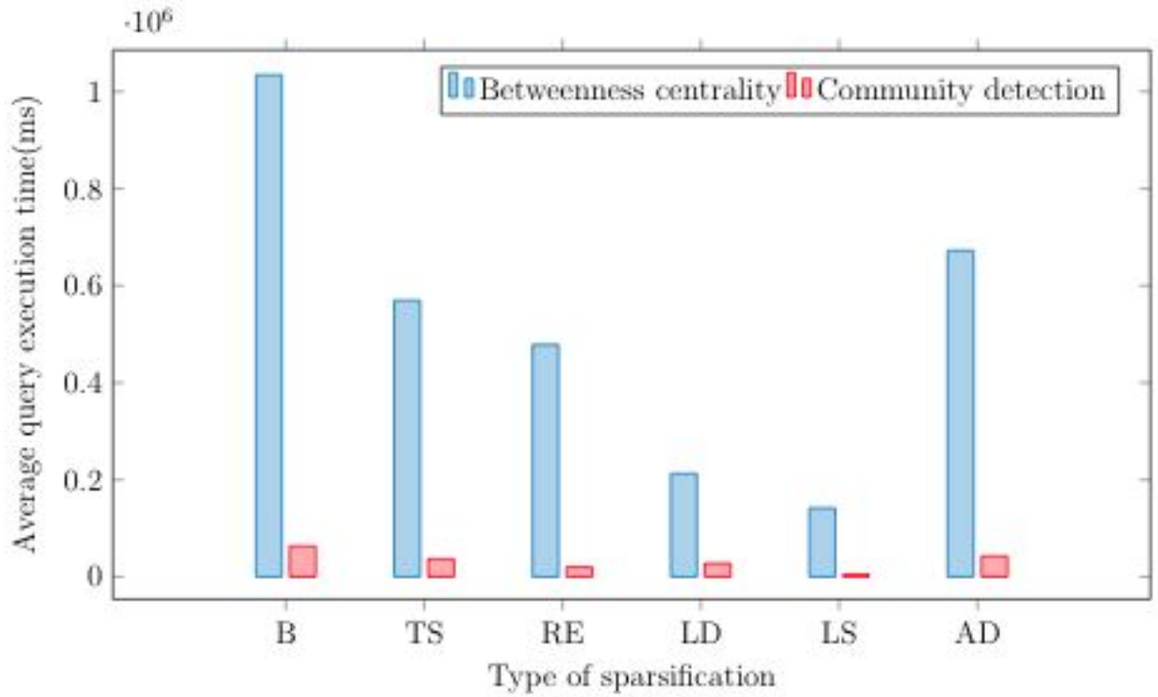
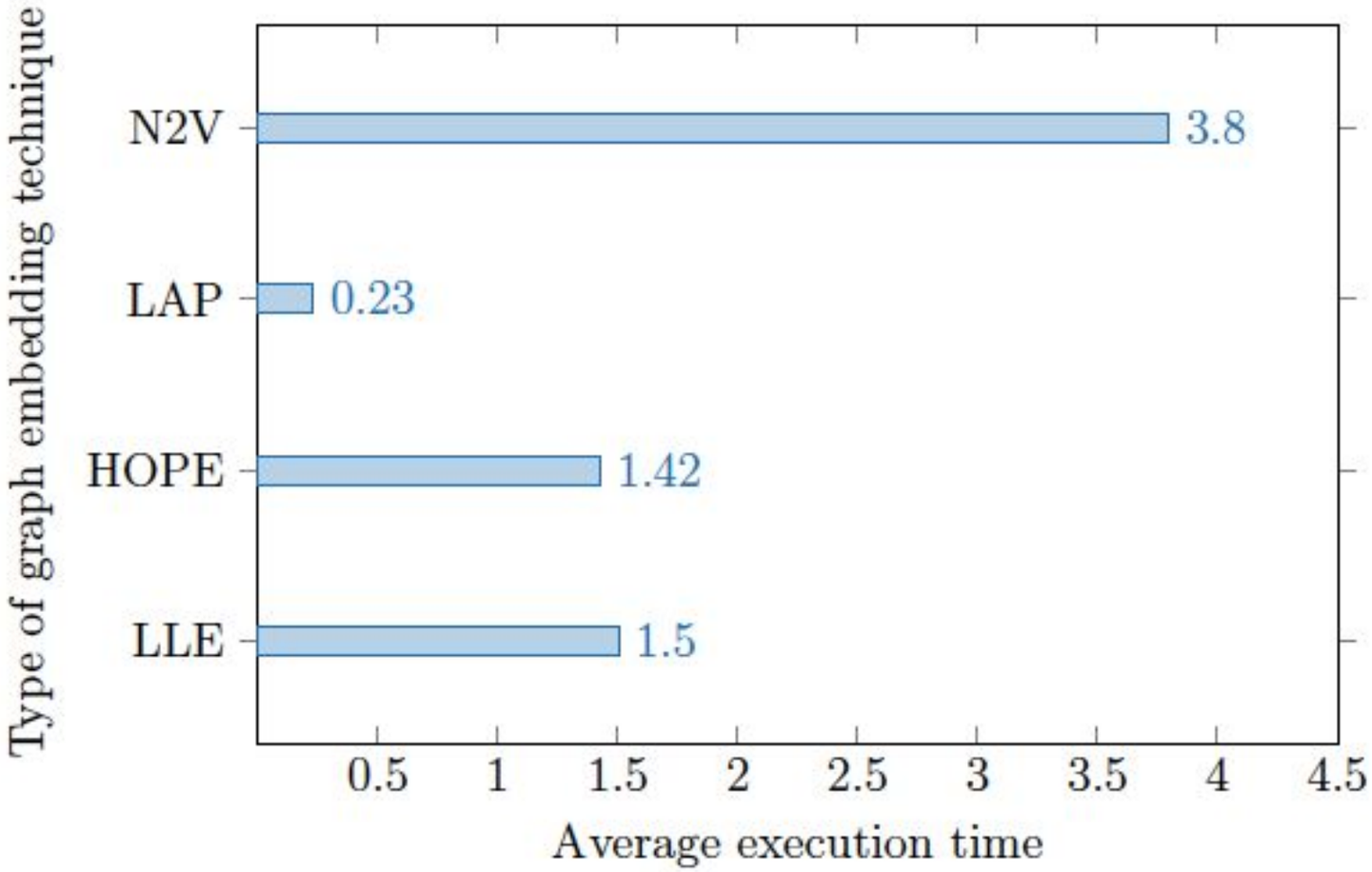


Figure 4.11: Average execution times for different types of sparsification

Embeddings



LAP takes the least time, N2V provides more informative results

1. The average time taken to calculate pairwise cosine similarities using self written python (on default config: 5349 ms) code is significantly less when compared to the time taken to calculate the same via stored procedure (14455 ms).

2. Time taken to calculate pairwise similarities on unembedded data (32985.6 ms) is nearly 6 times higher compared to average execution using self written python code on the respective embeddings of nodes.

6. Conclusion and Future Work

Conclusions

The most important takeaways that everyone should remember:

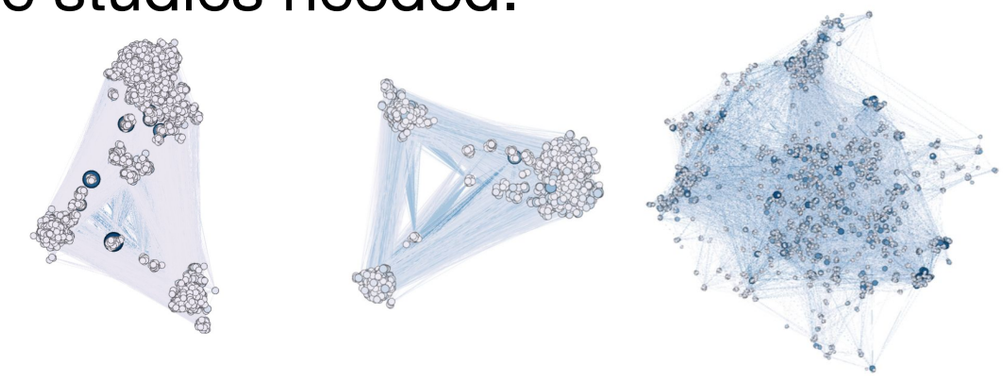
- Summary creation:
  - AD takes more sparsification time than others.
  - The runtime of sparsifications is not excessive (none takes more than a few minutes)
  - The embedding process with GEM is not scalable at the moment.
    - Embedding procedures are currently too slow.

Conclusions

The most important takeaways that everyone should remember:

- Accuracy:
  - **Most methods** preserve number of *connected components*.
  - Only **algebraic distance** sparsification preserves the *community count*.
  - For r*anking and centrality* more evaluations are needed.
  - **All methods are expected to scale well in accuracy as data grows.**
    - But we do not report evaluations on this.

  - In *visualization*, **all methods**, except local degree and local similarity, seem to preserve structure of graph. More studies needed.
  - **Node2Vec embeddings** seem to give better *similarities* than over un-embedded data. More studies needed.



Conclusions

The most important takeaways that everyone should remember:

- Execution times:
  - Notable speedups for *betweenness centrality* for **all** the sparsifications.
  - **Local similarity** shows good speed ups for *connected components*, *page rank, and partition size.*
  - **As data grows, querying over the sparsified data is comparatively better than the baseline.**
  - **For embeddings:** The average time taken to calculate *pairwise cosine similarities* is significantly less when compared to the time taken to calculate via stored procedures or unembedded data
    - Better algorithms are required in Neo4j to support the top pair-wise cosine similarity
- The embedding process with GEM is not scalable at the moment.
- AD takes more sparsification time than others.

Future Work

- Trade-off analysis between memory footprint, summary creation time, accuracy and query execution time is needed.
- Algorithms to deal with properties could be a valuable contribution
- Graph stream embedding (with change awareness) could be a high impacting area to explore
- Diverse edge types for summarization is also an unexplored area of research
- Integration of summarization into graph databases: caching with sparsification, others.
- Analysis to provide error and operation time bounds could be productive.

Thank you

Questions, thoughts?

References

- Liu, Yike, Tara Safavi, Abhilash Dighe, and Danai Koutra. "Graph Summarization Methods and Applications: A Survey." ACM Computing Surveys (CSUR) 51, no. 3 (2018): 62.

- Sahu, Siddhartha, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M. Tamer Özsu. "The ubiquity of large graphs and surprising challenges of graph processing." Proceedings of the VLDB Endowment 11, no. 4 (2017): 420-431.

- Goyal, Palash, and Emilio Ferrara. "Graph embedding techniques, applications, and performance: A survey." Knowledge-Based Systems 151 (2018): 78-94.

- Lindner, Gerd, Christian L. Staudt, Michael Hamann, Henning Meyerhenke, and Dorothea Wagner. "Structure-preserving sparsification of social networks." In Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on, pp. 448-454. IEEE, 2015.

- Chiba, Norishige, and Takao Nishizeki. "Arboricity and subgraph listing algorithms." SIAM Journal on Computing 14, no. 1 (1985): 210-223.

- Grover, Aditya, and Jure Leskovec. "node2vec: Scalable feature learning for networks." In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 855-864. ACM, 2016.

Extra Slides

Datasets used for each technique:

Sparsification

- LDBC SNB Social network dataset
- person, person-knows-person
- undirected graph
- SF1 and SF10

Embeddings

- LDBC SNB Social network dataset
- person, person-has-interest, tag
- directed graph
- SF0.05 with 500 persons

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | INF | FAKULTÄT FÜR INFORMATIK

Local similarity sparsification

- This sparsification technique is based on Jaccard similarity measure over the set of nodes in a given graph.

- It is given by:

$$Sim(A, B) = |A \cap B| / |A \cup B|$$

Where A and B are two sets

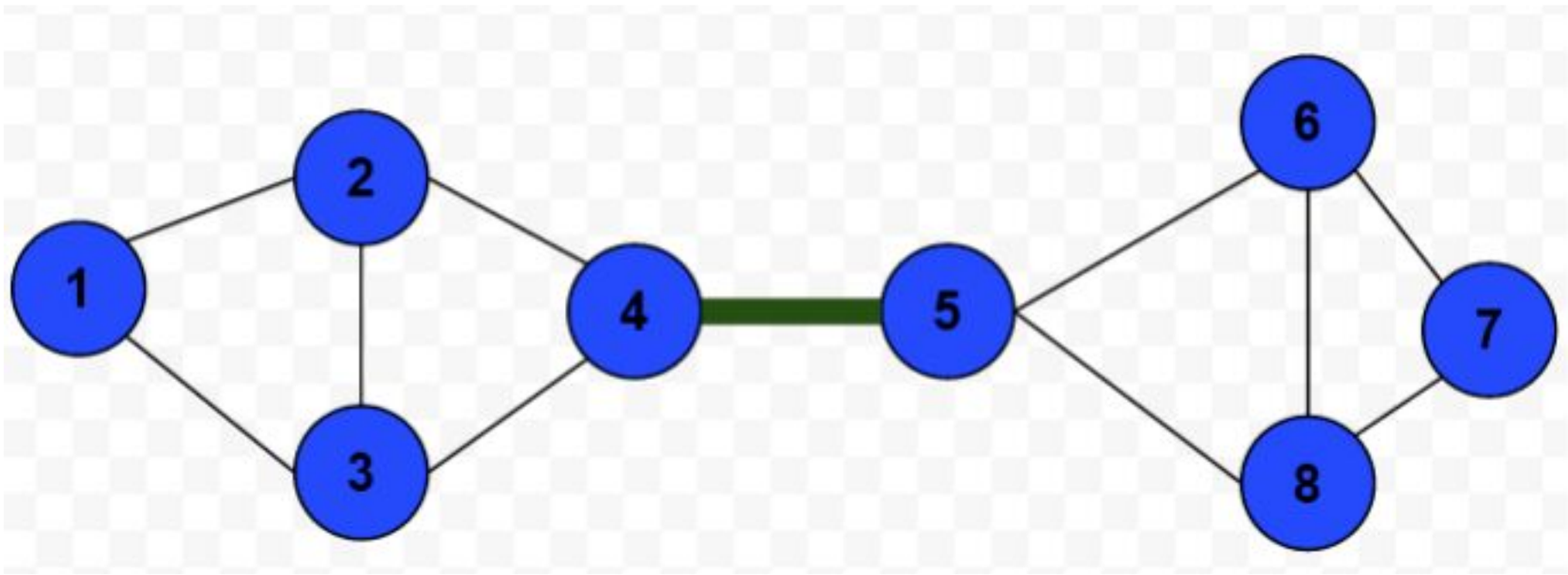
This strategy keeps the local but not the global

Satuluri, Venu, Srinivasan Parthasarathy, and Yiye Ruan. "Local graph sparsification for scalable clustering." In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pp. 721-732. ACM, 2011.

Triangle count sparsification

- Edges are removed based on the number of triangles they are present in.

- Keeps local, loses global structures.



Chiba, Norishige, and Takao Nishizeki. "Arboricity and subgraph listing algorithms." SIAM Journal on Computing 14, no. 1 (1985): 210-223.

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | INF | FAKULTÄT FÜR INFORMATIK

Research aim

- Evaluate two structure preserving summarization techniques namely scarification and graph embeddings on static graphs of different scale factors
- Attempt to quantify scalability of the techniques for improving graph tasks like community detection, page rank, betweenness centrality, connected components and pairwise cosine similarity

Prototypical implementation

Sparsificatio
n

- Networkit python library
- facilitates the implementation of different sparsification techniques
- Use cypher queries for community detection, connected components, page rank, partition size and betweenness centrality on both sparsified and unsparsified data on Neo4j database
- visualize the sparsified data

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | INF | FAKULTÄT FÜR INFORMATIK

Prototypical implementation

Embeddings

- GEM python library
- facilitates implementation of various embedding techniques
- Derive 2-d embeddings from all embedding techniques
- calculate pairwise cosine similarity manually
- calculate pairwise cosine similarity in neo4j using stored procedures