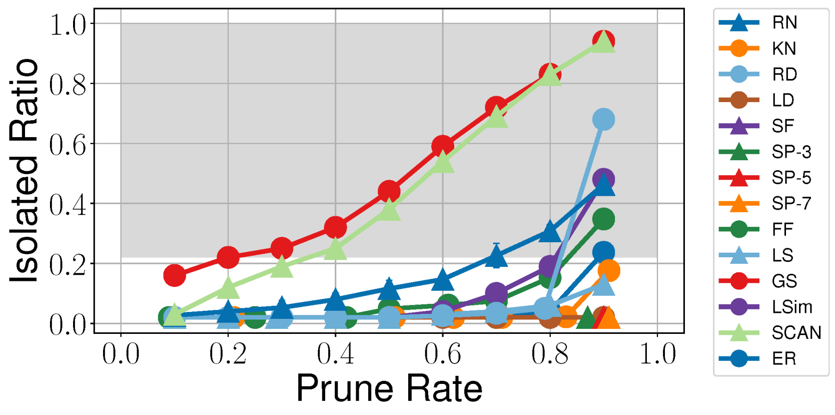
Summary of "Demystifying Graph Sparsification Algorithms in Graph Properties Preservation" by Y. Chen et al.

summary by Vasileios Papastergios (Academic ID: 3651)

April 13, 2024



Aristotle University of Thessaloniki
School of Informatics

SCHOOL OF INFORMATICS

Course name:	Big Data Algorithms
Course ID:	NIS-08-05
Semester:	8th
Professors:	Apostolos Papadopoulos
	Georgios Christodoulou

Contents

1 Introduction

In this section we provide basic information about the current document. In section 1.1 we explain the purpose of the current document, as well as the context within it is written. In section 1.2 we provide a short overview of what will be analyzed in the document. In section 1.3 we share useful definitions, so that domain-agnostic readers can catch up with the rest of the analysis.

1.1 General Information

The present document serves as formal report for the literature assignment in the course of Big Data Algorithms. The author attended the course during their 8th semester of BSc. studies at the School of Informatics, AUTh. The topic of the assignment is a *summary* of the research paper "Demystifying Graph Sparsification Algorithms in Graph Properties Preservation" [1], authored by Y. Chen et al. We refer to them as *"the authors"* or *"they"* and to their work as *"the paper"* for the rest of the current document. The paper is accepted to be presented in the Proceedings of 50th International Conference on Very Large Databases (VLDB 2024). We clearly state that our work is limited to summarizing the key points and results provided by the authors in the paper. All the work described in the pages that follow corresponds to effort conducted exclusively by the authors and, by no means, by us.

1.2 Abstract

In this document we provide a summary of the work conducted by Y. Chen et al. in [1]. We present in a concise way the main checkpoints of the authors' contribution to the research community. Special emphasis is given in the results provided, concerning the sparsification algorithms and their performance. In a line, the authors provide analysis, experiments, as well as a software framework for evaluating sparsification algorithms on real-world graphs. Based on their experimental results, they conclude that the sparsification algorithms are not "one size fits all" and thorough study is required before selecting any of them, depending on the downstream task at hand.

1.3 Preliminaries

In order to understand and analyze the approach adopted by the authors and the produced results, we consider it useful to provide definitions, explanations and examples for fundamental concepts appearing throughout the paper. The experienced reader can skip this content, moving straight into section todo.

1.3.1 Graph sparsification

Graph sparsification is a technique that approximates an arbitrary (given) graph with a sparse graph. The sparse graph is, essentially, a subset of the initial's graph edges and/or vertices. The most common case in sparsification algorithms is that the produced sparse graph consists of all the vertices and only a subset of edges of the original graph. There is a great number of proposed strategies in order to decide which edges to preserve and which to discard from the original graph, leading to a wide range of sparsification algorithms. A **sparsification algorithm**, thus, is an algorithm that performs graph sparsification on a given graph, by means that it takes a graph as input and produces a sparse approximation of it, based on some edge selection strategy. We elaborate more on various sparsification algorithms proposed in literature in section todo.

The majority of sparsification algorithms provide control over the **prune rate**, i.e. the fraction of the original edges that are preserved in the output graph. For instance, a sparsification algorithm with prune rate equal to 0.5 retains half of the original edges to the output. The strategy that is adopted for edge selection (i.e. the sparsification algorithm) is not related with the prune rate. That means multiple algorithms can operate with multiple prune rates (one at a time) providing a different output on every run. Based on that, we can argue that the prune rate is a *hyperparameter* for the sparsification algorithms.

Although the most sparsification algorithms provide control over the prune rate, there are certain algorithms that do not. This is due to the underlying edge selection strategy of these algorithms. For example, one sparsification algorithm produces as output a spanning tree of the given graph, i.e. a connected, acyclic graph that covers all vertices of the original graph. In this case, the algorithm has no control over the prune rate, since the sparse graph that is produces has fixed number of edges (equal to the number of vertices minus one). However, only a minority of graph sparsification algorithms cannot control the prune rate. The authors examine 12 sparsification algorithms, of which only 2 of them belong to this category.

In the case of the latter algorithm, the output of the sparsification process is **deterministic** (i.e. the same every time) for a specific input graph. This is the case for several sparsification algorithms, but not for all of them. In particular, there are **randomized sparsification algorithms**, whose execution is based on selecting some form of random number that affects the final output graph. In these cases, the sparse graph that is produced may not be the same in every execution, given a specific input graph. Evaluating the performance of such algorithms is not straightforward. The authors address this problem by executing the randomized algorithms many times (100) and calculate the mean value to draw safer conclusions.

1.3.2 Graph metrics

A **graph metric** is usually a number that represents some of the graph properties. Examples of graph metrics are the distribution of the degree of the vertices, the vertex eccentricity, the centrality of some kind (e.g. betweeness centrality) of its vertices, etc. It is profound that removing several edges from a graph affects its properties and, consequently, its metrics. An ideal sparsification algorithm needs to achieve a high prune rate while keeping the behavior of the downstream task as close to that of the original full graph. The authors investigate the extent to which several graph metrics are affected, when different sparsification algorithms are executed on real-world graphs.

2 Research context and motivations

Graphs are everywhere. They provide great expressiveness in representing complex relationships between entities. Social networks are a great application field where graphs can be a powerful tool to model the interaction between individuals but certainly not the only one. Graphs are also used for chemical and biological networks (e.g. modeling the interaction between protein units), as well as communication and transportation networks. A wide variety of graph algorithms has been proposed over the years, with algorithms such as Dijkstra's, Prim's, Bellman-Ford's to be the most well known among hundreds.

Despite their great attributes and usefulness, graphs come with inherent drawbacks, especially due to their big size. Living in the era of Big Data, a real-world graph can easily get out of handle in terms of the memory required to store and process it efficiently. This aspect of problems in graph processing correspond to "Volume" attribute from the 3 V's of Big Data. However, "Veracity" is applicable as well, since real-world applications usually operate in an *online* mode, i.e. the vertices and/or edges of the graph are not known from beforehand, but they dynamically arrive as a stream. Researchers have proposed many solutions over the years, however their main disadvantage is the fact that they are domain-specific, require special software/hardware and/or are applicable to certain graph types only.

A more general and widely accepted solution to the problems emerging in graph processing is graph sparsification, in a try to reduce the size of the graph, while at the same time preserving some of the graph attributes. There is a large number of sparsification algorithms, each one employing a different edge selection strategy, aiming in the preservation of different graph attributes. However, despite the large number of sparsification algorithms and graph metrics, there was not much research work conducted focusing on the connection between the use of specific sparsification algorithms and some specific metrics. The **research questions** that motivated the authors to produce their work in the paper can be summarized in the following questions:

1. How specific sparsification algorithms affect graph metrics and downstream tasks?

2. Is there a sparsification algorithm that excels in the preservation of a wide range of graph metrics?

3 Setting of the conducted research

In order to find answers in the above questions, the authors conducted an **extensive experimental study**, reported the results and provided insights on them. In this section we concisely summarize the execution setting of the study.

We recall that the authors wanted to reveal the connection between the many sparsification algorithms that have been proposed in the literature with the (also many) graph metrics that can be measured on graphs. Their approach was to select **12 representative graph sparsification algorithms** and construct an evaluation setting in order to compare them in terms of preserving graph properties.

Graph metrics are the key concept in the evaluation setting created. In particular, the authors selected **16 widely-used graph metrics**, aiming at performing an all-to-all evaluation. In simple words, for every graph sparsification algorithm, all metrics are computed in both the original and the sparse graph, comparing the results with one another. Lastly, in order to connect the evaluation setting with real-world problems, the authors selected **14 real-world input graphs** spanning various categories, exhibiting diverse characteristics, sizes, and densities. The setting of the experimental study, thus, consists of the following three modules shown in figure 1.

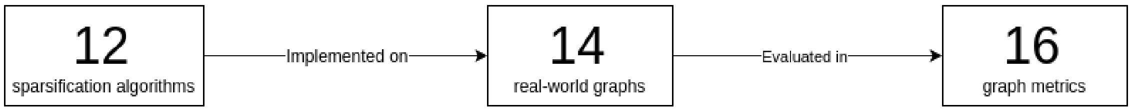


Figure 1: The setting of the experimental study

In order to assist the evaluation process, the authors also developed a framework, written in Python programming language. The framework provides

In the sections that follow we get a deeper insight into the building blocks of the evaluation setting. In section 3.1 we shortly present the sparsification algorithms used in the experimental study. In section 3.2 we list the graph metrics used to evaluate the performance of the algorithms. In section 3.3 we report the real-world graphs used by the authors to executes their experiments.

3.1 Graph sparsification algorithms

Over the past years, many sparsification algorithms have been proposed by researchers. Each one of them focuses on the preservation of specific graph attributes. The authors select 12 representative sparsification algorithms to examine and evaluate. The sparsification algorithms investigated in the paper are presented in the following list. For every algorithm, we provide its full name, its short name used in the paper's result plots to ease the interested reader, as well as a short description of its execution. For non-deterministic algorithms, we clearly state that they are randomized. If not stated, the algorithm is deterministic.

- **Random (RN)**: randomly samples a number of edges to preserve, based on the given prune rate. The algorithm is randomized.
- **K-Neighbors (KN)**: selects k edges for each vertex or all edges if the vertex degree is less than k. The algorithm is randomized.
- **Rank Degree (RD)**: starts with random vertices (seeds) and preserves edges to the neighbors with higher degree. The newly introduces vertices are the new seeds. The process is repeated until prune rate is achieved. The algorithm is randomized.
- **Local Degree (LD)**: Similar to Rank Degree but deterministic. For each vertex selects edges to the $deg(u)^a$ top-ranked neighbors, where $a \in [0, 1]$ is a hyper-parameter that controls the prune rate.
- **Spanning Forest (SF)**: constructs a spanning forest of the given graph. The algorithm is randomized and does not provide control over the prune rate.

- **t-Spanner (SP-t)**: constructs a spanning tree in which all pairwise distances of vertices are at most t times larger than the original.
- **Forest Fire (FF)**: constructs a graph by adding one vertex at a time and and forming edges to certain subsets of the existing vertices. It also simulates the "burning" of some edges under some specified probability. The algorithm is randomized.
- **G-Spar (GS)**: Sorts all edges based on the Jaccard similarity of the incident vertices (the higher, the better). Preserves top-k edges, where k is defined based on the desired prune rate.
- **L-Spar (LS)**: is the local variation of G-spar, where d^c edges are kept from each vertex locally, based on their Jaccard similarity score.
- **Local Similarity (LSim)**: similar to L-spar, but using a normalized, ranked similarity measure.
- **Scan Similarity (SCAN)**: similar to G-spar, but using structural similarity (SCANSimilarity) instead of Jaccard.
- **Effective Resistance (ER)**: calculates the effective resistance for all edges. Afterwards, edges are selected with a probability proportional to their effective resistance.

To that end, for the shake of completeness we have to report that not all sparsification algorithms evaluated in the paper work the same on undirected, directed, weighted and unweighted graphs. As a matter of fact, some algorithms are not even applicable to directed or weighted graphs. In order to evaluate them, the authors construct a respective undirected or unweighted graph respectively in such cases. Furthermore, some sparsification algorithms provide guarantees that the sparse graph will be connected, while other algorithms do not. The interested reader can find a complete analysis of which algorithms are applicable to which cases in table 2 of the paper.

3.2 Graph metrics

Graph sparsification algorithms are not the only ones that exist in a large scale of variations; graph metrics as well. Each metric measures some specific aspect(s) of the graph properties. The authors select 16 well-known graph metrics to cover the whole spectrum of graph properties. The selected metrics are divided by the authors in the following categories:

- **Basic metrics**: degree distribution, Laplacian quadratic form
- **Distance metrics**: all pairs shortest path (APSP), diameter, vertex eccentricity
- **Centrality metrics**: betweeness, closeness, eigenvector, Katz
- **Clustering metrics**: number of communities, local clustering coefficient (LCC), global clustering coefficient (GCC), clustering F1 score
- **Application-level metrics**: PageRank, min-cut / max-flow, graph neural networks (GNN)

For the shake of completeness, again, we state that not all metrics apply to both undirected and directed, unweighted and weighted graphs. The authors present thorough analysis on the way they handle these metrics in special graph categories. The interested reader can find a comprehensive catalog in Table 1 of the paper.

3.3 Real-world graphs

The evaluation framework would not be complete without some graph instances to implement sparsification algorithms on. The authors selected 14 real-world graphs spanning various categories, with diverse characteristics, sizes, and densities, in a try to produce more robust and general results. Inside the set of selected graph instances there are both directed and undirected, weighted and unweighted graphs sourced from various resources. The interested reader can find a comprehensive catalog of the graph instances used by the authors in Table 3 of the paper.

4 Results of the conducted research

The authors provide thorough analysis on the performance of all the sparsification algorithms in every computed metric. They opt for presenting one graph instance per category as a plot, while reporting verbally the results found for the rest graph instances. In every computed graph metric, the authors present not only the sparsification algorithms that are performing well, but also the algorithms that perform poorly, sharing any additional interesting outcome. The artifact (GitHub repository) accompanying the paper provides a folder with all the generated line and bar plots, depicting the results. Several of them are incorporated and thoroughly analyzed by the authors in the paper, as well.

The authors' findings indicate that there is no single sparsification algorithm that excels in preserving all graph properties. The authors highlight the importance of selecting appropriate sparsification algorithms based on the downstream task. A summary of the finding grouped by sparsification algorithm is the following:

- **Random**: preserves relative (distribution-based or ranking-based) properties, for example, degree distribution, and top centrality rankings. It struggles to preserve absolute (valued-based) properties, for example, number of communities, clustering coefficient, and min-cut/max-flow.
- **K-Neighbor, Spanning Forest and t-Spanners**: preserves graph connectivity; keeps pair unreachable ratio and vertex isolated ratio low.
- **Rank degree and Local degree**: preserves graph connectivity and edges to high-degree vertices (hub vertices). Perform well on distance metrics (APSP, eccentricity, diameter) and centrality metrics.
- **Forest Fire**: simulates the evolution of graphs and does not strictly stick to the original graph. Empirically it does not excel at any metrics evaluated.
- **G-spar and SCAN**: Empirically perform well in preserving ClusterGCN accuracy.
- **L-spar and Local Similarity**: preserves the edge to similar vertices, thus preserves clustering similarity.
- **ER**: preserves the spectral properties of the graph, specifically the quadratic form of the graph Laplacian. It perform well in preserving min-cut/max-flow results.

References

[1] Yuhan Chen et al. “Demystifying Graph Sparsification Algorithms in Graph Properties Preservation”. In: *50th International Conference on Very Large Databases (VLDB 2024)*. ACM, 2024.

[2] *Cut (graph theory)*. (last accessed date: 11/15/2023). Feb. 2023. URL: https://en.wikipedia.org/wiki/Cut_(graph_theory).

[3] E. W. Dijkstra. “A Note on Two Problems in Connexion with Graphs”. In: *Numer. Math.* 1.1 (Dec. 1959), pp. 269–271. ISSN: 0029-599X. DOI: 10.1007/BF01386390. URL: https://doi.org/10.1007/BF01386390.

[4] Michael Hamann et al. *Structure-Preserving Sparsification Methods for Social Networks*. 2016. arXiv: 1601.00286 [cs.SI].

[5] David R. Karger. “Minimum Cuts in Near-Linear Time”. In: *J. ACM* 47.1 (Jan. 2000), pp. 46–76. ISSN: 0004-5411. DOI: 10.1145/331605.331608. URL: https://doi.org/10.1145/331605.331608.

[6] David R. Karger. “Random Sampling in Cut, Flow, and Network Design Problems”. In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*. STOC ’94. Montreal, Quebec, Canada: Association for Computing Machinery, 1994, pp. 648–657. ISBN: 0897916638. DOI: 10.1145/195058.195422. URL: https://doi.org/10.1145/195058.195422.

[7] David R. Karger. “Using Randomized Sparsification to Approximate Minimum Cuts”. In: *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’94. Arlington, Virginia, USA: Society for Industrial and Applied Mathematics, 1994, pp. 424–432. ISBN: 0898713293.

[8] Rasmus Kyng and Sushant Sachdeva. “Approximate Gaussian Elimination for Laplacians - Fast, Sparse, and Simple”. In: *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. 2016, pp. 573–582. DOI: 10.1109/FOCS.2016.68.

[9] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. “Graph evolution: Densification and shrinking diameters”. In: *ACM Trans. Knowl. Discov. Data* 1 (2006), p. 2.

[10] Allan H. Murphy. “The Finley Affair: A Signal Event in the History of Forecast Verification”. In: *Weather and Forecasting* 11.1 (1996), pp. 3–20. DOI: https://doi.org/10.1175/1520-0434(1996)011<0003:TFAASE>2.0.CO;2. URL: https://journals.ametsoc.org/view/journals/wefo/11/1/1520-0434_1996_011_0003_tfaase_2_0_co_2.xml.

[11] Yu Rong et al. *DropEdge: Towards Deep Graph Convolutional Networks on Node Classification*. 2019. DOI: 10.48550/ARXIV.1907.10903. URL: https://arxiv.org/abs/1907.10903.

[12] Veeranjaneyulu Sadhanala, Yu-Xiang Wang, and Ryan J. Tibshirani. “Graph Sparsification Approaches for Laplacian Smoothing”. In: *International Conference on Artificial Intelligence and Statistics*. 2016.

[13] Venu Satuluri, Srinivasan Parthasarathy, and Yiye Ruan. “Local Graph Sparsification for Scalable Clustering”. In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’11. Athens, Greece: Association for Computing Machinery, 2011, pp. 721–732. ISBN: 9781450306614. DOI: 10.1145/1989323.1989399. URL: https://doi.org/10.1145/1989323.1989399.

[14] Daniel Spielman. *Laplacians.jl*. https://github.com/danspielman/Laplacians.jl. 2023.

[15] Daniel A. Spielman and Nikhil Srivastava. “Graph Sparsification by Effective Resistances”. In: *SIAM Journal on Computing* 40.6 (2011), pp. 1913–1926. DOI: 10.1137/080734029. eprint: https://doi.org/10.1137/080734029. URL: https://doi.org/10.1137/080734029.

[16] Daniel A. Spielman and Shang-Hua Teng. “A Local Clustering Algorithm for Massive Graphs and Its Application to Nearly Linear Time Graph Partitioning”. In: *SIAM Journal on Computing* 42.1 (2013), pp. 1–26. DOI: 10.1137/080744888. eprint: https://doi.org/10.1137/080744888. URL: https://doi.org/10.1137/080744888.

[17] Daniel A. Spielman and Shang-Hua Teng. "Nearly Linear Time Algorithms for Preconditioning and Solving Symmetric, Diagonally Dominant Linear Systems". In: *SIAM Journal on Matrix Analysis and Applications* 35.3 (2014), pp. 835–885. DOI: 10.1137/090771430. eprint: https://doi.org/10.1137/090771430. URL: https://doi.org/10.1137/090771430.

[18] Elli Voudigari et al. "Rank degree: An efficient algorithm for graph sampling". In: *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 2016, pp. 120–129. DOI: 10.1109/ASONAM.2016.7752223.

[19] R. Wickman, X. Zhang, and W. Li. "A Generic Graph Sparsification Framework using Deep Reinforcement Learning". In: *2022 IEEE International Conference on Data Mining (ICDM)*. Los Alamitos, CA, USA: IEEE Computer Society, Dec. 2022, pp. 1221–1226. DOI: 10.1109/ICDM54844.2022.00158. URL: https://doi.ieeecomputersociety.org/10.1109/ICDM54844.2022.00158.

[20] Xiaowei Xu et al. "SCAN: A Structural Clustering Algorithm for Networks". In: *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '07. San Jose, California, USA: Association for Computing Machinery, 2007, pp. 824–833. ISBN: 9781595936097. DOI: 10.1145/1281192.1281280. URL: https://doi.org/10.1145/1281192.1281280.

[21] Cheng Zheng et al. "Robust Graph Representation Learning via Neural Sparsification". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 11458–11468. URL: https://proceedings.mlr.press/v119/zheng20d.html.