Jurnal Kejuruteraan 36(2) 2024: 525–532
https://doi.org/10.17576/jkukm-2024-36(2)-14

Random Sampling Method of Large-Scale Graph Data Classification

Rashed Mustafa[a*], Mohammad Sultan Mahmud[b] & Mahir Shadid[c]

[a]*Department of Computer Science and Engineering, University of Chittagong, Chittagong 4331, Bangladesh*

[b]*College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China*

[c]*Department of Computer Science and Engineering, International Islamic University Chittagong, Chittagong, Bangladesh*

[*]*Corresponding author: rashed.m@cu.ac.bd*

Received 9 June 2023, Received in revised form 19 September 2023
Accepted 19 October 2023, Available online 30 March 2024

ABSTRACT

Graph data appears in broad real-world applications in modelling complex objects in big data. Effective analysis of graph data provides a deeper understanding of the data in data mining tasks, including classification, clustering, prediction, and recommendation systems. Mining a large number of graphs becomes a challenging task because state-of-the-art methods are not scalable due to the memory limit. To address this issue, we propose a novel approximate random sampling method for large-scale graph data classification. In this approach, we applied a representation method to encode each graph as a record of a vector string and a set of graphs as a set of N records in a file. Then, we partition the set of records into disjoint subsets of data blocks, making each data block a random sample of the data file. After that, we randomly select a subset of data blocks, each being a random sample of the graph dataset, and compute the different graph property distributions. Since the data blocks in this model are much smaller than the entire data set, it is more efficient to analyze them on a standalone small machine, and multiple data blocks can be analyzed on multiple nodes of the cluster in parallel. Finally, we classified the graphs of data blocks using the SVM algorithm. In experimental evaluation, our proposed method outperformed state-of-the-art graph kernels on graph classification datasets in terms of accuracy.

Keywords: Graphs classification; Random sample partitioning; Approximate computing; Distributed and parallel computing

INTRODUCTION

Nowadays, graph data is ubiquitous in many real-world applications, including web and social network data, molecular and biological data, internet intrusion data, and call pattern data in telecommunications. According to graph characteristics, graph data analysis tasks can be divided into two areas: (i) a big graph, e.g., web data, social networks (Devi & Kasireddy 2019; Rodrigues et al. 2013), and (ii) a large number of comparatively small graphs, e.g., biological and chemical data (Borgwardt et al. 2005), cybersecurity (Noel et al. 2016), and telecommunication patterns (Fan et al. 2010). Many practical applications require the analysis of complex objects that can be represented as large numbers of small graphs. In such contexts, analysts are interested in the statistical distribution of graph topological properties for classification, clustering, and predictions (Riesen & Bunke 2008; Kriege et al. 2018).

Graph data in real-world applications represent a very diverse range of entities in large numbers. The major challenge is processing and analyzing such graphs because algorithms for graph analysis are required to construct a single graph in big size but with distinct node labels, e.g., web and social networking data. On the other hand, algorithms designed for large numbers of small graph objects need to take into account repetitions in node labels, e.g., biological and chemical data. Divide and conquer is a common strategy for large-scale data analysis in distributed and parallel computing frameworks. Thus, a scalable and distributed architecture is often necessary for

processing large amounts of graph data. In consequence, it is indeed challenging to handle distributed complex graph data. Last several years, we have witnessed a growing interest in distributed graph computing tools, including Mizan (Khayyat et al. 2013), GPS (Salihoglu & Widom 2013), GraphLab (Low et al. 2012), PowerGraph (Gonzalez et al. 2012), and Pregel (Malewicz et al. 2010).

For large amounts of graph data analysis, we require distributed representations of the graph data. Also, it is inefficient to analyze the entire dataset on a single machine. Hence, we need to take samples of graphs to estimate an approximate distribution of the whole dataset. In fact, a key to analyzing large-scale graph data is to use a subset of the data to estimate the entire dataset. Recently, the random sample partitioning (RSP) (Salloum et al. 2019) distributed data model was proposed to enable random sampling from big datasets. In the RSP model, the statistical properties of the entire dataset are maintained in its data subsets, i.e., the RSP data blocks. These RSP data blocks can be utilized to compute the statistical properties of the big dataset and facilitate approximate analysis. In the RSP model, the large-scale data analysis becomes an analysis of a subset of RSP data blocks; therefore, it is scalable to big data. Several approaches are proposed using the RSP data model, including unsupervised learning (Mahmud et al. 2023a; 2023b).

In this paper, we introduce an innovative distributed graph computing framework that can synthesize the approximate result of a big set of small graphs. We adopted the RSP data model to partition the graph dataset into disjoint subsets of the same-sized data blocks. Following the definition of the random sample partition data model in (Salloum et al. 2019), we call the new set of RSP graph data blocks as the graph random sample partition data model, or GRSP data model for short. From the GRSP data model, we can randomly select GRSP data blocks as random samples of the entire graph data and conduct graph mining analyses efficiently and effectively, as well as in parallel. The approximate results are used as estimates of the results from the entire graph dataset. In this framework, using a small part of the entire dataset can produce a result that is equivalent to results computed from the entire dataset. Finally, the selected GRSP data blocks are classified using the SVM algorithm.

Specifically, we extended the general RSP data model to the GRSP data model for graph data analysis. We are interested in approximate graph properties, which are based on measuring the distribution of properties of graphs in subsets. The experimental results have shown that the distribution of graph properties and sample statistics from GRSP samples is equivalent to the whole dataset. Moreover, randomly selecting a few GRSP data blocks from a big dataset reduces execution time significantly and is efficient to analyze.

RELATED WORK

In recent years, graph representation learning (Wu et al. 2020; Xie et al. 2022; Yi et al. 2022) has become increasingly popular since graph data are everywhere in real-world applications. Several methods have been studied to explore graph processing tasks, including visualization (Holten & Wijk 2009), evaluating community structure in networks (Trinh & Vuongthi 2022), classification (Zeng & Xie 2021), clustering (Liu et al. 2023), and query languages (Pienta et al. 2016). Graph analysis is essential, but most existing techniques are computationally expensive and space-intensive (Ma et al. 2016; Heidari et al. 2018). Many studies have been devoted to graph data analysis efficiently in distributed frameworks, e.g., GraphFrames (Dave et al. 2016), GraphX (Gonzalez & Xin 2014), GraphLab (Low et al. 2012), G-store (Kumar & Huang 2016), and GraphChi (Kyrola et al. 2012). In addition to these approaches, graph embedding techniques offer an effective and efficient solution to graph data analysis problems. Precisely, graph embedding transforms a graph into a low-dimensional space, and the graph information is well-maintained.

Graph embedding output can be classified into edge embedding, node embedding, hybrid embedding, and entire-graph embedding (Cai et al. 2017). The entire graph embedding is commonly applied to small graphs, for example, molecules and proteins. In such a situation, a graph is characterized as a vector, and two similar graphs are embedded closer together. The entire graph embedding gives an efficient and straightforward outcome for computing graph similarities (Mousavi et al. 2017). But an entire graph embedding is time-consuming compared to others because it requires capturing the properties of the entire graph. The difficulty of entire-graph embedding is selecting an optimal choice between the efficiency and expressive learning power of the embedding algorithm. Recent developments of graph embedding methods in the biomedical domain is highlighted by Wu et al. (2023).

To deal with graph data, graph contrastive learning and multi-view graph convolutional networks have become active research problems in recent years (Zhu et al. 2021; Chen et al. 2023). An effective way of transforming big graph data is sampling, which involves selecting nodes or edges to construct a subgraph that represents the original unfiltered graph. Graph sampling is fundamental research in many areas. Graph sampling methods can be categorized into two types: non-graph structure-aware and graph structure-aware. Non-graph structure-aware sampling does not consider the structure of the graph, such as node sampling and edge sampling. On the other hand, structure-aware sampling considers the structure of the graph and will select nodes with a high degree, e.g., random walk sampling (Ribeiro & Towsley 2010), fire forest sampling

(Leskovec & Faloutsos 2006), page-rank sampling, and induced edge sampling (Kim & Rinaldo 2017).

Matching the properties of the sample graphs with the original graph has been studied by Leskovec & Faloutsos (2006), Ahmed et al. (2010) and Ahmed et al. (2013). Node sampling (NS), edge sampling (ES), and traversal-based sampling (TBS) are the most commonly studied and used sampling techniques. Node sampling (NS) selects nodes based on uniform sampling to represent the original degree distribution accurately. The sample degrees in the NS-induced graph are lower than the original degrees since the nodes are sampled independently. Consequently, due to projected degrees of less than one, numerous low-degree nodes are separated from the sample. Since Edge Sampling (ES) includes the selected edges only in the sample graph, its high-degree nodes are more frequent than NS, but the sampled degrees of those nodes are lower. However, these algorithms do not show a good similarity to the main graph.

Forest Fire Sampling (FFS) (Leskovec & Faloutsos 2006) is a graph generation model to perform graph sampling. Ahmed et al. (2010) considered FFS to show network distributions with a sample size of 20%. The result in degree distribution showed that FFS captured low-degree nodes with a large fraction but failed to capture high-degree nodes. Also, for most of the datasets, FFS failed to find the core graph structures. After constructing the actual distribution, the max-core in the sampled graphs is compared with its real counterparts for a sample size of 20%. Moreover, it is observed that FFS's max-core number is consistently smaller than the real max-core number by an order of magnitude. It points out that the local density of the sampled subgraph structures is not well maintained in the FFS. Thus, it is evidence that the FFS method failed to approximate the original graph eigenvalues.

METHODOLOGY

In this section, we propose a new random sample graph classification method (abbreviated as RSGC) for analyzing big graph datasets containing a large number of graph objects. The comprehensiveness of analysis refers to the analytical tasks that require multiple steps to complete, and some steps perform complex analytical operations. The divide-and-conquer strategy is adopted to divide a big set of graph objects into subsets, each of which is a random sample of the whole dataset; i.e., the topological property distributions of the subsets are similar to the distributions of the whole set of graph objects. Given the subsets of graph objects, which we call GRSP data blocks, we can randomly select a few GRSP data blocks and use them to compute approximate results as estimates of the entire graph dataset. This methodology can be implemented on computing clusters to efficiently and effectively analyze big graph data in parallel and distributed fashions.

The pseudocode for RSGC is given in Algorithm 1. The main steps and technologies used in this methodology are presented in detail as follows:

Algorithm 1: RSGC algorithm.
Input:
Dataset, $\boldsymbol{G}$
Sample size, n
procedure GRSP(G,n)
$G \leftarrow$ AWE ($\boldsymbol{G}$) // Apply encoding on dataset $\boldsymbol{G}$ to represent graph as vector
$\{G_1, G_2, \ldots, G_L\} \leftarrow$ GRSP (G, n) // Apply GRSP data model to create GRSP data blocks
Output: a set of L graph data blocks of **G**
end procedure
procedure LOCALGP(G_i)
for each G_i **do**
GP_i = *PropertiesMeasure* (G_i) // Measure locally graph properties in each GRSP data blocks
end for
Output: a set of L sets of local graph properties
end procedure
procedure CLASSIFICATION(G)
$CA \leftarrow$ SVM (G_i) // For all GRSP data block obtain classification results
Output: Classification accuracy of G
end procedure

1. *Graph object encoding*: Given a graph dataset ***G***. We first encode it into a vector representation of graphs using the recently discovered anonymous walk embedding (AWE) (Ivanov & Burnaev 2018). ***G*** is preprocessed such that each record represents one graph.
2. *Generating GRSP data model*: In this step, the encoded vector dataset G is transformed into GRSP data blocks. Random sample partitioning (RSP) is the basis of this work. We convert a graph dataset into a set of disjoint graph random samples using the GRSP data model. Assume that $G = \{g_1, g_2, ..., g_N\}$ be a graph dataset of N graphs, and G cannot be analyzed efficiently on traditional computing. To avoid the computational burden, the GRSP process is applied into G and we prepared ready-to-use GRSP data blocks $\{G_1, G_2, ..., G_L\}$, i.e., random samples.
3. *Topological property metrics*: In this step, topological graph property metrics algorithms are employed to estimate the local properties of each GRSP blocks $\{G_1, G_2, ..., G_L\}$. Finally, L sets of local graph properties are obtained.
4. *Graph classification*: Graph classification is the process of predicting a class label for the graphs. The sets of GRSP data blocks in AWE vector-embedded graphs are classified with a simple SVM classifier. The goal is to achieve a faster solution to the SVM problem without a significant loss in prediction error.

EXPERIMENT

In this section, we evaluate the performance of the proposed RSGC method with both synthetic and real-world datasets. Our experiment is two-fold to test the proposed methodology. First, the graph properties metric was applied to the entire dataset to retrieve global distributions. These collections are used as a benchmark to testify to the collection graph's property distribution. Second, we focus on graph classification.

DATASET

We employed the following two synthetic and two real-world datasets in our experiments. Table 1 shows the characteristics of the datasets.

1. Synthetic datasets: Two synthetic graph datasets are generated with different numbers of nodes and edges based on random graphs whose nodes are endowed with the normal distribution N(0;1). Each synthetic dataset belongs to two classes with different attributed graphs that rewire edges and permute node attributes randomly.

2. Proteins (Dobson & Doig 2003): This dataset describes proteins, enzymes or non-enzymes. As SSEs (secondary structure elements), proteins are symbolized as graphs with nodes that are connected with their neighbours' amino acid sequences.

3. Airways (Petersen et al. 2011): This dataset classifies healthy individuals and patients suffering from COPD (chronic obstructive pulmonary disease). It is extracted from CT scans of lung cancer screens, where each node denotes an airway branch attributed to its length, and edges denote adjacencies between airway branches.

TABLE 2. Dataset description in a glance.

Dataset	Dataset size	Number of nodes	Number of edges	Class distribution
Synthetic 1	10,000	100	200	5000/5000
Synthetic 2	10,000	200	200	5000/5000
Proteins	1113	40	73	663/450
Airways	1966	221	220	980/986

EXPERIMENT SETUP

We generated GRSP data blocks with data points each. The train-test test is used for each data block to calculate the accuracy of the model. We set a training data subset of 50% and a test data subset of 50% for each GRSP data block in experiments. More specifically, each data block is divided into two folds. One acts as training, while the other is for testing. We repeated it for all the data blocks, and the average score of all the data blocks is the accuracy of the model. For the other methods compared, we used a single training data subset (50%) and a test data subset (50%) of the entire dataset and a similar number of random runs. For each applied method, the average classification accuracy (mean) and standard deviation (std) are tabulated in Table 3. For classification, we employed the widely used

SVM (support vector machine) classifier (Cortes & Vapnik 1995).

The configuration of the used machine was an Intel (R) Core i7-4790 x64-based processor with a CPU speed of 3.6 GHz, 16.0 GB of RAM, and Windows 10 Pro 64-bit. The experiments were conducted in the Python framework.

EVALUATED METHODS

We compare the proposed RSGC algorithm with the state-of-the-art graph kernel-based methods listed as follows:

1. Weisfeiler-Lehman Kernel (WLK) (Shervashidze et al. 2011): It is an efficient graph kernel scheme based on the Weisfeiler-Lehman isomorphism test on graphs. Its runtime is linearly related to the Weisfeiler-Lehman graph sequence length and the number of edges on the graphs.
2. Shortest Path Kernel (SPK) (Borgwardt & Kriegel 2005): It is a graph kernel based on the shortest paths and retains expressivity and positive definiteness. SPK is calculable in polynomial time.
3. Connected Subgraph Matching Kernel (CSMK) (Kriege & Mutzel 2012): It is a subgraph-matching-based graph kernel. CSMK uses a relation between common subgraphs of two graphs and cliques in their product graph to compute the kernel.
4. GraphHopper (Feragen et al. 2013): It is a convolution kernel based on sub-path similarity counting that quadratically scales with the number of nodes in the dataset.

EXPERIMENTAL RESULT ANALYSIS

In the experiment, we first explored the graph property distribution and statistics of GRSP data blocks. We summarized the average and standard deviation results of ten random trials in Table 2. From the summary statistics, we can observe that there is no significant difference among GRSP data blocks.

TABLE 2. Statistics (mean ± std) of GRSP data blocks.

Dataset	Number of nodes	Number of edges
Synthetic 1	99.7 ± 1.8	198.4 ± 2.6
Synthetic 2	98.2 ± 1.3	197.0 ± 3.1
Proteins	38.4 ± 3.2	69.8 ± 2.7
Airways	216.6 ± 4.1	218 ± 2.9

GRSP data block size, n=200

To demonstrate the quality of the proposed RSGC algorithm, we reported the average classification accuracy on two synthetic and two bioinformatics datasets, as presented in Table 3. From Table 3, we can see a comparison between the state-of-the-art methods and RSGC. The highest accuracy was achieved by the RSGC method, and GH obtained the second-best accuracy on the three datasets. Also, RSGC is consistent with the average rank obtained by comparing various existing methods on four experimental datasets, which further exemplifies the advantage of our proposed algorithm.

Figure 1 presents an empirical evaluation to compare the computation time across different methods for four datasets. The average runtime of ten trails is shown. It is clear to see that the execution time consumption of the proposed RSGC algorithms is much lower than that of WLK, SPK, CSMK, and GH on all datasets.

TABLE 3. Classification accuracy (mean ± std of 10 trails).

Dataset	WLK	SPK	CSMK	GH	RSGC
Synthetic 1	48.3 ±1.9 (4)	82.5 ±2.8 (3)	Out of memory	84.3 ±0.9 (2)	**88.9** ±1.6 (1)
Synthetic 2	45.6 ±2.3 (4)	78.4 ±3.4 (3)	Out of memory	81.6 ±1.2 (2)	**86.4** ±1.8 (1)
Proteins	76.8 ±0.8 (2)	75.6 ±1.1 (3)	59.4 ±1.4 (5)	75.1 ±0.6 (4)	**81.2** ±2.3 (1)
Airways	63.2 ±1.1 (3)	61.1 ±1.3 (4)	52.7 ±0.9 (5)	67.6 ±0.8 (2)	**69.6** ±2.5 (1)
Avg. rank	3.2	3.2	5	2.5	1

Best result in bold and parenthesis () shows rank of the method.

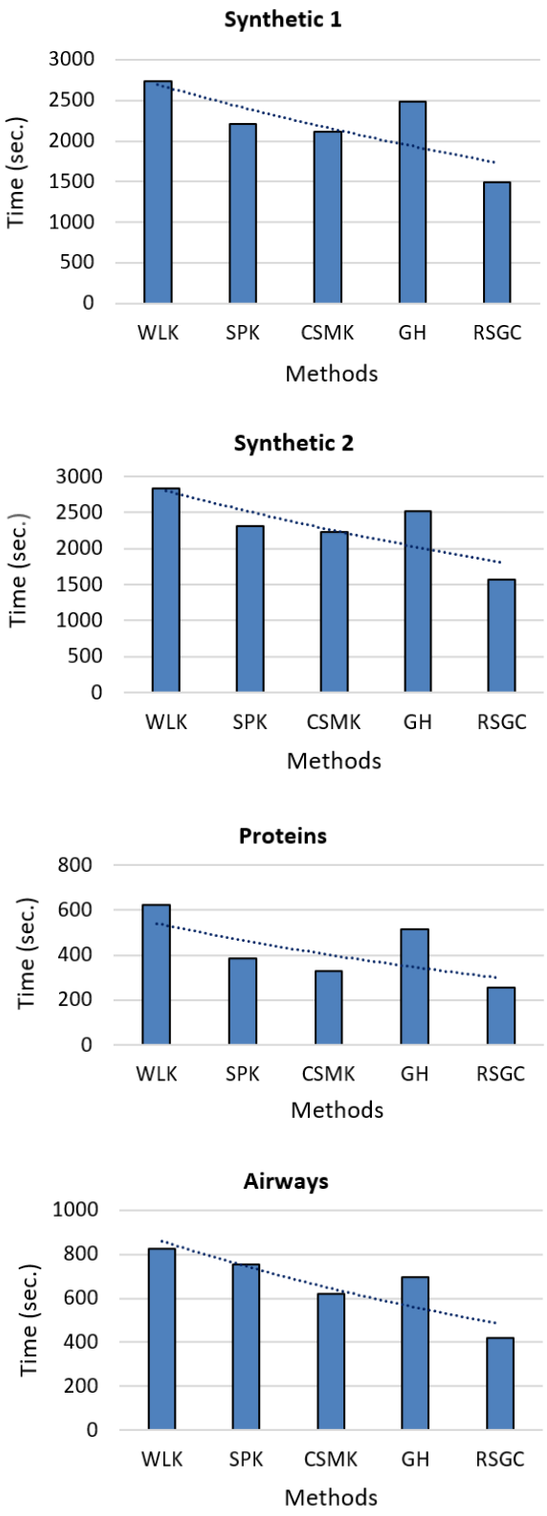

FIGURE 1. Runtime comparison of the methods.

CONCLUSION

In this paper, we present a new methodology, RSGC, for graph data classification based on a GRSP data model. First, we adopt the anonymous walk embedding to encode the graph dataset into vector representations. Then, the GRSP data model is applied to generate multiple graph random samples, along with SVM as a classification method. Our experiments show that embedding graph vectors in GRSP data blocks with a simple SVM classifier can be very beneficial and achieve better classification accuracy compared to state-of-the-art methods and graph kernels.

An important sampling and partitioning approach to solving the graph dataset problem in the context of large numbers of graph sets is presented and evaluated. The advantage of the proposed method is that it can significantly reduce the size of the data to be mine with better accuracy. We observe that there are two drawbacks to the proposed method: 1) In the node encoding process, existing methods usually overlook the learning of structural information. Consequently, the discriminative capability of representations is limited. (2) The proposed algorithm does not perform well in a big graph, e.g., a social networking or web graph. For further work, we are going to expand the proposed methodology on a cluster computing framework and implement the same algorithm on MapReduce to compare experiments with real big graph data datasets. Also, a theoretical analysis will be conducted to prove the obtained results from a statistical perspective.

ACKNOWLEDGEMENT

Support for this research was provided by the Chittagong University, Bangladesh. Research cell, Annual Research Grant 2021-2022, grant no. 599/2021-22/3rd call/21/2022.

DECLARATION OF COMPETING INTEREST

None

REFERENCES

Ahmed, N. K., Berchmans, F., Neville, J. & Kompella, R. 2010. Time-based sampling of social network activity graphs. Proceedings of the Eighth Workshop on Mining and Learning with Graphs. 1–9.

Ahmed, N. K., Neville, J. & Kompella, R. 2013. Network sampling: From static to streaming graphs. *ACM Transactions on Knowledge Discovery from Data* 8(2): 1–56.

Borgwardt, K. M. & Kriegel H. P. 2005. Shortest-path kernels on graphs. Proceedings of the 5th IEEE International Conference on Data Mining. 74–81.

Borgwardt, K. M., Ong, C. S., Schonauer, S., Vishwanathan, S. V. N., Smola, A. J. & Kriegel, H. P. 2005. Protein function prediction via graph kernels. B*ioinformatics* 21(1): 47–56.

Cai, H., Zheng, V. W. & Chang, K. C. C. 2017. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* 30: 1616–1637.

Chen, Z., Fu, L., Xiao, S., Wang, S., Plant, C. & Guo, W. 2023. Multi-view graph convolutional networks with differentiable node selection. *ACM Transactions on Knowledge Discovery from Data* 18(1): 1–21.

Cortes, C. & Vapnik, V. 1995. Support-vector networks. *Machine Learning* 20, 273–297.

Dave, A., Jindal, A., Li, L. E., Xin, R., Gonzalez, J. & Zaharia, M. 2016. GraphFrames: An integrated API for mixing graph and relational queries. Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems, 1–8.

Devi, N. M & Kasireddy, S. R. 2019. Graph analysis and visualization of social network big data. *Social Network Forensics, Cyber Security, and Machine Learning* 93–104.

Dobson, P. D. & Doig, A. J. 2003. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology* 330(4): 771–783.

Fan, W., Li, J., Ma, S., Tang, N., Wu, Y. & Wu, Y. 2010. Graph pattern matching: from intractable to polynomial time. Proceedings of the VLDB Endowment 3(1–2): 264–275.

Feragen, A., Kasenburg, N. Petersen, J. Bruijne, M. D. & Borgwardt, K. 2013. Scalable kernels for graphs with continuous attributes. Proceedings of the 26th International Conference on Neural Information Processing Systems 1: 216–224.

Gonzalez, J. E., Low, Y., Gu, H., Bickson, D. & Guestrin C. 2012. PowerGraph: Distributed graph-parallel computation on natural graphs. Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation, 17–30.

Gonzalez, J. E., Xin, R. S., Dave, A., Crankshaw, D., Franklin, M. J. & Stoica, I. 2014. GraphX: Graph processing in a distributed dataflow framework. Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation, 599–613.

Heidari, S., Simmhan, Y. L., Calheiros, R. N. & Buyya, R. 2018. Scalable graph processing frameworks: A taxonomy and open challenges. *ACM Computing Surveys* 51(3): 1–53.

Holten, D. & Wijk J. J. V. 2009. A user study on visualizing directed edges in graphs. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2299–2308.

Ivanov, S. & Burnaev, E. 2018. Anonymous walk embeddings. Proceedings of the 35th International Conference on Machine Learning 80: 2186–2195.

Khayyat, Z., Awara, K., Alonazi, A., Jamjoom, H., Williams, D. & Kalnis, P. 2013. Mizan: a system for dynamic load balancing in large-scale graph processing. Proceedings of the 8th ACM European Conference on Computer Systems 169–182.

Kim, N. & Rinaldo, A. 2017. Edge-induced sampling from Graphons. *Arxiv,* 1–12.

Kriege N. M., Fey, M., Fisseler, D., Mutzel, P. & Weichert, F. 2018. Recognizing cuneiform signs using graph based methods. Proceedings of Machine Learning Research 88: 31–44.

Kriege, N. & Mutzel, P. 2012. Subgraph matching kernels for attributed graphs. Proceedings of the 29th International Conference on Machine Learning, 291–298.

Kumar, P. & Huang, H. H. 2016. G-store: High-performance graph store for trillion-edge processing. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 830–841.

Kyrola, A., Blelloch, G. & Guestrin, C. 2012. GraphChi: Large-scale graph computation on just a PC. *10th USENIX Symposium on Operating Systems Design and Implementation* 31-46.

Leskovec, J. & Faloutsos, C. 2006. Sampling from large graphs. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* 631-636.

Liu, B., Che, Z., Zhong, H. & Xiao, Y, 2023. A ranking based multi-view method for positive and unlabeled graph classification. *IEEE Transactions on Knowledge & Data Engineering* 35(3): 2220–2230.

Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A. & Hellerstein, J. M. 2012. Distributed Graphlab: A framework for machine learning and data mining in the cloud. Proceedings of the VLDB Endowment 5(8): 716–727.

Ma, S., Li, J., Hu, C., Lin, X. & Huai, J. 2016. Big graph search: challenges and techniques. *Frontiers of Computer Science* 10(3): 387–398.

Mahmud, M. S., Huang, J. Z., Ruby, R. & Wu, K. 2023a. An ensemble method for estimating the number of clusters in a big data set using multiple random samples. *Joural of Big Data* 10(40): 1–33.

Mahmud, M. S., Huang, J. Z., Ruby, R., Ngueilbaye, A. & Wu, K. 2023b. Approximate clustering ensemble method for big data. *IEEE Transactions on Big Data* 9(4): 1142–1155.

Malewicz, G., Austern, M. H., Bik, A. J., Dehnert, J. C., Horn, I., Leiser, N. & Czajkowski, G. 2010. Pregel: A system for large-scale graph processing. Proceedings of the ACM SIGMOD International Conference on Management of Data, 135–146.

Mousavi, S. F., Safayani, M., Mirzaei, A. & Bahonar, H. 2017. Hierarchical graph embedding in vector space by graph pyramid. *Pattern Recognition* 61: 245–254.

Noel, S., Harley, E., Tam, K., Limiero, M. & Share, M. 2016. CyGraph: Graph-based analytics and visualization for cybersecurity. *Handbook of Statistics* 35: 117–167.

Petersen, J., Nielsen, M., Lo, P., Saghir, Z., Dirksen, A. & Bruijne, M. 2011. Optimal graph based segmentation using flow lines with application to airway wall segmentation. *Information Processing in Medical Imaging* 49–60.

Pienta, R., Tamersoy, A., Endert, A., Navathe, S., Tong, H. & Chau, D. H. 2016. VISAGE: Interactive visual graph querying. *Proceedings of the International Working Conference on Advanced Visual Interfaces* 272-279.

Ribeiro, B. & Towsley, D. 2010. Estimating and sampling graphs with multidimensional random walks. *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement* 390-403.

Riesen K. & Bunke, H. 2008. IAM graph database repository for graph based pattern recognition and machine learning. *Structural, Syntactic, and Statistical Pattern Recognition* 287–297.

Rodrigues, J. F., Tong, H., Pan, J. Y., Traina, A. J. M., Traina, C. & Faloutsos, C. 2013. Large graph analysis in the GMine system. *IEEE Transactions on Knowledge and Data Engineering* 25(1): 106–118.

Salihoglu, S. & Widom, J. 2013. GPS: a graph processing system. Proceedings of the 25th International Conference on Scientific and Statistical Database Management 1–12.

Salloum, S., Huang, J. Z. & He, Y. 2019. Random sample partition: A distributed data model for big data analysis. *IEEE Transactions on Industrial Informatics* 15(11): 5846–5854.

Shervashidze, N., Schweitzer, P., Leeuwen, E. J., Mehlhorn, K. & Borgwardt, K. M. 2011. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research* 12: 2539–2561.

Trinh, T. & Vuongthi, N. 2022. A predictive paradigm for event popularity in event-based social networks. *IEEE Access* 10(1): 125421–125434.

Wu, Y., Chen, Y., Yin, Z., Ding, W. & King, I., 2023. A survey on graph embedding techniques for biomedical data: Methods and applications. *Information Fusion* 100(1): 1–24.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. & Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32(1): 4–24.

Xie, Y., Xu, Z., Zhang, J., Wang, Z. & Ji, S. 2022. Self-supervised learning of graph neural networks: A unified review. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45(2): 2412 - 2429.

Yi, H. C., You, Z. H., Huang, D. S. & Kwoh, C. K. 2022. Graph representation learning in bioinformatics: trends, methods and applications. *Briefings in Bioinformatics* 23(1): 1-24.

Zeng, J. & Xie, P. 2021. Contrastive self-supervised learning for graph classification. *Proceedings of the AAAI Conference on Artificial Intelligence* 35 (1): 10824–10832.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S. & Wang, L. 2021. Graph contrastive learning with adaptive augmentation. Proceedings of the Web Conference 2069–2080.