Original Article

Graphlab Geometric Multiway Recurrent Pattern Extraction Algorithm For Recurrent Pattern Generation In Distributed Graph System

Sadhana Priyadarshini[1], Sireesha Rodda[2]

[1,2] Department of Computer Science and Engineering, GIT, GITAM, Gandhi Nagar, Rushikonda, Visakhapatnam, Andhra Pradesh, India.
Email: [1] sadhana.priyadarshini00@gmail.com, [2] sireesharodda@gmail.com

*Corresponding Author: Sadhana Priyadarshini
Department of Computer Science and Engineering, GIT, GITAM, Gandhi Nagar, Rushikonda, Visakhapatnam-530045, Andhra Pradesh, India.
DOI: 10.47750/pnr.2022.13.S10.551

Abstract

A recurrent pattern is required for fact-finding analysis on graph databases and it is a subgraph that presents frequently in the graph database. Recurrent pattern extraction is a part of Data Mining where more research is going on to fulfill the current era requirements as the data size grows at a faster rate. The researcher has to focus on developing a methodology for candidate generation (with duplicate elimination) and frequency count that is mathematically efficient and policy-wise effective. In a distributed system, it is very difficult to calculate the global frequency of any pattern during the calculation phase. In a top-level parallel framework, like Pregel, MapReduce doesn't support it. In this paper, we propose a GraphLab Geometric Multiway Recurrent Pattern Extraction (GGMRPE) algorithm that solves these issues present in existing methods. A cache consistency process arises in GraphLab, which is to be solved by the versioning method. Load balancing is done by a homogeneous partition and an equal number of ghosts. We used canonical ordering of locks to avoid the deadlock and implement the Lamport-Chandy algorithm to overcome the fault tolerance. We performed an experimental investigation with four real graph datasets with different criteria and found drastic improvement (i.e. 75% improved) with various algorithms such as PageRank, Connected Component, Triangle Counting, Collaborative Filtering, and Belief Propagation.

Index terms: recurrent pattern, graph database, support, graph mining algorithms.

INTRODUCTION

Recurrent Pattern Mining, a component of Data Mining plays an incredible role in data analysis despite the complexity that arises due to faster growth in the input graph. Several application fields include such as program flow, social networks, chemical compounds, bioinformatics, etc. Recurrent pattern mining is broadly classified into two phases: the first is the generation of candidate patterns, and the second is the frequency count of the pattern. The R-Gaston algorithm implemented frequent support calculation of existing ones and aggregated frequent support for incoming frequent patterns in a distributed graph database. It is mainly flexible to use in transaction datasets, as revised by the existing Gaston algorithm [1].

Generally, a recurrent pattern algorithm is based on a graphical structure where the significance of entities and their association cannot be taken care of. Hence, weighted graphs are the best option for such areas as transportation networks, mobile-communication networks, and social networks. The Weighted-Frequent-Subgraph mining using the Map Reduce framework (WFSM-MR) was developed to extract frequent patterns from edge-weighted graph datasets. The runtime is increased by 3.5 and 7 percent in defect localization and explorative extraction respectively [2 . . . 4]. In the case of a single labeled graph, discovering a globally allocated recurrent pattern (G-Pattern) is done by G-Measures. The G-Miner method is proposed to extract the G-Pattern [5]. In the current era, the complex large graph collaboration in general applications like Robotics, and Facebook, lead to the development of dynamic graph mining. Distributed Frequent Subgraph Mining (DFSMS) implements SPARK to extract recurrent patterns for maintaining a set of patterns between frequent and infrequent patterns to eliminate explore space [6].

In the first section, we explore the main need of the proposed work with overall ideas. The next section is elaborate on the work already done associated with the existing problem. In the third section, we explain our proposed algorithm in detail. In the fourth section, we explain all methods used in GraphLab distributed system. In the fifth section, we make a comparative analysis of our proposed algorithm. Then next section, we come to the end of our research and the future core of the action.

RELATED WORK

Many algorithms and tools are available for extracting frequent patterns for both single large graph databases and small graph sets. This section explores the related work done in distributed and single-graph datasets. Schreiber F et.al proposed a MAVisto application that deals with both directed and undirected graphs in biological analysis. The network patterns of local interconnection are taken. The Frequent Pattern Finder (FPF) algorithm is used to detect motifs that are used in MAVisto. It is a Java Web application that is capable to find all motifs of a fixed number of edges and nodes size [8].

Kashani ZR et al. dev developed the Kavosh method to generate motifs with reduced CPU time and space requirements. It performs iteration to generate all subgraphs with a specific node and then removes duplicates. It well suits both undirected unlabelled and directed graphs to extract any size of nodes [9]. Inokuchi et al. developed the Apriopri graph mining(AGM) algorithm to extract subgraphs in a breadth-first search(BFS) way that represents the graph in the adjacent matrix. At each iteration, two candidate subgraphs are joined if they have the same size as the last iteration [10].

Kuramochi M et al. proposed the Frequent subgraph generation algorithm to produce candidate subgraphs by merging new links at each superstar using linked-based candidate production [11,12]. Bismita S. Jena et al designed Object Oriented Frequent Subgraph Mining (OO_FSM) that not only generates candidate subgraphs but also prunes the unwanted links from the graph database. The frequency count of any links is achieved by the multiway and multivalued map on the whole dataset based on user given threshold value [13].

Peng Jiang et al. proposed a novel two-node extraction method to generate a recurrent pattern that over issues related to huge space necessity and limited execution time. They implement the raised isomorphism checks. The subgraph explosion problem reduces by the subgraph sampling methodology [14]. M.Sailaja et al, conducted a comparative study on different FSM in a distributed system to find efficient algorithms to reduce time and space complexity [15].

PROPOSED GRAPHLAB MULTIWAY RECURRENT PATTERN EXTRACTION ALGORITHM

GraphLab is an open-source large-scale graph processing framework. It is implemented in C++, initiated at CMU, and presently supported by Dato Inc. It is fully based on shared memory abstraction and the GAS (Gather, Apply, and Scatter) processing model. The GraphLab abstraction consists of three main parts (i.e. data graph, modify function, and synchronization operation).

A. GraphLab Data Graph

The abstraction keeps the program state as a directional graph said as *Data Graph*. The Data Graph is a holder that controls the user-defined data and is represented by G= (V, E, D), where V is a set of vertices, E is a set of edges and D is a set of user-defined Data. The Data mainly contains algorithm state, statistical information, model parameters, etc. An individual user can able to add data randomly with each node $\{D_V: v \in V\}$ and link $\{D_{u\rightarrow V}: \{u, v\} \in E\}$ in G. Due to GraphLab abstraction being independent of edge direction, we denote D_{u-v} to represent bidirectional links. The Data Graph is mutable and its structure is static.

B. GraphLab Modify function

All algorithms are calculated in form of a modified function in GraphLab abstraction. The modify function in GraphLab is a stateless plan of action that changes the data within the scope of a node and schedules the succeeding execution of other to modify functions. The modify function takes to hold up as input vertex v and its scope S_v and gives back the next version of data in the scope as well as a set of tasks T that encodes subsequent task implementation.

Update: $f(v, S_v) \rightarrow (S_v, T)$ (1)

Once the modify function is executed, the new value of S_v returns to the data graph. Individual task comprising a modified function f and a vertex v.

C. GraphLab Synchronization operation

The synchronization operation is used for aggregating the global state. It will help in our paper to calculate the global frequency of a subgraph to extract frequent patterns.

Z= Finalise ($\sum_{v=0}^{V} Map(Sv)$) (2)

The sync operation is denoted over all scopes of graphs. This mechanism is automatically executed in the background of the main global statistical information of the data graph.

GRAPH MINING METHODS USED IN GRAHLAB

In this paper, to evaluate our proposed GraphLab Geometric Multiway Recurrent Pattern Extraction (GGMRPE) algorithm performance to extract the frequent subgraph from a distributed graph database, we do experimental analysis with five different graph mining algorithms such as PageRank, Connected Component, Triangle Counting, Collaborative Filtering, and Belief Propagation. Based on different parameters, the performance is checked by taking different datasets.

A. Page Rank

We use Auditing PageRank on the large graph (AURORA) to rank a node on the basis of importance such as the highest influence on ranking results. For the graph elements (i.e. vertices, links, patterns), we extracted frequent subgraphs that have to satisfy the threshold value. The algorithm is able to generate a recurrent pattern concerning its ranking result [16].

B. Connected Component

In our research work, the Connected Component algorithm is used for the parallel execution of Candidate Generation to reduce the time complexity. We implement *Linear Algebraic Connected Components* (LACC) using the linear algorithm as it is well-suited for distributed memory graphs and able to find out imbalanced collective communication patterns. We can use it for large graph sizes such as billions and trillions [17].

C. Triangle Counting

We use *Random Projection* based approach for triangle counting and generate well-defined conditions to check the frequency of subgraphs to extract recurrent subgraphs. This algorithm was used before partitioning the entire graph into different subparts [18, 19].

D. Collaborative Filtering

This paper implements a user-based nearest-neighbor collaborative filtering algorithm to extract subgraphs based on similarities. The similarity checking is done based on the user-given threshold value. If its value is greater than or equal to the given threshold value, we consider it as a recurrent pattern, otherwise discard it. We use a rating matrix to measure its prediction or neighbourhood, and the model-learning phase before candidate generation to eliminate the infrequent vertices of input graph datasets. This reduces execution time and decreases frequency count cost [20…22].

E. Belief Propagation

In this work, we use the estimation of the Bayesian network algorithm to enhance the search capability by giving user support value. We mainly use this algorithm to reduce computation costs at each iteration during candidate generation. The graphical methods are used to find out marginal or posterior probability or estimate the node with the highest probability based on the threshold value [23…25].

EXPERIMENTAL ANALYSIS

In this section, we implement and evaluate our proposed framework and compared it with the existing algorithm by hybridization. For experimental analysis, we used Intel (R) CPU 3.10 GHz PC with 32 GB RAM running on 64-bit windows 10 OS.

- **Relato Business Graph Database**

This graph dataset is downloaded from https://data.world/datasyndrome/relato-business-graph-database and is a collection of 373,663 company links from the web, gathered by the start-up Relato between 2015 and 2016. The node and links are used to represent the company and partnership respectively. Russell Jurney was used to assemble the database and loaded by Titan and Gremlin, and the Mongo database was used to organize the data and send queries.

- **Indie Map**

This dataset is downloaded from http://data.world/snarfed/indie-map and represents a social network of the liveliest Indie Web sites. It consists of 5.8M pages, 631M links, 706K relationships, and more. Here people have their private website for posting, sharing, organizing events, replying, and interacting on social sites such as Twitter, Facebook, etc.

- **Newberry Well 55-29 Stimulation Data 2014**

This dataset is downloaded from https://catalog.data.gov/dataset/newberry-well-55-29-stimulation-data-2014-f89b0. It was used to limit the cost of power consumed by EGS in central Oregon in 2008. There is three stimulation done in 2008, 2012, and

2014 respectively. It consists of 3452931 vertices and 12549123 edges representing the individual zone with partial blocking of the power flow.

- **Women and Career Advancement**

This database is downloaded from https://data.world/scuttlemonkey/women-and-career-advancement. This dataset collects information about women who successfully developed a pathway to achieve it properly. There are 32891654 vertices and 87695449123 edges to represent the individual ladies with the association between them.

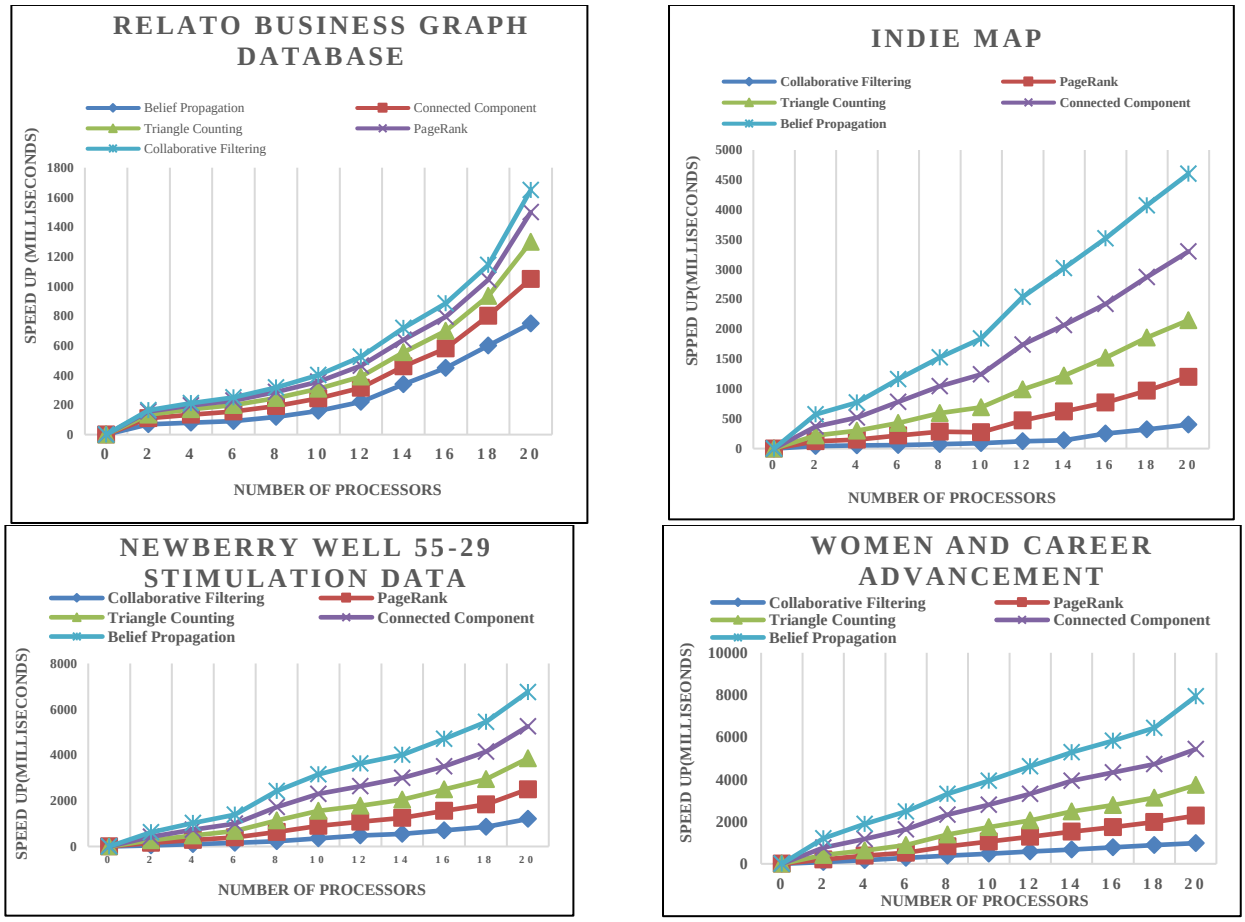
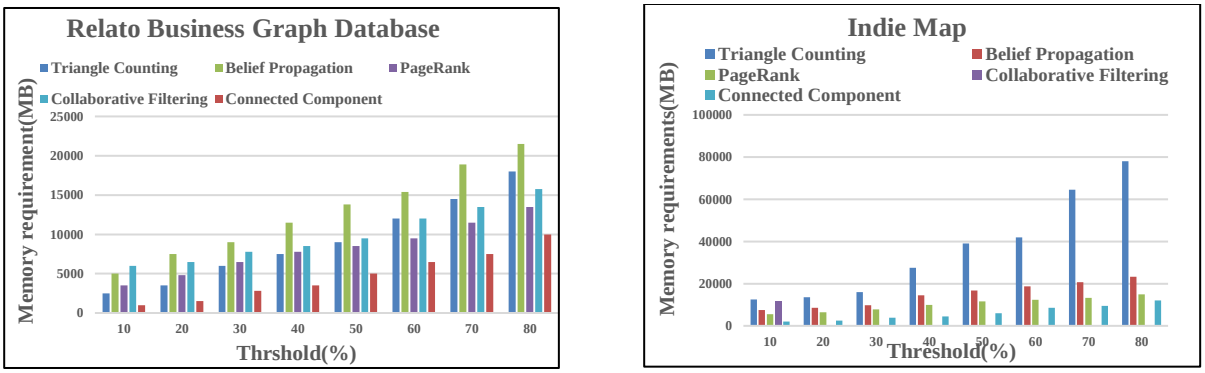
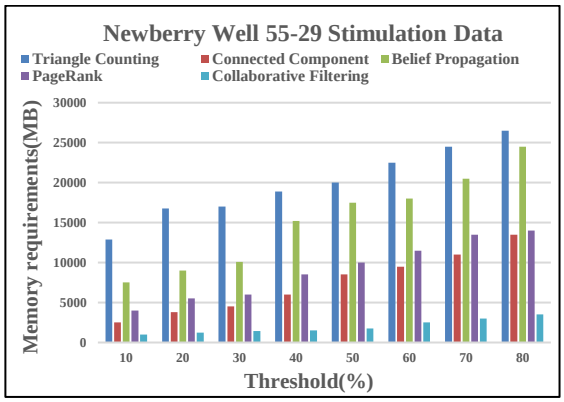


Figure 1: *Speed up vs various processors with four datasets*





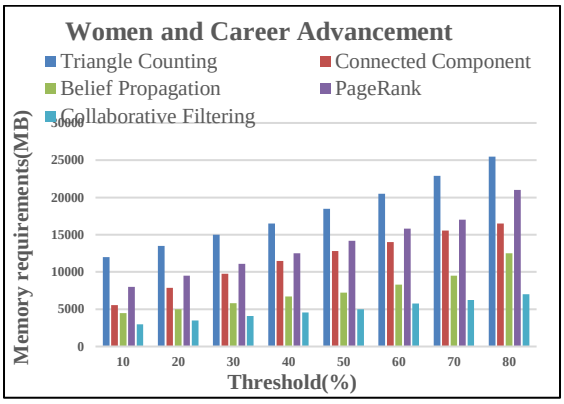
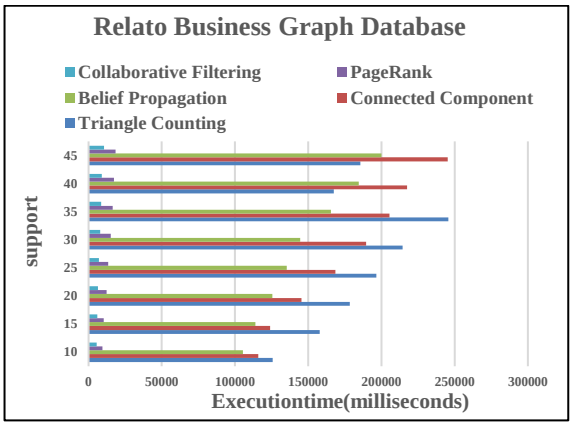
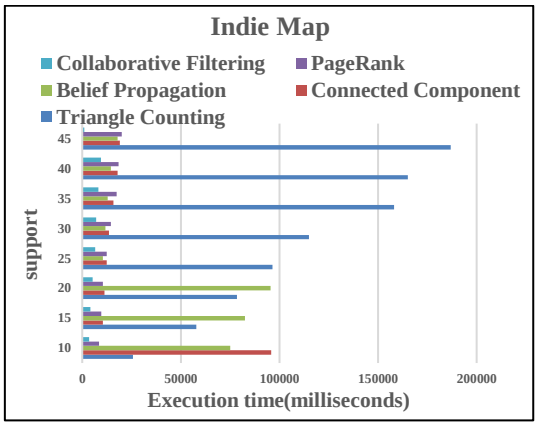
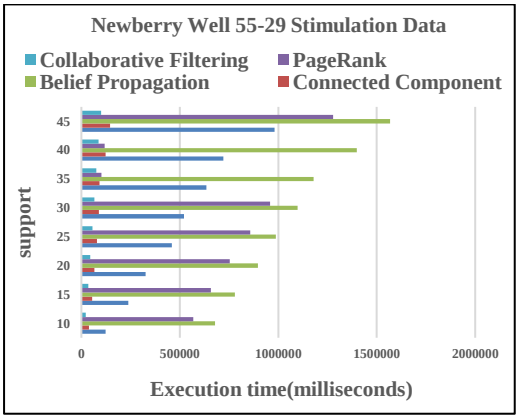


Figure 2: *Memory requirement vs threshold values*







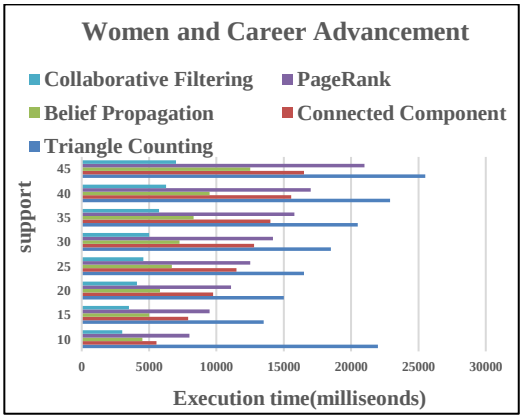


Figure 3: *Execution time vs support value*

We executed our proposed GraphLab Geometric Multiway Recurrent Pattern Extraction (GGMRPE) algorithm with four real datasets and merged them with algorithms such as PageRank, Connected Component, Triangle Counting, Collaborative Filtering, and Belief Propagation. We used different parameters to evaluate its performance such as speed-up time, number of processors, memory requirement, threshold values, execution time, etc. We performed the experimental analysis with a range of 20 different processors to achieve better speed as shown in Fig.1. With the growth in the number of processors, the speed up changed as per datasets. The highest speed-up is achieved with an increase in processors as follows: Collaborative Filtering algorithm in Relato Business Graph Database, Belief Propagation in Indie Map, Newberry Well 55-29 Stimulation Data 2014, Women and Career Advancement. When we increase the threshold from 10% to 80%, the memory requirement also increases due to the generalization of more candidate subgraphs. Relato Business Graph Database and Indie Map datasets have less memory requirement with Connected Components with increased support value, whereas Newberry Well 55-29 Stimulation Data 2014, Women and Career Advancement have low space storage with Collaborative Filtering among all five algorithms as shown in Fig.2.When we performed the analytical experiment with our proposed GGMRPE algorithm with four datasets, found

the execution time is directly proposal to support value and the Collaborative Filtering has the best among all remaining algorithms for all graph databases as shown in Fig.3.

CONCLUSION AND FUTURE WORK

We examined so many graph mining methodologies to generate recurrent patterns from distributed graph databases with our proposed GGMRPE algorithm with various datasets with respect to different parameters. The Collaborative Filtering algorithm gives the best result for execution time with changes in threshold value in all databases. In this paper, we are able to reduce memory requirements as per input graph database changes.

REFERENCES

1. Jagannadha Rao D. B., "Distributed Frequent Subgraph Mining Using Gaston and MapReduce", International Journal on Semantic Web and Information Systems (IJSWIS) 17(2) 2021 Pages: 18.
2. Nisha Babu, Ansamma John, "A distributed approach to weighted frequent Subgraph mining", International Conference on Emerging Technological Trends (ICETT) 2016.
3. Eichinger Frank, Matthias Huber and Klemens Böhm, "On the usefulness of weight-based constraints in frequent subgraph mining" in Research and Development in Intelligent Systems XXVII, London:Springer, pp. 65-78, 2011.
4. Jiang Chuntao, Frans Coenen and Michele Zito, "Frequent sub-graph mining on edge weighted graphs", International Conference on Data Warehousing and Knowledge Discovery, 2010.
5. XingJiang, HuiXiong, ChenWang, Ah-HweeTan, "Mining globally distributed frequent subgraphs in a single labeled graph", Data & Knowledge Engineering Volume 68, Issue 10, October 2009, Pages 1034-1058.
6. N. Senthilselvan, V. Subramaniyaswamy, V. Vijayakumar, Hamid Reza Karimi, Logesh Ravi, "Distributed frequent subgraph mining on evolving graph using SPARK", Intelligent Data AnalysisVolume 24Issue 32020 pp 495–51
7. Aida Mrzic, Pieter Meysman, Wout Bittremieux, Pieter Moris, Boris Cule, Bart Goethals and Kris Laukens, "Grasping frequent subgraph mining for bioinformatics applications", BioData Mining (2018) 11:20 https://doi.org/10.1186/s13040-018-0181-9.
8. Schreiber F, Schwöbbermeyer H. "Towards motif detection in networks: Frequency concepts and flexible search". In Proceedings of the International Workshop on Network Tools and Applications in Biology (NETTAB04); 2004. p. 91–102.
9. Kashani ZR, Ahrabian H, Elahi E, Nowzari-Dalini A, Ansari ES, Asadi S, Mohammadi S, Schreiber F, Masoudi-Nejad A. Kavosh: a new algorithm for finding network motifs. BMC Bioinforma. 2009;10 (1):318.
10. Inokuchi A, Washio T, Motoda H. An apriori-based algorithm for mining frequent substructures from graph data. In: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery. PKDD '00; 2000. p. 13–23.
11. Kuramochi M, Karypis G. Frequent subgraph discovery. In: Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference On; 2001. p. 313–20.
12. Kuramochi M, Karypis G. An efficient algorithm for discovering frequent subgraphs. IEEE Trans Knowl Data Eng. 2004;16(9):1038–51.
13. Bismita S. Jena, Cynthia Khan, and Rajshekhar Sunderraman, "High Performance Frequent Subgraph Mining on Transaction Datasets: A Survey and Performance Comparison", BIG DATA MINING AND ANALYTICS ISSN 2096-0654 02/05 pp159–180 Volume 2, Number 3, September 2019 DOI: 10.26599/BDMA.2019.9020006.
14. Peng Jiang, Rujia Wang, Bo Wu, "Efficient Mining of Frequent Subgraphs with Two-Vertex Exploration", arXiv:2101.07690v2 [cs.DB] 5 Feb 2 2021.
15. M. Sailaja, Dr.C.V.P.R. Prasad, "A Research on Frequent Sub Graph Mining From Distributed Database ,Journal of Advanced Research in Dynamical and Control Systems, volume 11 | Issue 8, Pages: 263-268.
16. P. Adler, C. Falk, S. A. Friedler, G. Rybeck, C. Scheidegger, B. Smith, and S. Venkatasubramanian, "Auditing black-box models for indirect influence," in Data Mining (ICDM), 2016 IEEE 16th International Conference on. IEEE, 2016, pp. 1–10.
17. Yogzhe Zhaga , Ariful Azadb , Aydın Buluc¸" Parallel Algorithms for Finding Connected Components using Linear Algebra", Preprint submitted to Elsevier May 12, 2020.
18. Abdurrahman Yas¸ar, Sivasankaran Rajamanickam , Jonathan Berry, and Umit V. C¸ ataly ¨ urek ¨ ," A Block-Based Triangle Counting Algorithm on Heterogeneous Environments", arXiv:2009.12457v1 [cs.DS] 25 Sep 2020
19. Mihail N. Kolountzakis , Gary L. Miller , Richard Peng , Charalampos E. Tsourakakis," Efficient Triangle Counting in Large Graphs via Degree-based Vertex Partitioning" , Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 803-812 (2008) .
20. Zhang, Y., J. Callan and T. Minka, 2002. Novelty and redundancy detection in adaptive filtering. Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, New York, pp: 81-88, 2002.
21. Ziegler, C., Semantic web recommender systems. Proceedings of the EDBT Workshop, 2004 pp: 78-89.
22. Ziegler, C.N., S.M. Mcnee, J.A. Konstan and L. Georg,. Improving recommendation lists through topic diversification. Proceeding of the 14th International World Wide Web Conference (WWW). ACM, Chiba, Japan, pp: 22-32, 2005.
23. J. M. Coughlan and S. J. Ferreira. Finding deformable shapes using loopy belief propagation. In ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III, pages 453–468, London, UK, 2002. Springer-Verlag.
24. C. Crick and A. Pfeffer. Loopy belief propagation as a basis for communication in sensor networks. In Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI-2003), pages 159–166. Morgan Kaufmann Publishers, 2003.
25. C. Echegoyen, J. A. Lozano, R. Santana, and P. Larra˜naga. Exact Bayesian network learning in estimation of distribution algorithms. In Proceedings of the 2007 Congress on Evolutionary Computation CEC-2007, pages 1051–1058. IEEE Press, 2007.