# Graphical Analysis for Big Data Analytics

Pathan Faisal Khan

***Abstract*—The term big data has quite significance. It deals with data of huge volumes. This data might be and might not be structured and if structured it might be and might not be of the same data structure. So for different data structures different types of analytical approaches have to be defined. This paper specifically deals with analysis of graph data in big data. The focus of this paper will be on distributed computing algorithms. Different types of graph analysis are covered in this paper. There are 4 types of graph analytics [7], path analysis, connectivity analysis, community analysis and centrality analysis. Path analysis is used for finding out the shortest distance between two nodes. Connectivity analysis is used for analyzing weaknesses in the network. Community analysis is used to focus on the interactions between nodes. Centrality analysis method is used to find the relevance between each node. Along with these techniques, the paper will investigate Graph Neural Networks. Because of challenges in GNN, many methods were proposed recently. This paper has cover some of those methods. Recurrent Graph Neural Networks, Convolutional Graph Neural Networks, Graph Autoencoders, Spatial-Temporal Graph Neural Networks are some of the most common GNN models [33].**

***Index Terms*—Big Data Analytics, Centrality Analysis, Community Analysis, Connectivity Analysis, Graph Analytics, Graph Neural Network, Path Analysis**

## I. Introduction

WITH the introduction of internet in the 90s, there has been tremendous innovation in tech industry. This changed the way organizations, businesses, governments function. It even changed the lifestyle of the people. Major contributions to the tech space were not until early 2000s due to innovations in computational power and during this period, the volume of data generated with introduction of social media and other services for the masses has risen a lot. Data is being created every second of the data. In 2013, Instagram users shared 3600 photos every minute, while in 2019, the number of photos shared every minute reached 46,740. The world internet population has increased from 2.5 billion to 3.7 billion [1]. It is estimated that by 2020, 40 trillion GB of data would be generated [2] which means internet user generates nearly 2500000 terabytes of data every day [1]. Most of the data being generated is contributed by social media on which an average user spends 33% of his/her online time. This is why in 2019, there are 2.3 billion users active on Facebook [3].

Because of this vast amount of data, there was a need to develop more efficient and cost-effective data storage. This led to the introduction of the term Big Data in early 2005 [4]. Big data is the type of data that has a high variety, large volume, high velocity, greater veracity, and extreme value also it is continuously growing on a large scale. These characteristics of the big data are referred to as the 5Vs. It will not be a surprise that the data is unstructured as it is being collected from multiple sources. Big data can be comprised of logs of the traffic coming in on a website, messages generated on a social media site, attributes of mouse clicks, details of products stored on an e-commerce website, medical data of a hospital, bank transactions, satellite data, and many other sources which generate data.

Under the supervision of Dr. Shagufta Henna, Letterkenny Institute of Technology, Letterkenny, CO. Donegal

Since generating data is an easier task than getting useful insights out of it, there was a need to emphasize on its analysis. But because of the sheer volume of high dimensional, unstructured, and highly inconsistent data, running traditional methods for analysis might miss out on the hidden structures of the data. Thus, there was a need to devise powerful algorithms and provide high computational powers that can solve these problems. Due to the introduction of cloud computing and its scalable nature, researchers were able to develop algorithms to mine and make out meaningful insights from this data. With the right analysis methods, it can yield greater insights leading to stronger and strategic decisions. Using big data analysis, Netflix manages to easily save $1 billion every year [5]. Wikibon, an organization sharing tech-related knowledge, has estimated the market worth of big data analytics to a whopping $49 billion for the year 2019 [6].

This paper will discuss some graph analysis methods used on big data in Section II. A comparison will then be done on some of these techniques in Section III. Also, Graph Neural Network will be briefly explored along with other graph analytic techniques in II-E. The paper will finally conclude with the observations from this report in Section IV.

## II. Literature Review

Often data generated has relations among themselves. This data can be structured or unstructured or a mix of both. Since it is not feasible to understand these relations using the traditional big data analytics techniques, a better model had to be devised. A graph model was proposed to connect the data. Graphs are effective for analyzing, making recommendation systems, and mining social networks. Due to the flexibility of this model it allows large quantities of information from many sources to be quickly absorbed and linked in ways that addressed the limitations in the source structures. A good way of representing the graph model is connections of a social media account; it represents a graphical structure with connections (edges) formed between different accounts (nodes/entities). This model enabled analyzing relationships and deducing interesting patterns between accounts (entities) in the structure. Graph analytics is the term used to define these methods of analysis. It is defined as an alternative to the conventional data warehouse model as a system for allowing analysts to check structured and unstructured data from different sources. Some business use cases of graph analytics include healthcare quality analysis, cybersecurity, and correlation findings.

## A. Path Analysis

Path analysis algorithms are used for exploring a graph which may either lead to discovery of new or optimal paths. A path may be decided as an optimal path on basis of the number of hops required to traverse, weights of visited nodes, avoiding/including specific nodes/paths or in some cases based on an optimization function. One of the most common use cases is getting the shortest path using Google Maps directions. Some other applications include but not limited to are customer behavior analysis on an e-commerce website and re-routing in network to fix problems with network capacity. The path analysis algorithms being covered in this paper are Parallel Breadth-First Search [8], Parallel Depth-First Search [9], [10], Single-Source Shortest Path [11], All-Pairs Shortest Path, Minimum Weight Spanning Tree [12], [13], and Random Walk [14].

*1) Parallel Breadth-First Search:* To understand parallel breadth-first search, serial or conventional breadth-first search (BFS) will be explored first. Since, the paper is dealing with graph based analysis methods, BFS will be covered for graph implementation rather than tree implementation. The difference between both implementations will also be covered in this section.

The BFS algorithm starts at an arbitrary node in the graph and travels to all nodes at current level $n$ before moving to the next level $n+1$ of the graph. The following figure 1 shows the order of expansion/exploration of nodes. This example shows a graph without loops in it. This is the major difference between a graph and a tree; a graph may have loops in it, while a tree doesn't have loops. Hence, to avoid the loops, BFS implementation of a graph is different as it requires a mechanism to track visited nodes to avoid unnecassary infinite loops.

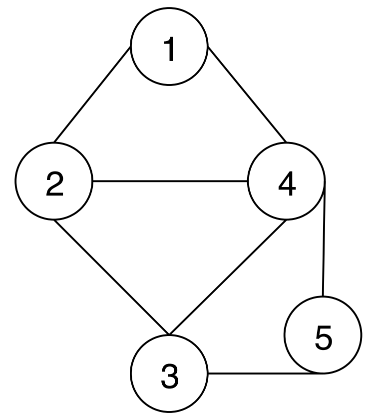


Fig. 1. The numbers depict order of traversal of nodes in a BFS implementation

The problem with serial BFS is that time and memory consumption depends on the number of branches and the depth of the graph. In worst case the time and space complexity is O($b^d$) where $b$ is branches and $d$ is total levels in the graph. Since we are dealing with big data analysis where data is of huge size, it will consume a lot of time.

To solve this issue with BFS, concept of parallelism was introduced. Parallel computing distributes the load to multiple processors hence taking of load from a single process and distributing it to others to reduce the overhead. The parallel version of BFS can be done using two approaches either using shared memory or by using distributed memory. The shared memory implementation of BFS generates Breadth First Spanning Trees (BFSTs) of a given graph G with $n$ nodes. The time complexity of this step is given by O($log(d).log(n)$) where $d$ is the diameter of G which is far more better than serial BFS with time complexity O($b^d$). The number of processors used for this process depends on the number of nodes; the time complexity is defined by O($n^3$) [36]. If the graph is undirected, the algorithm will generate $n$ BFSTs.

In the distributed memory version of BFS, each process has their own memory and they have to share messages amongst each other to share their data. Since there is a overhead of communication in this approach, shared memory BFS will provide higher bandwith with low latency [37].

*2) Parallel Depth-First Search:* Parallel depth-first search also deals with parallelism as discussed in the previous section. To understand this, it's necessary to go through serial depth-first search (DFS).

The DFS algorithm starts at an arbitrary node and travels deep into a path before coming back a step and exploring the next path. This algorithm is used on a hierarchical data. Since the algorithm might face issues with infinite loops due to the fact we are traversing on a graph, similarly to BFS, DFS for graphs also have to be implemented with a mechanism to store visited nodes. The time complexity of DFS is similar to BFS, O($b^d$) while the space complexity is O($d$).

Serial DFS also faces same problem as serial BFS, that is it will also consume a lot of time for big data analysis. So to avoid this, parallel DFS was proposed. To parallelize DFS, the graph is splitted among different processors. Each processor performs it's task independently until it finishes it after which the processor requests unfinished section of the graph from other processors. Some of the models of implementation of parallel DFS include shared memory models, boolean circuits, and parallel comparison trees [38].

There are many implementations of parallel DFS and most of them showed logarithmic running time [38]. A logarithmic running time is much more efficient than an exponential running time which is showed by a serial DFS.

*3) Single-Source Shortest Path:*

*4) All-Pairs Shortest Path:*

*5) Minimum Weight Spanning Tree:*

*6) Random Walk:*

## B. Connectivity Analysis

Connectivity analysis helps in identifying the strength of connections/vertices between two nodes in a graph. It is most commonly used for analyzing weaknesses in a graph network. Before diving into the algorithms, it's important to understand how strength of a connection is defined in a graph.

A component comprises of 2 or more nodes connected with each other. A concept of Strongly Connected Components (SCCs) [39] is used. A component is said to be strongly connected when a given pair of nodes in the component has connections with each other.

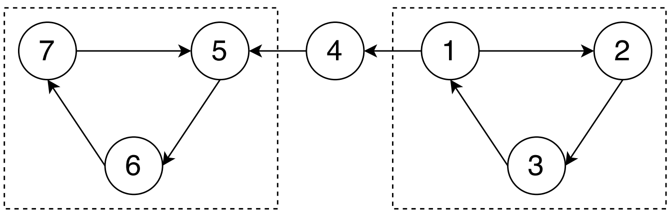


Fig. 2. Graph shown has two SCCs with one node connecting both of them

In the above figure 2, there are two SCCs, one with $1, 2, 3$ nodes and the other with $7, 5, 6$ nodes connected with node $4$. As it can be seen that $1$ is connected with $2$ and $2$ is connected with $3$ which is then connected with $1$, thus it is said that both of these components are strongly connected.

SCCs help in forming clusters in a graph. With the help of such clusters, analysis can be performed in the network. Some use cases of connectivity analysis includes social network and internet network analysis. Now the problem lies in finding out such SCCs. DFS and BFS are helpful in finding such components. Kosaraju devised an algorithm which in linear time helped in finding out SCCs [40]. This paper will cover Kosaraju's algorithm's modification on BFS and DFS.

*1) Kosaraju's BFS and DFS:* Kosaraju's original proposed algorithms was based on DFS and some implementations exists of it on BFS. The DFS implementation does a simple trick by performing DFS two times on the same graph. Taking figure 2 as example, the algorithm will perform first DFS on the graph. It will reverse the directions in the graph and start at 7, explore in the order $7-6-5-4-1-3-2$. Now the node which finished first or visited at the end will be assigned the finisihing position starting from 1. The nodes will be renamed with their corresponding positions as shown below in figure 3.

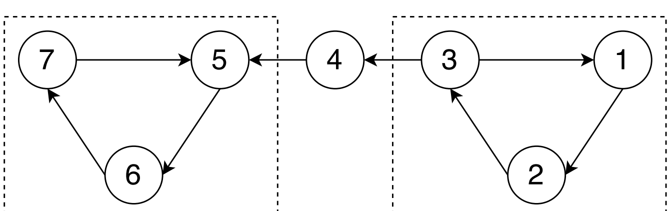


Fig. 3. Nodes renamed with their position after the first DFS

Now second DFS will be performed and as 7 was the start node previously, this time also it will start from 7. The order will be then $7-6-5$ which is our first SCC and then starting DFS from 3, the order will be $3-1-2$ which is our second SCC. Thus, the algorithm pointed out two SCCs in the graph. The time complexity of this algorithm will be O$(v+e)$ where $v$ is vertices of the graph and $e$ is edges.

Kosarju's BFS algorithm also works in the same way; it performs BFS twice instead of DFS.

## C. Community Analysis

A community in a graph network is defined by a group of nodes which have dense connections amongst themselves and sparse connections with the other group of nodes. Community analysis points out interactions between nodes. It is used for clustering nodes with similar attributes. This analysis also helps in analyzing how nodes are clustered or partitioned. The strength of a particular group can also be determined with this analysis. Community analysis is used in indentifying proteins involved in a biochemical process, recommendations for a group of people, and evaluating social networks.

*1) Label Propagation:* Raghavan et al. proposed this algorithm back in 2007 [16]. The algorithm worked on exploring communities in a graph network.

The algorithm functions in the following way. Consider a node $n$ with neighbors $n1, n2, n3$ and so on also every node has a tag identifying them with which community they are a member of. Therefore, $n$ will identify its community based on its neighbors' communities. So the algorithm starts with assigning a unique tag on all nodes present in the graph; this part of the algorithm suggests that it is a sem-supervised algorithm. These tags will then propagate in the network. On each step of the propagation, nodes will update their tags to the tag which is assgined to the majority of its neighbors. When the propagation ends, nodes are grouped based on their tags. This process results in formation of communities. The following figure 4 explains how step by step propagation leads to community formations.

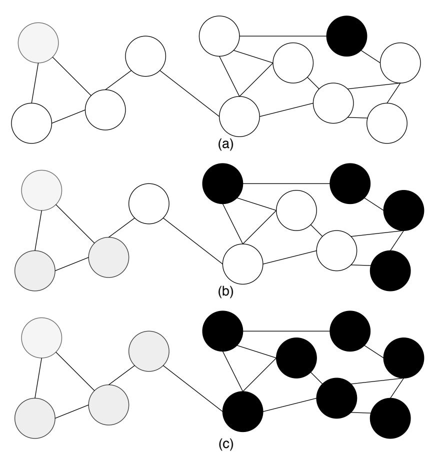


Fig. 4. (a) Step 1- Random tags have been initialized which is a sem-supervised method, (b) Step 2- In this step tags are adopted from neighbors, and (c) Step 3- Finally all nodes have been assigned tags

This algorithm performs near linear time as mentioned by Raghavan et al. [16] which makes it a better contender among other community analysis aglorithms. The time complexity is given by O($m + n$) where $m$ is nodes in a graph and $n$ is vertices.

2) *Strongly Connected Components:*

3) *Weakly Connected Components:*

4) *Louvain Modularity:*

5) *Node Clustering Coefficient:*

6) *Average Clustering Coefficient:*

7) *Bron-Kerbosch:*

### D. Centrality Analysis

Centrality analysis is used to identify the relevance between each node. It measures the importance of nodes. It is used to find the influence of people in a social network, used to get the most popular web page based on access, and key nodes in a network. Although this method was introduced specifically for social network analysis, it has been adopted in many other fields.

1) *Degree Centrality:*

2) *Eigenvector Centrality:*

3) *Katz Centrality:*

4) *PageRank Centrality:*

5) *Closeness Centrality:*

6) *Betweenness Centrality:*

### E. Graph Neural Networks

## III. Discussion

## IV. Conclusion

## References

[1] "Resources - Whitepapers, Infographics & Webinars — Domo." [Online]. Available: https://www.domo.com/learn.

[2] C. Petrov, "Big Data Statistics 2019," Tech Jury. [Online]. Available: https://techjury.net/stats-about/big-data-statistics

[3] "Domo Resource - Data Never Sleeps 7.0." [Online]. Available: https://www.domo.com/learn/data-never-sleeps-7.

[4] "A Short History Of Big Data," Datafloq. [Online]. Available: https://datafloq.com/read/big-data-history/239.

[5] E. Team, "How Netflix Uses Big Data to Drive Success," inside BIG-DATA, 20-Jan-2018.

[6] "Wikibon's 2018 Big Data Analytics Trends and Forecast," Wikibon Research, 28-Feb-2018. .

[7] "What is graph analytics?," IBM Big Data & Analytics Hub. [Online]. Available: https://www.ibmbigdatahub.com/blog/what-graph-analytics. [Accessed: 12-Dec-2019].

[8] A. Buluç and K. Madduri, "Parallel Breadth-First Search on Distributed Memory Systems," p. 12.

[9] V. N. Rao and V. Kumar, "Parallel depth first search. Part I. Implementation," Int J Parallel Prog, vol. 16, no. 6, pp. 479–499, Dec. 1987.

[10] M. Naumov, A. Vrielink, and M. Garland, "Parallel Depth-First Search for Directed Acyclic Graphs," in Proceedings of the Seventh Workshop on Irregular Applications: Architectures and Algorithms - IA3'17, Denver, CO, USA, 2017, pp. 1–8.

[11] M. Kranjčević, D. Palossi, and S. Pintarelli, "Parallel Delta-Stepping Algorithm for Shared Memory Architectures," arXiv:1604.02113 [cs], Apr. 2016.

[12] J. W. Kim, H. Choi, and S.-H. Bae, "Efficient Parallel All-Pairs Shortest Paths Algorithm for Complex Graph Analysis," in Proceedings of the 47th International Conference on Parallel Processing Companion, New York, NY, USA, 2018, pp. 5:1–5:10.

[13] L. Fitina, J. Imbal, V. Uiari, N. Murki, and E. Goodyear, "An Application of Minimum Spanning Trees to Travel Planning," vol. 12, p. 11, 2010.

[14] L. L. Sz, "Random Walks on Graphs: A Survey," p. 46.

[15] M. Sharir, "A strong-connectivity algorithm and its applications in data flow analysis," Computers & Mathematics with Applications, vol. 7, no. 1, pp. 67–72, Jan. 1981.

[16] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," Phys. Rev. E, vol. 76, no. 3, p. 036106, Sep. 2007.

[17] R. Tarjan, "Depth-First Search and Linear Graph Algorithms," SIAM J. Comput., vol. 1, no. 2, pp. 146–160, Jun. 1972.

[18] A. Monge and C. Elkan, An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records. 1997.

[19] Y. An, J. Janssen, and E. E. Milios, "Characterizing and Mining the Citation Graph of the Computer Science Literature," Know. Inf. Sys., vol. 6, no. 6, pp. 664–678, Nov. 2004.

[20] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," J. Stat. Mech., vol. 2008, no. 10, p. P10008, Oct. 2008.

[21] H. Lu, M. Halappanavar, and A. Kalyanaraman, "Parallel Heuristics for Scalable Community Detection," arXiv:1410.1237 [physics], Oct. 2014.

[22] T. Schank and D. Wagner, "Finding, Counting and Listing All Triangles in Large Graphs, an Experimental Study," in Experimental and Efficient Algorithms, vol. 3503, S. E. Nikoletseas, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 606–609.

[23] H. C. Johnston, "Cliques of a graph-variations on the Bron-Kerbosch algorithm," International Journal of Computer and Information Sciences, vol. 5, no. 3, pp. 209–238, Sep. 1976.

[24] S. C. Antoro, K. A. Sugeng, and B. D. Handari, "Application of Bron-Kerbosch algorithm in graph clustering using complement matrix," presented at the International Symposium On Current Progress In Mathematics And Sciences 2016 (ISCPMS 2016): Proceedings of the 2nd International Symposium on Current Progress in Mathematics and Sciences 2016, Depok, Jawa Barat, Indonesia, 2017, p. 030141.

[25] L. C. Freeman, "Centrality in social networks conceptual clarification," Social Networks, vol. 1, no. 3, pp. 215–239, Jan. 1978.

[26] Phillip Bonacich Reviewed, "Power and Centrality: A Family of Measures," American Journal of Sociology, vol. 92, no. 5, pp. 1170–1182, 1987.

[27] C. F. A. Negre et al., "Eigenvector centrality for characterization of protein allosteric pathways," Proc Natl Acad Sci U S A, vol. 115, no. 52, pp. E12201–E12208, Dec. 2018.

[28] L. Katz, "A new status index derived from sociometric analysis," Psychometrika, vol. 18, no. 1, pp. 39–43, Mar. 1953.

[29] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web.," 11-Nov-1999. [Online]. Available: http://ilpubs.stanford.edu:8090/422/. [Accessed: 12-Dec-2019].

[30] G. Sabidussi, "The centrality index of a graph," Psychometrika, vol. 31, no. 4, pp. 581–603, Dec. 1966.

[31] S. P. Borgatti, "Centrality and network flow," Social Networks, vol. 27, no. 1, pp. 55–71, Jan. 2005.

[32] L. C. Freeman, "A Set of Measures of Centrality Based on Betweenness," Sociometry, vol. 40, no. 1, pp. 35–41, 1977.

[33] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," arXiv:1901.00596 [cs, stat], Dec. 2019.

[34] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," IEEE Trans. Neural Netw., vol. 20, no. 1, pp. 61–80, Jan. 2009.

[35] "Special Issue on Graph-based Methods for Large Scale Financial and Business Data Analysis - Call for Papers - Elsevier." [Online]. Available: https://www.journals.elsevier.com/pattern-recognition/call-for-papers/graph-based-methods. [Accessed: 09-Dec-2019].

[36] R. K. Ghosh and G. P. Bhattacharjee, "Parallel breadth-first search algorithms for trees and graphs," International Journal of Computer Mathematics, vol. 15, no. 1–4, pp. 255–268, Jan. 1984, doi: 10.1080/00207168408803413.

[37] "Parallel breadth-first search," Wikipedia. 03-Nov-2019. [Online]. Available: https://en.wikipedia.org/wiki/Parallel_breadth-first_search. [Accessed: 06-Jan-2020].

[38] J. Freeman, "Parallel Algorithms for Depth-First Search," Technical Reports (CIS), Oct. 1991.

[39] "Strongly Connected Components Algorithm Optimized," Weiming Hu. [Online]. Available: https://weiming-hu.github.io/strongly-connected-components/. [Accessed: 06-Jan-2020].

[40] "Kosaraju's algorithm," Wikipedia. 07-Mar-2019. Available: https://en.wikipedia.org/wiki/Kosaraju's_algorithm. [Accessed: 06-Jan-2020].