AN ADAPTIVE GRAPH DATA EXTRACTION USING PATTERN MATCHING AND HYPER GRAPH FORMATION

Anuja Krishna Gaikwad

Assistant Professor
Computer Engineering Department
Sinhgad Academy of Engineering , Pune, India

Abstract : **As the adoption of social media has been exponentially increasing, there has been an increase in the amount and frequency of data being shared between the users. Due to the increase in the user base of the websites, it is highly beneficial for the organizations to retrieve users with similar interests into the relational graph database. There has been substantial research in this area for the extraction of the similarity in the users with the help of identifying the matching sub-graphs. But this process is cumbersome and usually leads to a lot of false positives. Therefore, the proposed system solves this problem by utilizing pattern identification for determining the similarity between users by introducing pattern matching and hyper graph analysis accompanied by strong pruning techniques.**

Index Terms - **Hyper graph, Neo4j, Pattern matching, preprocessing**

I. Introduction

Graphs are one of the most efficient data structures that have ever been produced. They are one of the most useful data structures and are used very extensively for the purpose of representing complex entities. Most of the entities we are acquainted with in our day to day lives can be represented in the graph data structure. It is one of the most versatile data structures there, that can successfully be modified to fit various purposes. The Graphs are very powerful data structures capable of representing various entities that are complex in nature. The representation with the help of graphs is very simplified and can be made simplistic in that form for the understanding of the concept. The representation of the graphs is very simple, where various aspects of the entities are presented with the help of the vertices, edges and the general flow of the graph.

As the data is being represented in the graph, various symbols and entities signify a part of the representation of the data. Such as, the vertices in the graph represent the objects of the entities, most of these entities have certain attributes related to the entities or the objects on that vertex. The edges in a graph stand for the relationships between the entities in the graph. The graph is composed of certain objects that form the vertex of the graph and these objects are interrelated and these relationships are maintained and represented by the edges of the graphs. There has been an increase in the usage of these graphs as they are versatile and the representations are very easily understood in the form of a graph. The graphs form a legible representation of various entities.

The inherent simplicity and the powerful nature of the graphs have been due to the fact that it has always been present and not a new age discovery. But due to the recent advances in computer processing technologies and the steady increase in the capacities, graphs,and other structural representations have been increasingly adopted and used extensively for the purpose of Pattern Recognition. There are various different techniques that can be utilized for the purpose of pattern recognition with the help of graphs. Graphs can help identify patterns in two distinct ways: -

1- Graph Matching
2- Graph Embedding

These techniques are quite powerful and useful implementations of the Graph Data structure.

Graph Matching is one technique for the extraction of frequent patterns with the help of this innovative graph data structure. This method is utilized for determining the correlation between the vertices and edges of the graphs. This correspondence is subject to certain stringent constraints that are common between the two graphs or sub-graphs. The graphs are then compared to the various similarities between them. There are various categories of Graph matching that are defined for a particular type of graph and its sub-graphs. there are, therefore, two extensive categories that are responsible for the various graph matching techniques. The first one is very precise and will only be utilized for a very strict matching. The source and destination objects or graphs have to be explicitly similar including the sub graphs as the correspondence follows a very stringent approach to the matching.

The other type of matching is not that strict and can be a lot more complaisant in comparison with the technique that has been discussed above. This technique is a lot more tolerant of dissimilarities and can accept two graph structures that are not similar structurally. This is a lot more realistic approach for the identification of patterns as it is highly unlikely that all the elements in the graph are similar for the pattern.

The former process has been named as an intolerant network where the stringent matching is considered, the latter on the other hand is a very tolerant network that has relaxed matching guidelines that are error-tolerant and the matching can happen even if the graphs and their sub-graphs are structurally different. This process is also aptly called structural matching process as it utilizes the structure of the graphs in question.

In this paper, section 2 is dedicated for literature review of past work and Section 3 describes the details of the proposed methodology. Section 4 deals with the results and discussions and then finally section 5 concludes this paper.

II. RELATED WORK

S. Au [1] expresses the usefulness of graphs as a very crucial data structure that has various uses. The concept of graphs can be applied to various applications such as governance, engineering, commerce and science. Due to the fact that most of the implementation that use this utilise the static model, which introduces a lot of pragmatic and conceptual issues. Therefore, to ameliorate this issue the authors have proposed an elastic and dynamic model called Joy Graph that solves all of the issues encountered in the static model.
Research Gap – as most of the researchers have only begun to understand the concept of elasticity in the context of processing graphs, there is still a lot of optimization that can be done.

C. Mayer [2] introduces the various different methods of analyzing and processing graphs that are prevalent, such as GraphX or Power Graph. These techniques take advantage of the parallel analytics offered by partitioning the graph using algorithms. But these graph partitions are difficult to process individually as partitioning increases the workload in a homogenous network. To solve this problem the authors have introduced a graph processing system that takes into consideration the network costs and traffic into account, called as GrapH.
Research Gap – Due to the orthogonal nature of the task placement procedure in the graph partitioning phase, the authors could have utilized this of top of their system.

T. Chen explains the concept of processing huge graphs, as it is typically done with the help of partitioning the graphs into smaller pieces. The researchers convey that the concept of graph partitioning is as powerful as the partitions. Therefore, for efficient and optimised partitioning of the graphs on distributed memory systems, the authors propose Bulk Swap (BS). BS is a partitioning algorithm which relies on the scatter-gather scheme to achieve the partitioning. Extensive experiments show that this technique is highly effective.
Research Gap – The technique has been very recently developed by the authors and has a lot of potential to be scaled further and optimise it for better efficiency. [3]

J. Wang [4] explores the realm of distributed graph computation. The authors explain that the larger graph that are there are processed by partitioning the graph. This is usually done by the GAS model that utilises the edge vectors to facilitate the production of graphs. This is a lengthy process that can slow down the process. Therefore, the researchers presented an innovative concept for optimisation of the performance of the graph analysis system. The authors concluded that the computational power and the communication bandwidth are the bottlenecks faced.
Research Gap – Due to extensive amounts of experimentation the authors have concluded that most of the limitations are caused by a bottleneck and can be resolved by a higher bandwidth and a better processor.

Z. Li [5] states that graphs are becoming increasingly popular as a data structure and are seeing wide spread increase in their usage currently. But most the researchers have not implemented the support for searching through the graph traversals. This is the reason why the authors have implemented a concurrent breadth-first -

search algorithm. This technique has been proven to be highly effective and has one of the highest efficiencies in comparison to the traditional techniques.

Research Gap – As the researchers have only recently implemented this technique, still a lot of optimisation can be achieved such as to improve the scalability, performance and have a nice memory.

H. Long explains the concept of processing large-scale graphs, as it is usually done with the help of distributing the graphs into smaller pieces and processing them parallelly. But this is not without its shortcomings as distribution of the graphs can lead to slow iterative convergence and large communication overhead. To ameliorate the effects, the researchers propose a distributed graph processing technique that utilises cluster-cased mechanism to get rid of the limitations posed by traditional methods.[6]

U. Cheramagalath [7] explains that graphs model especially of social information systems have numerous edges amounting to trillions. These types of graphs cannot be processed in one go, as they have to be partitioned and segregated among various machines to be analyzed in clusters. Therefore, the authors have proposed DH-Falcon a domain-specific language that can be used for the processing of large graphs parallelly. A lot of experimentation has been done which confirms the superiority of the technique in comparison with the traditional techniques.

Research Gap – The DH-Falcon approach has been only limited to input in only one format, future work should enable a lot more formats as input.

P. Liakos [8] expresses concern over the ever increasing sizes of graphs, some of them having traversals into trillions in number. This is very problematic as it gets increasingly difficult to process graphs that are very large without hitting a bottleneck. As large graphs require an ever-increasing amount of memory to process, this is a great hurdle to pass. Therefore, to ameliorate these effects, the authors present an innovative concept of analyzing graphs by utilizing the empirically observed properties in a graph generated due to human activity. This technique has been proven to be highly effective in comparison to traditional techniques.

G. Slota states that there has been an unprecedented increase in the usage of graphs as a data structure and it benefits have been very convincing. But due to the increased amount of research being focused more on the problem of optimizing the various graph analysis algorithms that are being used extensively. Therefore, the authors have presented a technique that could aggregate the two most important factors of graph processing, ease of use and scalability. As both of these issues have been at the centre of prominent research. The technique has been demonstrated to outperform several conventional methods and techniques. [9]

Research Gap – As this is an area of experimental research, there are certain areas which can be improved in the technique, such as load balancing methods, and a compression method to reduce the memory usage.

Y. Guo [10] introduces the concept of graphs as a data structure that is used a lot nowadays in various fields such as logistics and engineering. Graphs have been in widespread use and their applications and size keeps getting bigger. To efficiently process graphs there are various algorithms that use a lot of GPU processing power to perform the processing, some algorithms use CPU intensive distributed systems where as the third ones use both in conjunction. The authors have conducted extensive experiments for the evaluation of all these methods and have come to a conclusion that the GPU+CPU is the best approach currently.

C. Mayer [11] expresses the practice of processing graphs as they have been in extensive use in various applications ranging from their use in online games and social networking sites. Due to its nature, graphs have been increasing in size and can only processed by portioning the graph into several smaller pieces. The partitioning is not always optimal, therefore, the authors propose a technique called ADWISE for the smart partitioning of the graph for the optimal processing of the graph. Extensive evaluations have proved that ADWISE is a very powerful technique for the partitioning.

A. Zhou narrates that the usage of graphs has been the highest in history now and they are being extensively used in the applications of social media. Most social networks utilize graphs that are situated in geo-distributed centers, this is very challenging as there are a lot of calculations to take into consideration such as WAN usage and the partitioning methods. Therefore, to simplify the process the authors present a novel technique named G-Cut which can keep into consideration the WAN budget and also be able to partition the graphs with interdependent heterogeneous networks. [12]

Research Gap – The authors have kept the G-Cut, as it has been in the experimental stages, very limited and an inclusion of various other processing models for the graphs would make it a lot more versatile.

S. Pollard [13] explores the world of graphs and claims that most of the difficulties associated with them are very difficult to grasp and most developers cannot solve them outright. There are also very infeasible solutions that cannot be implemented and will always stay in the research phase, as the computing power that it would require would dwarf even supercomputers of today for a large enough graph. Therefore, the authors have developed a technique that can analyze the problems and solve it while being completely scalable to accommodate various graphs of different sizes.

Research Gap – The authors have provided a complete package that can be used in a number of applications, but the limitation faced is the lack of algorithms, addition of which would make this feasible.

J. Firoz [14] expresses the usage of graphs in a lot of major applications such as Social networking, health care sector and the online gaming industry. This has led to an extensive use of the graph technology everywhere. And as the graphs are getting bigger and extensive, the authors have proposed a technique for the scheduling and runtime support for the processing of the distributed graphs. This is due to the fact that there is quite a large overhead for the SPU when processing bigger graphs. This technique has been experimented on to prove that it is highly effective.

III. RESEARCH METHODOLOGY

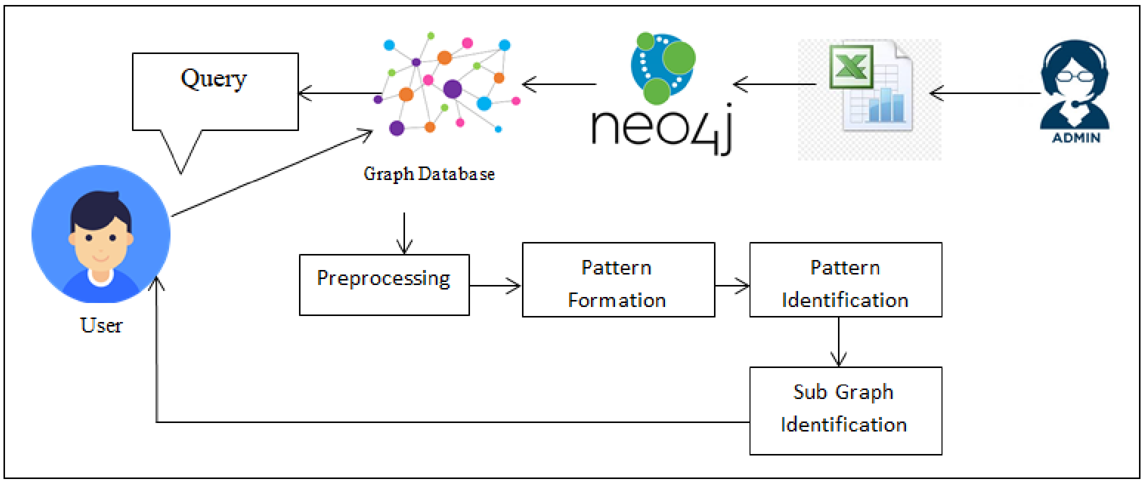


Figure 1: Research Methodology Overview

The above picture depicts the system architecture for the proposed system. The working of the system has been elaborated further in the section below.

Step1: Graph Base Creation -This is the first and initial step and is responsible for the data collection and creating the base of the graph. The system generates an Excel file by extracting usernames and the comments made from that username as dataset for the proposed model. This dataset is used further to generate a graph with the usernames as the vertex and the comments serve as the weights. This makes it easier to connect the relevant users together corresponding to their common weights resulting in a proper graph in neo4j.

Step 2: Query Pre-processing **-** The Graph database can be queried by the user if needed for the extraction of the relevant sub graph data. When fired, the query is processes by the pre-processing module which breaks the query string down to its basic component words with the help of the following activities.

1. *Sentence Segmentation:* Separates the source texts into a sentence and is responsible for the detection of boundaries.
2. *Tokenization:* Separates the query that is being fired by the user into individual words and added to the array.
3. *Stop Word Removal:* Stop Words are words which are used as fillers of conjunctions that serve to link different parts of a words (such as: for, an, the, is etc.). These words are not useful as they do not have much meaning and usage other than aesthetic and narration. Therefore, these stop words are deleted to decrease the computational overhead of the system.

4. *Stemming:* Stemming is the process of further decomposing the words in the query to their basic or the root form (like: going to go, coming to come etc.). This is particularly useful as most of the extensions on the root words do not provide an increased foresight into the meaning and would rather slow down the whole process by consuming a lot of energy. Therefore, stemming reduces the words into their root words for increased productivity.

Step 3: Pattern Creation- Here in this step the query words are subject to yield the word patterns, Where each of the query words are modulated into their combination words that eventually help to identify the proper semantic results. For this step proposed system uses the concatenation of the substring words into the basic word starts with the length of 3. For example national word can have patterns like { nat,nati,natio,nation,nationa,national}. This process can be more effectively depicted in algorithm 1.

ALGORITHM 1: PATTERN CREATION

```
//Input: Query Q
//Output: Pattern Set P
  0:Start
  1: Initialize  p=NULL
  1: String word[ ]=q.split( " ");
  2: FOR i=0 to Size of words
  3: W=word[i]
  4: FOR j=0 to length of W
  5: $sb_i$=substring($W_i$,2→j)
  6: Add $sb_i$ to P
  7: END FOR
  8: END FOR
  9: return P
  10: Stop
```

Step 4: Sub graph formation- Here in this step the fired query is being estimated in the graph database for the matched query tuples. And then for these matched tuples it forms a hyper graph semantically to provide the result more meaningfully. The hyper graph formation process can be depicted in the below algorithm 2.

ALGORITHM HYPER GRAPH FORMATION

```
//Input : Graph collection Set S ={$s_i$,$h_i$,$s_m$}
//Output: Hyper Graph G($s_i$,$h_i$,$s_m$)
Where
$S_i$ – From Attributes(node)
$h_i$ – To attributes(node)
$s_m$- Semantic
  0:Start
  1: Get the Set S
  2: FOR i=0 to Size of S
  3: Separate $s_i$, $h_i$ into List $L_s$,$L_h$
  4: END FOR
  5:Get unique elements form $L_s$ and$L_h$
  6: $N_s$=Size of $L_s$( Number of nodes for From attributes)
  7: $N_h$=Size of $L_h$( Number of nodes for To attributes)
  8: FOR i=0 to Size of $N_s$
  9: FOR j=0 to Size of $N_h$
  10: Identify the relational Edges E using $s_m$
  11: Form Graph G
  12: END FOR
  13: END FOR
  14: return G
  15: Stop
```

IV. RESULTS AND DISCUSSIONS

The proposed methodology has been extensively tested on a machine with a standard configuration such as a Core i5 processor having 4GB RAM. The machine was equipped with a Windows operating system with Java being handled with the Netbeans IDE and Neo4j performing the Graph database duties.

The various experiments that have been performed for the evaluation of the proposed methodology have been elaborated below.

Performance Evaluation Based on Precision and Recall

Precision and recall are one of the fundamental parameters for the evaluation of the performance of a system. Precision predicts the positive values that depict the amount of suitable information matched successfully. The precision can, therefore, be defined as the ratio of all the useful patterns matched from the user comments to the sum of all the relevant and irrelevant patterns. This process yields the actual effectiveness of the algorithm which is very effective in the evaluation of the proposed system.

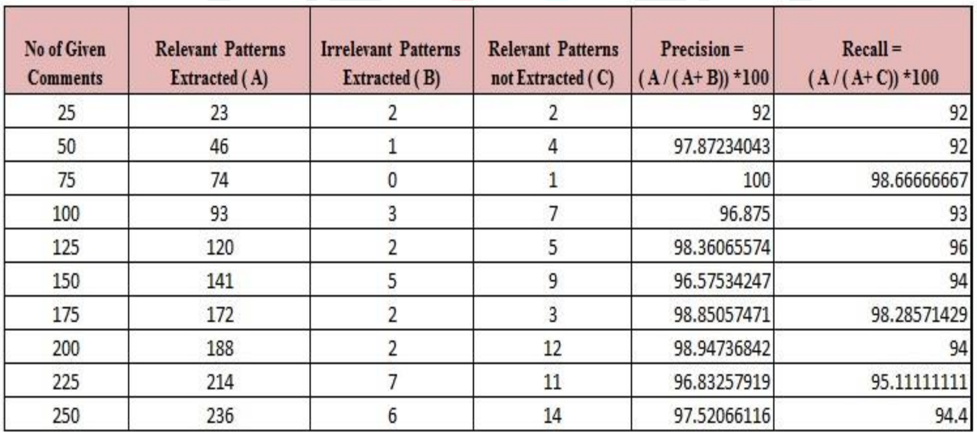
The recall is another one of the evaluation methods and is responsible for the indication of the relevant results matched versus the matched relevant results. Recall can be stated as the ratio of all the matched and relevant patterns versus all the patterns that did not match. Recall, therefore, can be defined as the indicator of Absolute Accuracy of the system being evaluated.

Precision can be accurately elaborated below

- A = The number of relevant patterns matched for the given number of comments
- B= The number of irrelevant patterns matched for the given number of comments
- C = The number of relevant patterns not matched for the given number of comments

So, precision can be given as

Precision = (A / (A+ B)) *100
Recall = (A / (A+ C)) *100



No of Given Comments	Relevant Patterns Extracted (A)	Irrelevant Patterns Extracted (B)	Relevant Patterns not Extracted (C)	Precision = (A / (A+B)) *100	Recall = (A / (A+C)) *100
25	23	2	2	92	92
50	46	1	4	97.87234043	92
75	74	0	1	100	98.66666667
100	93	3	7	96.875	93
125	120	2	5	98.36065574	96
150	141	5	9	96.57534247	94
175	172	2	3	98.85057471	98.28571429
200	188	2	12	98.94736842	94
225	214	7	11	96.83257919	95.11111111
250	236	6	14	97.52066116	94.4

Table 1: Precision and Recall Values

The experiments stated above were used to extensively evaluate the system. Various inputs were given as user comments to the system and their results were recorded. These values were then used to evaluate the Precision and Recall, as given in the above Table 1.

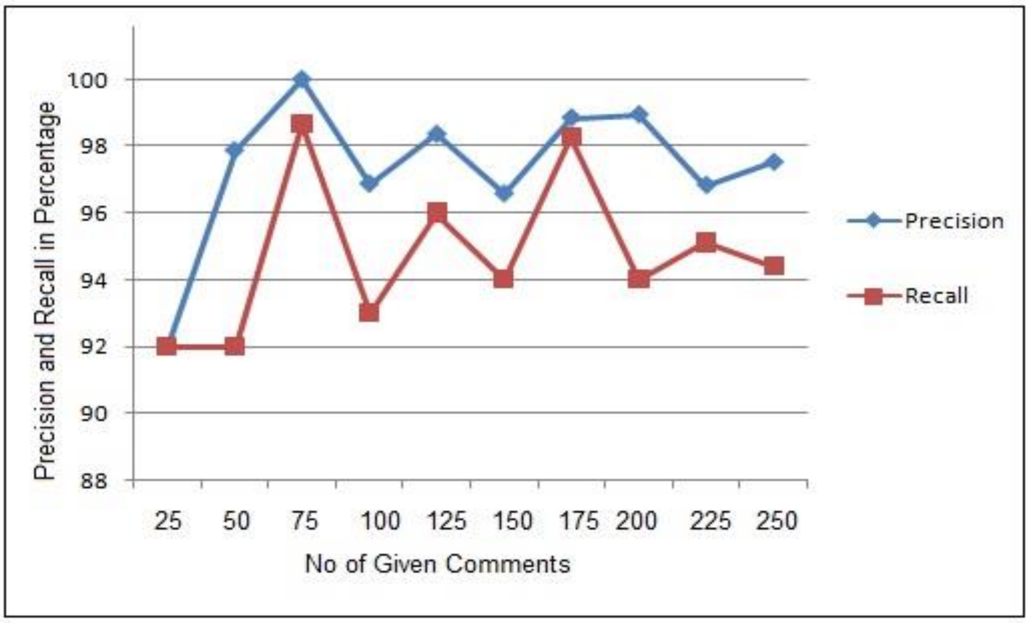


Figure 2: Performance Evaluation (Precision and Recall)

The graph in figure 2 depicts the output values of the performance evaluation of the system with the help of Precision and Recall. The values plotted on the graph clearly show that our proposed methodology is far superior as it manages to achieve 97.3% of precision in sub-graph matching andfor the given user comments the system manages 94.7% of average recall. These values are quite remarkable and are an indicator of the successful implementation of the proposed methodology.

V. CONCLUSION AND FUTURESCOPE

As we know the graphs are playing a vital role in the handling the interconnecting nodes for a given relationship. These kinds of graph data are extensively using in the online social networks to maintain the semantic relation between the users of the specific relational data. So this research paper analyzes all the past works on graph processing and try to evaluate the search gap in the most of the works. And finally this paper presents a work of graph handling and matching using hyper graph and pattern matching concepts which yields best accuracy as discussed in the section 4. This concept can be enhanced to work in the future by preparing the readymade API and this can deploy in the distributed paradigm too.

REFERENCES

[1] S. Au, A. Uta, A. Ilyushkin and A. Iosup, “An Elasticity Study of Distributed Graph Processing”, 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2018.

[2] Christian Mayer, Muhammad Adnan Tariq, Ruben Mayer and Kurt Rothermel, “GrapH: Traffic-Aware Graph Processing”, IEEE Transactions on Parallel and Distributed Systems, 2018.

[3] Tefeng Chen, Bo Li, “A Distributed Graph Partitioning Algorithm for Processing Large Graphs”, IEEE Symposium on Service-Oriented System Engineering, 2016.

[4] J. Wang and C. Zhang, “Analysis and Evaluation of the GAS Model for Distributed Graph Computation”, IEEE 37th International Conference on Distributed Computing Systems Workshops, 2017.

[5] Z. Li, S. Ren, S. Lu, J. Guo, W. Cai, Z, Qin, R. Siow and M. Goh, “Concurrent Hybrid Breadth-First-Search on Distributed Power Graph for Skewed Graphs”, IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS), 2018.

[6] Hao Long, Pingpeng Yuan, Hai Jin, Xiaofeng Ding, “Distributed Path Graph: A Cluster Centric Framework for Distributed Processing Graph”, IEEE 10th International Conference on Service-Oriented Computing and Applications, 2017.

[7] U. Cheramagalath, R. Nasre and Y. Srikant, “Falcon: A language for large-scale graph processing on Distributed Heterogeneous systems”, IEEE International Conference on Cluster Computing, 2017.

[8] Panagiotis Liakos, Katia Papakonstantinopoulou, and Alex Delis, "Realizing Memory-Optimized Distributed Graph Processing", IEEE Transactions on Knowledge and Data Engineering, 2017.

[9] G. Slota, S. Rajamanickam and K. Madduri, "A Case Study of Complex Graph Analysis in Distributed Memory: Implementation and Optimization", IEEE International Parallel and Distributed Processing Symposium, 2016.

[10] Y. Guo, A. Varbanescu, D. Epema and A. Iosup, "Design and Experimental Evaluation of Distributed Heterogeneous Graph-Processing Systems", 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, 2016.

[11] Christian Mayer, Ruben Mayer, Muhammad Adnan Tariq, Heiko Geppert,Larissa Laich, Lukas Rieger, and Kurt Rothermel, "ADWISE: Adaptive Window-based Streaming Edge Partitioning for High-Speed Graph Processing", IEEE 38th International Conference on Distributed Computing Systems, 2018.

[12] Amelie Chi Zhou, Shadi Ibrahim and Bingsheng He, "On Achieving Efficient Data Transfer for GraphProcessing in Geo-Distributed Datacenters", IEEE 37th International Conference on Distributed Computing Systems, 2017.

[13] S. Pollard and B. Norris, "Comparison of Parallel Graph Processing Implementations", IEEE International Conference on Cluster Computing, 2017.

[14] JesunSahariar Firoz, Marcin Zalewski, Andrew Lumsdaine and Martina Barnas, "Runtime Scheduling Policies for Distributed Graph Algorithms", IEEE International Parallel and Distributed Processing Symposium, 2018.

*****