2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing

NSLPA:A node similarity based label propagation algorithm for real-time community detection

Qi Song, Bo Li, Weiren Yu, Jianxin Li, Bin Shi
School of Computer Science and Engineering
Beihang University
Beijing, 100191, China
Email: {songqi, libo, yuwr, lijx, shibin}@act.buaa.edu.cn

***Abstract*—With the development of Internet, online social networks and websites generate a large amount of data. At the same time, several distributed systems, represented by Hadoop, has been proposed to handle mass data. These systems provide both efficient and convenient way to construct different kinds of algorithms. Community detection, a traditional research area, is now facing the challenge of Big Data. Draw support from a powerful distributed graph processing system, GraphLab, we redesign and implement several classical community detection algorithms using very large real-life datasets. Using node similarity parameter $AdjPageSim$, we propose a new community detection algorithm based on label propagation, namely NSLPA. Experiments and benchmarks reveal that several quite powerful algorithms perform bad in distributed environments. However, NSLPA is not only faster but more accurate compared with other community detection algorithms. NSLPA can process a graph with 60 million nodes and 2 billion edges in less than 1000 seconds with a relatively high accuracy.**

***Keywords*—community detection, GraphLab, LPA, PageSim**

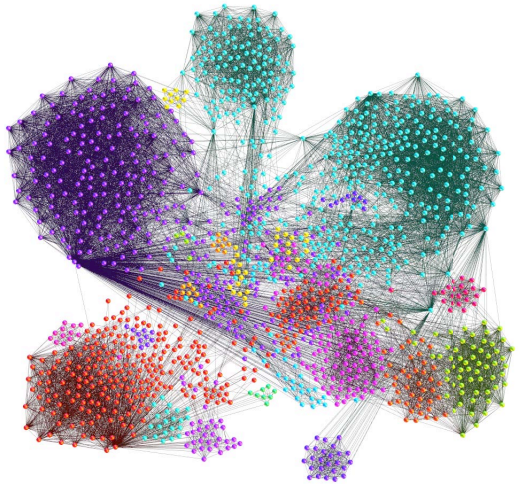

Fig. 1. Community structure of a small Weibo dataset

I. Introduction

Recent years have seen the flourishing of Big Data, which can be used to describe a very large scale data set with complicated features.Existing database management systems can not handle this kind of data very well. The significance of Big Data has attracted the concern of governments, companies and scientific institutions. Big data contains online social networks and webdata like Facebook, Twitter, Weibo, World Wide Web and Wikipedia. These data can be represented as gragh: nodes denote persons or pages, while edges represent the relationship between person or page nodes. This relationship can be represented as follow in facebook or hyperlinks in WWW. Since the nature of the data, this kind of online graph data contains an immense number of person nodes which are sparsely connected, namely, they constitute several communities.

Detecting community structures from online graph data is an important function in data analysis. It can be used to analyze the information propagation or even devise business strategies. Graphically, communities are characterized by a group of nodes which are densely connected by internal edges. From Figure 1, we can see there exist several densely connected communities in a small Weibo dataset [1]. However, no definition of community is universally accepted. [1] provides several kinds of definitions like clique and quasi-clique. Different kinds of community algorithms have been proposed and some of them can effectively detect the community structure. Bad usability refers to large scale online graph data is one of the most critical problems for community detection algorithms. On one hand, as community detection is a NP-hard problem, some algorithms are quite time consuming, thus its performance in single machine is intolerant. On the other hand, it lacks a convenient and efficient way to implement a community detection algorithm. Since distributed data processing model like MapReduce [24] provides a basic idea to solve large scale data processing problems, we can intuitively use these systems to implement such community detection algorithms.

Efficient processing of large graph is challenging as graph algorithms exhibit very little work per node. Moreover, traverse the graph may cause poor locality of memory access due to discontinuous node storage. Distributed processing may exacerbate the locality issue and introduce large amount of message passing. As the MapReduce framework is disk-oriented and graph structure unaware, researchers proposed several graph processing systems like Pregel [8], GraphLab [9] and GraphX [15]. These systems provide several APIs for developers to build graph

[1] Sina Weibo: http://weibo.com

978-1-4799-7881-6/14 $31.00 © 2014 IEEE
DOI

algorithms. The greatest advantage lies in liberating developers to consider bottom details, like message passing, task allocation, resource manage and error recovery. In this paper, we choose GraphLab as our programing platform mostly because of its high performance brought by deeply optimizing.

In this paper, we first implement several community detection algorithms in GraphLab. Then, dealing with the random allocation problem, we use node similarity to improve the basic LPA algorithm. The major contributions of this paper are three-fold:

1) Besides basic label propagation algorithm(LPA) [18] provided by GraphLab in its graph processing toolkits, we implement Kernighan-Lin algorithm [3], Blondel's modularity based algorithm [16] and coefficient based algorithm [17] in GraphLab. Further experiments are performed to compare their performance.
2) We propose a modified label propagation algorithm which use node similarity to choose labels instead of random allocate. Random walk based PageSim is used to evaluate node similarity which can effectively leverages the capability of GraphLab.
3) Both accuracy and execution time are used to compare the performance of four algorithms and NSLPA. We use several real-world labeled datasets as input. A statistic parameter called community similarity is defined and used to evaluate the detection accuracy. Furthermore, we provide the community-size distribution of these datasets as an intuitive method to show the result.

The rest of this paper is organized as follows. Related works are reviewed in Section II. An introduction of four algorithms is given in Section III. An analysis of LPA and the detail of NSLPA are described in Section IV. Detailed experiment and comparison analysis are shown in Section V. A brief conclusion is given in Section VI.

II. Related Work

Detailed survey and comparison of the development of community detection can be found in [1] [21]. As community structure detection is known as an NP-hard problem [2], many effective but not perfect methods have been proposed. These methods can be divided into several different types. Divisive methods contains calculating the edge centrality and removing those edges possibly lying between communities. Edge betweenness [4], current-flow betweenness [4], random-walk betweenness [4] are proposed to define edge centrality. Modularity-based methods rely on modularity proposed by Newman [5].

In addition to the methods mentioned above, label propagation (LPA) is one of the fastest methods for community detection. However, the basic propagation algorithm holds some critical problems, such as the fluctuation of results because of randomness [19]. Thus there exists some modification to improve the efficiency and accuracy of LPA. [20] suggests adding a score associated with the label which decreases as it traverses from its origin and [19] use node balancer to accessing the nodes in some predefined order to enhance the robustness. [22] employs a multistep greedy agglomerative algorithm that can merge multiple pairs of communities at a time. [23] presents a semisynchronous version of LPA which aims to combine the advantages of both synchronous and asynchronous models.

In recent years, distributed data processing systems are widely used to effectively process large number of data. Hadoop is one of the most widely used systems. However, as MapReduce is not suitable for graph processing [6], researchers proposed different kinds of graph processing systems. These systems aim to build a high performance and convenient way to implement graph algorithms. Some critical aspects should be considered, includes: architecture, I/O bottleneck, special requirements of different kinds of graph and algorithms, and stream data. Trinity [7] is a distributed graph storage and (SPARQL) querying system. In addition to GraphLab [9], one of the most highly optimized systems, Pregel [8] and GraphX [15] are also representative systems. Pregel is a distributed graph system based on message passing and Bulk synchronous protocol (BSP). Implemented on top of Spark, Graphx unifies graph-parallel and data-parallel computation and is able to efficiently executing the entire graph analytics pipeline. Even the performance is not that good compared with GraphLab, GraphX provides an efficient way to implement graph algorithms, for example, we can implement the PowerGraph and Pregel abstractions in less than 20 lines of code. Based on GraphLab, PowerGraph [10] improves the parallel performance for natural graph. Other distributed graph processing systems include Giraph [11], GPS [12], Mizan [13], SPAR [14].

III. Selection of Algorithms

In this section, we'll give a brief introduction about the four algorithms implemented upon GraphLab.

A. Kernighan-Lin algorithm(K-L)

Kernighan-Lin algorithm [3], scaling as $O(n^2 logn)$, is one of the earliest attempt to solve the graph partitioning problem. The original algorithm can only partition a graph into two disjoint subsets. The algorithm starts by dividing the nodes into two subsets in any way. Then, for each pair (i,j) of nodes such that i and j lies in different groups, the algorithm calculates the cut size of interchanging i and j. Among all pairs (i,j), we swap a pair of nodes which reduces the cut size by the largest amount or increases it by the smallest amount. This process is then repeated until there are no pairs left to be swapped. Each node in the network can only be moved once.

We can divide a graph into more than two pieces by repeating the progress. Using some other algorithms as the starting configurations, this method can achieve better performance thus is treated as an improvement of other algorithms.

B. Blondel's modularity-based algorithm(BA)

Blondel proposed in [16] a heuristic method based on modularity optimization. It is a quite fast algorithm with time complexity $O(m)$ but is limited by storage demands. This method consists of a sequential sweep over all nodes. We assign a different community to each node first. Then for each i we consider the neighbours j of i and evaluate the gain of modularity that would take place by removing i from its community to the community of j. The node i is then placed in the community for which this gain is maximum, but only if this gain is positive. If no positive gain is possible, i stays in its original community. This process is applied repeatedly for all nodes until no further improvement can be achieved. This method can also detect hierarchy using a further step.

C. coefficient-based algorithm(C-B)

Algorithm proposed by [17] leverages the clustering coefficient, which is defined as

$$\widetilde{C}_{i,j}^{(g)} = \frac{z_{i,j}^{(g)} + 1}{s_{i,j}^{(g)}} \tag{1}$$

where i and j are the extremes of the edge, $z_{i,j}^{(g)}$ the number of cycles of length g(here we choose $g = 3$ which means triangles). The number of actual cycles in the numerator is augmented by 1 to avoid the coefficient $\widetilde{C}_{i,j}^{(g)}$ to be 0. At each iteration, the edge with smallest clustering coefficient is removed and the measure is recalculated again. The algorithm stops when all clusters produced by the edge removals are communities in the strong sense, and further splits would violate this condition. A strong community, or *LS_set*, is a subgraph with the internal degree of any node of the community exceeds the number of edges that the node shares with any other community.

D. Label propagation algorithm(LPA)

Label propagation [18] is a simple and fast community detection method. Each node is initially given a unique label which are their node id. At each iteration, a sweep over all nodes is performed in random sequential order: each node takes the label shared by the majority of its neighbors. If there is no unique majority, one of the majority labels is picked at random or the node keeps its label unchanged. The process reaches convergence when each node has the majority label of its neighbors.

These four algorithms are implemented using GraphLab API and the detailed performance comparison is shown in section V.

IV. NODED SIMILARITY BASED LPA

Label propagation is a near linear algorithm while accommodate to GraphLab very well. However, in order to achieve such a linear complexity, basic LPA sacrifices some stability, namely introduce random propagate. This kind of randomness will cause some disaster consequence, especially in very large networks. In this section, we'll first give a biref analysis of LPA and its randomness. Based on the analysis, a node similarity based label propagation algorithm is proposed which can fully leverages the computing capability of GraphLab.

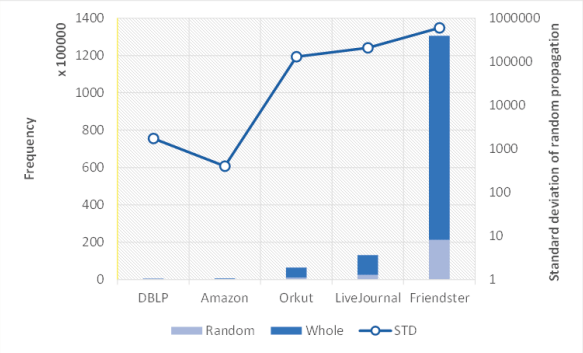

Fig. 2. Statistics of LPA

A. Analysis of label propagation algorithm

Let the network be represented by a simple undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. The procedure of label propagation is quite simple (III-D). Alternatively, an equivalent mathematical formulation can be given as

$$l'_v = argmax_l \sum_{u \in \sigma(v)} \delta(l_u, l) \tag{2}$$

Where l_u is the label for node u, l'_v is the new label for node v, $\sigma(v)$ is the set of nodes neighboring v in the graph, and δ is the Kronecker delta. If there exists multiple labels, the result of argmax should be taken as for the procedural description lf LPA, i,e, take a label at random that satisfies Eq.2.

Label propagation with asynchronous updating randomly accesses the nodes and randomly chooses a label while there exists multiple max-number labels. On one hand, randomly access hampers the robustness of the algorithm. Thus some constraints are used to predefine the access order [19] [20]. On the other hand, random choose may also affect the iteration time and accuracy of final result. We have conducted several experiments to calculate the number of random choose in LPA using five datasets choosed from Table I.

Figure 2 shows the evaluation results. As LPA is not a deterministic algorithm, we run each dataset ten times and calculate the standard deviation of random propagation frequency. Obviously, there exists a lot of random choose in LPA , most of these random propagation is observed in the first few iterations. Furthermore, as the size of dataset is getting larger and larger, the stability of LPA decreases. This phenomenon can be explained by the complexity and magnitude of large dataset. As random propagation brings uncertainty, LPA may produce quite different results in each evaluation. Moreover, another disadvantage of LPA is that it may sometimes result in a single large community.

We further use a toy example network in Figure 3 to illustrate the consequence caused by random propagation.

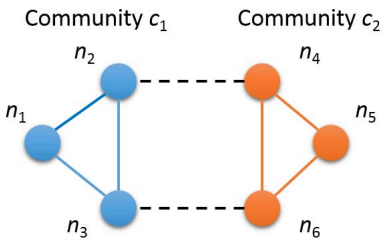

Fig. 3. Toy example network with two communities. Dashed links represent inter-community edges and nodes colors indicate their community labels

The network consists of two communities c_1 and c_2, which can be distinguished by that each node has more intra-community than inter-community edges. When it turns to n_2, it will adopt the label of either n_1, n_3 or n_4 randomly. If it choose n_1 or n_3 then at the end of this iteration, all nodes in community c1 will be labeled with the same label (that initially belongs to node n_1 or n_3). The outcome thus corresponds to the natural community of the network. But if n_2 choose n_4 as its label, and then n1 may also choose n_4 label. As a result, it is straightforward to see that all nodes in the network will be classified to the same community c_2. This is the reason why LPA may converges into a very large community. One effective way to avoid this is to accessing the nodes in some predefined(deterministic) order, namely placing higher propagation preference to the nodes that are updated first and lower propagation preference to the nodes that are updated last. But considering the computing characteristic of GraphLab, we can use a more powerful method to avoid random propagation and make the procedure deterministic.

B. The basic idea

Intuitively, using node similarity to choose label from a closer node when there exists multiple nodes with the same maximal label can avoid the random propagation. In Figure 3, it is obviously that n_1 or n_3 is closer to n_2 than n_4 does. Thus we can force n_2 choose n_1 or n_3 instead of n4 to avoid resulting in a single community.

As GraphLab provides an effective way to run random walk algorithms like PageRank, we could use this feature to introduce a fast method to calculate node similarity, namely *PageSim* [27].Based on PageSim, a node similarity based label propagation algorithm(NSLPA) is proposed.

The original definition of $PageSim$ calculate the similarity of every node pair. However, in NSLPA, we only consider the adjacent nodes. A mutation of $PageSim$, namely $AdjPageSim$ is defined as follows :

DEFINITION 1. *Let $PR(v)$ denotes the **PageRank score** of node v $PR(v) = (1-\gamma) + \gamma \sum_{u \in in(v)} \frac{PR(u)}{|out(v)|}$, for $v \in V$. The parameter γ is a damping factor which is usuallt setted to 0.85. For undirected graph, $PR(v)$ is calculated by treating the graph as a directed graph, namely, makeing each edge bidirectional. Let $PG(u,v)$ denotes the PageRank score that node u propagates to node v, as u and v are adjacent, $PG(u,v) = PR(u)$, where $u, v \in V$.*

DEFINITION 2. *Let $\overrightarrow{SV}(v)$ denotes the **similarity vector** of node v , we have $\overrightarrow{SV}(v) = (PG(v_i, v))^T, v_i \in \delta(v)$ contains all adjacent nodes of v. Let $APS(u,v)$ denotes the **adjacent pageSim score** of page u and v, $APS(u,v) = \sum_{j=1}^{n} min(PG(v_j,u), PG(v_j,v))$, where $u, v \in V$ and $v_j \in \delta(v) \cap \delta(u)$*

Based on $AdjPageSim$, the updating rule of the NSLPA is thus rewritten into

$$l'_v = argmax_l \sum_{u \in \sigma(v)} \delta(l_u, l) APS(u,v) \tag{3}$$

C. The design and implementation of NSLPA algorithm

The main steps of NSLPA include two steps: initialization and label propagation. In the initialization step, NSLPA first calculate the PageRank value of each node and then the APS value of each node with their adjacent nodes. These APS values are then used in label propagation step to help choosing labels. Initialization calculates APS value for each node. As the algorithm is implemented based on GraphLab, the GAS model must be followed. Two GAS circles are used, the first one calculates PageRank value for each node and the second one calculate the final APS value for each node with their adjacnet nodes. After the initialization, each node now owns the similarity value with each of its adjacent node. Propagation procedure is almost the same compared with basic LPA, the only difference is use APS velue to select labels when there exists multiple majority labels.

V. Experiment

We proceed by evaluating the performance of four algorithms described in section III and NSLPA proposed in section IV. Several large datasets are used to examine the accuracy and execution time of these algorithms.

A. Experiment Setup

All algorithms are implemented in GraphLab2.2 which is deployed in an 8 node cluster. Each node has two 2GHz Intel Xeon E5-2650 CPUs and 256 Gb memory.

B. Datasets

We choose five networks with ground-truth communities from SNAP Datasets [2] as samples. [26] gives some detailed description of communities in these six datasets which is briefly shown here in Table I. N means the number of nodes while E the number of edges. C represents the number of communities and S is average community size, these two values are determined by the algorithm proposed in [26].

[2]Datasets are available at http://snap.stanford.edu

Dataset	N	E	C	S
Friendster	65,608,366	1,806,067,135	1.5M	26.72
Orkut	3,072,441	117,185,083	8.5M	34.86
DBLP	317,080	1,049,866	2.5k	429.79
LiveJournal	3,997,962	34,681,189	310k	40.46
Amazon	334,863	925,872	49k	99.86

TABLE I. DATASET STATISTICS

C. Evaluation metrics

We consider both execution time and accuracy of different algorithms. A key problem in community detection is the termination condition of iterations. As K-L algorithm is basically a dichotomic method, we must manually set the iteration time if we want to detect more than two communities. Using the data in Table I, we set K-L algorithm to run several times to receive almost the same number of communities. Coefficient based algorithm owns the ability to converge itself by checking if all communities are strong communities. LPA and its derivative algorithms does not need any constraints as it can converge eventually. Each algorithms is performed 10 times and the average value is choosen to be the result.

- **Execution Time**

Performance is easy to evaluate by recording the execution time. Note that the whole execution time contains reading files from HDFS and finalizing graph, which will be neglected as we only care about the pure execution of the graph engine.

- **Accuracy**

However, how to check the accuracy of communities detected is a difficult question, especially for very large datasets. With the development of visualizer, it is convenient to see the actual community structure in small datasets. But for large datasets, the only approximate solution is counting the stastistics value of the result. As SNAP provides ground-truth communities, we can use a more accurate value instead of modularity to evaluate the accuracy of different algorithms. We define a community similarity parameter as:

$$CS = \frac{\sum_m \delta(c_i, c_j)}{\sum_m \binom{n_m}{2}} \tag{4}$$

where m is the number of ground-truth communities C_m, c_i and c_j are any pair of nodes, n_m is the size of C_m. The δ function yields 1 if c_i and c_j belongs to the same community which is detected by the algorithm. This parameter enables us to check if any pair of intra-community nodes in the result really belongs to the ground-truth community. Larger CS value means the algorithm is more accurate. If the result is completely overlapped then CS is equals to 1. However, this is an approximate parameter because we do not consider if one detected community contains several small ground-truth communities. As there exists a lot of small ground-truth communities which can have little compact in whole community structure, this approximation is acceptable.

D. Evaluation results

- **Execution Time**

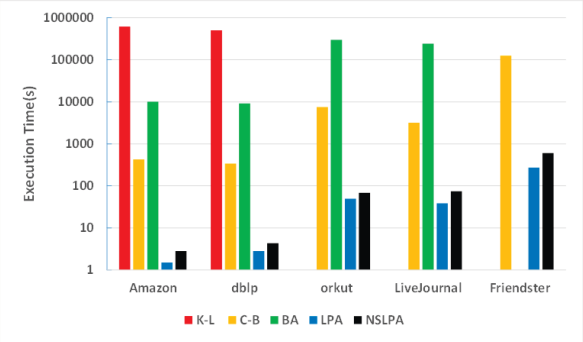

Fig. 4. Execution time

Dataset	K-L	C-B	BA	LPA	NSLPA
Amazon	0.751	0.833	0.676	0.612	0.812
DBLP	0.789	0.854	0.801	0.654	0.824
Orkut	-	0.798	0.699	0.628	0.807
LiveJournal	-	0.754	0.701	0.615	0.793
Friendster	-	0.711	-	0.691	0.761

TABLE II. CS VALUE

Figure 4 shows the result of execution time. K-L algorithm is obviously the slowest one as it has the largest time complexity. Building upon a distributed system, checking all pairs of nodes is quite time consuming as the system must coordinate nodes in different workers. Even GraphLab is able to calculate coefficient effectively, determing the termination condition consumes most of the execution time.However, even the time complexity of Blondel's algorithm is $O(m)$, it performs bad in GraphLab. The reason also results from the distribution of data. Calculating ΔQ requires sweeping over all nodes, which introduces a great many of inter-worker computation.

LPA remains the fastest algorithm, which can process mora than 60 million nodes in less than 300 seconds. As choosing a label only relies on the adjacent nodes' labels, LPA does not need extra computation in a distributed environment. GraphLab places nodes (main nodes) in different workers and keeps their adjacent nodes (ghost nodes) in the same worker. After each iteration, different workers exchange data for the next iteration. LPA is fairly closely to the GraphLab model, which ensures its performance in distributed environment. As NSLPA needs to calculate the pagerank and APS value, it consumes more time than original LPA. But this extra time does not cause a dramatically increase. NSLPA is still much faster than other algorithms and can handle Friendster in less than 1000 seconds.

- **Accuracy**

Table II shows the result of CS value which represents the difference between detected and ground-truth communities. Some large datasets exceeds the capability of some algorithms, thus the result is shown as "-" in the table. K-L, C-B and BA performs well in relatively small datasets. However, with the increase of complexity and scale, they can not ensure their accuracy. Furthermore, from Figure 1, we can see that there exists some nodes which are hard to define their status. They may belong to more than one communities simultaneously, or remains

relatively idependent. For large social networks, these nodes are common as they represent some zombie account or some sociable guys belong to different communities in different areas. This is the main deviation for detected communities.

Even performs better in execution time, the accuracy of LPA is the worst. Because of the randomness introduced above, LPA's results experiences a fluctuation. Thus the CS value is lower than that of other algorithms. Due to the node similarity, NSLPA performs much better than LPA. Even its result is not perfect, it achieves almost the same or even better accuracy than other algorithms. This enables NSLPA to be a powerful community detection method with both high efficiency and accuracy.

VI. Conclusion and Future Work

Community detection is a useful method to reveal the structure of social network. With the development of distributed graph processing systems, we now have the ability to detect communities in very large datasets.

In this paper, we have built several community detection algorithms upon GraphLab and evaluated their performance. Several classical algorithms, like Kernighan-Lin algorithm, Blondel's modularity based algorithm and coefficient-based algorithm, experienced bad performance in the evaluation. They are not quite Graphlab's style as distribution introduced large amount of extra message passing. However, one of the fastest method, LPA, performed very fast in the distributed environment. But the randomness can result in an inaccurate result. As calculating PageRank value is efficient using GraphLab, we define a value called $adjPageSim$ to help nodes to choose labels during the propagation. NSLPA can effectively increase the accuracy of LPA while preserve its low time complexity.

However, NSLPA remains a very simple attempt in solving community detection problem for large datasets. Our future work includes further algorithm improvement, how to measure the effect of partition method in such algorithm and how to evaluate the accuracy of CD algorithms in very large datasets.

Acknowledgment

This work is supported by China MOST project (No. 2012BAH46B04), 863 project (No.2013AA01A213), and SKLSDE-2014ZX-04.

References

[1] Fortunato S. Community detection in graphs[J]. Physics Reports, 2010, 486(3): 75-174.

[2] Brian Karrer, Elizaveta Levina, and M. E. J. Newman. Robustness of community structure in networks. Physical Review E (Statistical, Nonlinear, and Soft Matter Physics), 77(4), 2008

[3] Kernighan B W, Lin S. An efficient heuristic procedure for partitioning graphs[J]. Bell system technical journal, 1970, 49(2): 291-307.

[4] Girvan M, Newman M E J. Community structure in social and biological networks[J]. Proceedings of the National Academy of Sciences, 2002, 99(12): 7821-7826.

[5] Newman M E J, Girvan M. Finding and evaluating community structure in networks[J]. Physical review E, 2004, 69(2): 026113.

[6] Guo Y, Biczak M, Varbanescu A L, et al. Towards benchmarking graph-processing platforms[J]. Poster at Supercomputing, 2013.

[7] Shao, B., Wang, H., Li, Y. (2013, June). Trinity: A distributed graph engine on a memory cloud, In SIGMOD 2013: pp. 505-516.

[8] Grzegorz Malewicz, Matthew H. Austern, Aart J. C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In: SIGMOD 2010. 135-146.

[9] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. GraphLab: A new parallel framework for machine learning. In Proc. Conf. Uncertainty in Artificial Intelligence, UAI '10, July 2010.

[10] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin. PowerGraph: distributed graphparallel computation on natural graphs. In OSDI '12: pp 1730, 2012.

[11] https://giraph.apache.org

[12] Salihoglu S, Widom J. Gps: A graph processing system[C]//Proceedings of the 25th International Conference on Scientific and Statistical Database Management. ACM, 2013: 22.

[13] Khayyat Z, Awara K, Alonazi A, et al. Mizan: a system for dynamic load balancing in large-scale graph processing[C]//Proceedings of the 8th ACM European Conference on Computer Systems. ACM, 2013: 169-182.

[14] Pujol, J. M., Erramilli, V., Siganos, G., Yang, X., Laoutaris, N., Chhabra, P., Rodriguez, P. (2011). The little engine (s) that could: scaling online social networks. ACM SIGCOMM Computer Communication Review, 41(4), 375-386.

[15] Xin R S, Crankshaw D, Dave A, et al. GraphX: Unifying Data-Parallel and Graph-Parallel Analytics[J]. arXiv preprint arXiv:1402.2394, 2014.

[16] Blondel V D, Guillaume J L, Lambiotte R, et al. Fast unfolding of communities in large networks[J]. Journal of Statistical Mechanics: Theory and Experiment, 2008, 2008(10): P10008.

[17] Radicchi F, Castellano C, Cecconi F, et al. Defining and identifying communities in networks[J]. Proceedings of the National Academy of Sciences of the United States of America, 2004, 101(9): 2658-2663.

[18] Raghavan U N, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks[J]. Physical Review E, 2007, 76(3): 036106.

[19] ubelj L, Bajec M. Robust network community detection using balanced propagation[J]. The European Physical Journal B-Condensed Matter and Complex Systems, 2011, 81(3): 353-362.

[20] Leung I X Y, Hui P, Lio P, et al. Towards real-time community detection in large networks[J]. Physical Review E, 2009, 79(6): 066107.

[21] Lancichinetti A, Fortunato S. Community detection algorithms: a comparative analysis[J]. Physical review E, 2009, 80(5): 056117.MLA

[22] Liu X, Murata T. Advanced modularity-specialized label propagation algorithm for detecting communities in networks[J]. Physica A: Statistical Mechanics and its Applications, 2010, 389(7): 1493-1500.

[23] Cordasco G, Gargano L. Community detection via semi-synchronous label propagation algorithms[C]//Business Applications of Social Network Analysis (BASNA), 2010 IEEE International Workshop on. IEEE, 2010: 1-8.

[24] Jerey Dean and Sanjay Ghemawat, MapReduce: Simplied Data Processing on Large Clusters. in Proc. 6th USENIX Symp. on Operating Syst. Design and Impl., 2004, 137-150.

[25] Mislove A, Marcon M, Gummadi K P, et al. Measurement and analysis of online social networks[C]//Proceedings of the 7th ACM SIGCOMM conference on Internet measurement. ACM, 2007: 29-42.

[26] Yang J, Leskovec J. Overlapping community detection at scale: a nonnegative matrix factorization approach[C]//Proceedings of the sixth ACM international conference on Web search and data mining. ACM, 2013: 587-596.

[27] Lin, Zhenjiang; King, I.; Lyu, M.R., "PageSim: A Novel Link-Based Similarity Measure for the World Wide Web," Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on , vol., no., pp.687,693, 18-22 Dec. 2006