Identification of Key Nodes by Algorithm of Betweenness Centrality Based on Traversal of Spark Graph

Hongye Xue[1, a], Tianmei Li[1, b] and Xiangyu Luo[1, c]

[1]College of Computer Science and Technology Xi'an University of Science and Technology, Xi'an, China;

[a]xuehy60@qq.com, [b]1282093489@qq.com, [c]lxyseu@gmail.com

Abstract. The betweenness center algorithm, which relies on the global network topology, can effectively identify the key nodes of the network. In the existing algorithm for accurately calculating betweenness centrality, the algorithm proposed by Ulrik brands based on the backtracking of graph is recognized as the fastest running, and its running time is still difficult to accept. The peer proposes approximate calculation and parallel computation of the betweenness Center to shorten the running time, and the running results and time are still difficult to accept. For this reason, this paper proposes an algorithm of betweenness obtains the total number of shortest paths in the network and the number of bars each node appears in the middle of the shortest path through the traversal of the graph, and then accurately calculates betweenness centrality of the nodes by using the formula of betweenness centrality, and implements the algorithm on the platform of Spark GRAPHX distributed graph processing. Experiments show that, this method is consistent with the method proposed by Ulrik brands to identify key nodes, the running time is halved, and it is suitable for large-scale networks.

Keywords: Global network topology, the backtracking of the graph, betweenness centrality, the traversal of graph, the platform of Spark GRAPHX distributed graph processing.

1. Introduction

There are a large number of complex systems in the real world that can be described through a variety of networks, such as the Internet, power networks, viral networks, network of criminal relationships, rumors communication networks, and so on. In the basic research work of complex networks, it is of great significance to identify the key nodes of the network, to analyze their properties in a targeted way, and to make use of them[1].For example, in infectious disease networks, the source of infection can be effectively detected, and isolate it first and then take measures to treat it can prevent the further spread of the disease[2].In the power network, it is possible to quickly discover and protect the important power generation or transmission units in the network, which can effectively reduce power outages or avoid other power accidents[3].In a large-scale internet, it is possible to identify key gateways or server nodes, protect them, and thus improve the resistance of the network to the destruction. In addition, it is of great significance to accurately identify the key nodes in various specific networks, such as transportation network, scientific research cooperation Network and food chain network.

The betweenness centrality of a node quantifies the ability of a node to serve as a bridge on the shortest path between other nodes, depicts the influence of one user on the communication between other users in the social network, and is an important index to characterize the importance of network nodes, which is widely used to identify key nodes of the network. In recent years, a number of outstanding achievements in the calculation of betweenness centrality have emerged[4-9].Traditional algorithms for calculating betweenness centrality, such as the Floydwarshall algorithm [10], have a time complexity of $o(m^2 * n)$, a time complexity of $o(n^3)$ in a sparse network, and a high time complexity (where M is the number of edges in the network and n is the number of nodes in the network).In 2001, Brandes proposed an algorithm to quickly compute the betweenness centrality of nodes based on graph backtracking[5],with a time complexity of $o(mn)$ on graphs without direction and weight, and a time complexity of $o(mn+n^2+log\ n)$on Weighted networks. In 2006, Zhang

Guoqiang proposed a method based on backtracking to calculate betweenness centrality of nodes for simple powerless networks， with a time complexity of *o(mn)* [11].In 2008, Kintali proved that the lower bound of the time complexity of the algorithm for calculating betweenness centrality of nodes is *o(mn)* [12].

Because of the long running time required by the existing algorithm to compute the intermediary center of nodes on large-scale diagrams, relevant researchers at home and abroad have proposed in recent years to solve the above problems by approximate calculation and parallelization calculation, and shorten the calculation time.

In the aspect of approximate calculation, sariyüce and others[6] proposed a strategy to split and compress the graph, by splitting the complex network graph into several simple sub-graph, and merging many nodes with equivalent functions, greatly simplifying the original network, thus speeding up the calculation of the betweeenness centrality. Deng Xiaolong [13] proposed a novel approximate calculation method of the betweenness centrality of nodes based on the Axis node selection strategy. Yang Jianxiang[14] introduced the method of calculating the local betweenness centrality of nodes, which approximate calculates the betweenness centrality of nodes of the large graph by calculating the sub-graphs with distance the node 5 to 9 hops. Although these methods can shorten the time of computing betweenness centrality of nodes, the approximate results deviate greatly from the exact calculation results. In this paper, an algorithm based on graph width traversal is proposed to realize the accurate calculation of the betweenness centrality of nodes.

In parallel computing, it is mainly based on the two categories of MPI and MapReduce to realize the parallel calculation of the algorithm of betweenness centrality of nodes. In 2006, Bader implemented the algorithm proposed by Brandes on MPI, processed a graph of 3 million nodes and 16 million edges on the 16*1.9GHZ processor, 25G shared memory, and spent 42 minutes completing the betweenness centrality of nodes[15];In 2010, Yang Cheng constructed a social network analysis tool XRIME based on Hadoop, through the parallelization of Brandes proposed algorithm on MapReduce to complete betweenness centrality of nodes[16].But the calculation time of these methods is still difficult to accept. For this reason, this paper uses spark GRAPHX based on memory calculation to realize the parallelization of the algorithm proposed in this paper.

Experiments show that key nodes identifyed by using the method proposed in this paper are consistent with key nodes identifyed by using Brandes proposed the method based on Graph backtracking, and the calculation time is shortened by half. And the method proposed in this paper is implemented on spark GRAPHX and can be applied to large networks.

2. Related Knowledge

2.1 Betweenness Centrality.

The concept of benweenness centrality stems from the analysis of the importance of individuals in social networks. The value of betweenness centrality of a node i represents the proportion of the number of shortest path through the node i account for the shortest path between all node pairs, so it can depict the importance of the node of the network. The value of betweenness centrality of any node k is defined as follows [17]:

$$B(k) = \sum_{i,j(i\neq j)} \frac{\lambda_k(i,j)}{\lambda(i,j)} \tag{1}$$

Where $\lambda(i,j)$ is the number of shortest path between any two nodes i and j in the graph, $\lambda_k(i,j)$ is the number of bars passing node k in the shortest path between node i and node j.

2.2 Spark GraphX.

Spark is a distributed, memory-based, universal computing framework that is ideal for iterative computing [18]. GraphX is a distributed Graph computing framework combining graph parallelism and data parallelism, which makes it easy for users to implement graph algorithm on spark. The platform of spark GRAPHX distributed processing is described below [19].

2.2.1 Spark.

Spark is a common parallel computing framework similar to Hadoop MAP reduce, an open source for UC Berkeley AMP lab. Spark based on the map reduce algorithm to achieve distributed computing, with the advantages of Hadoop map reduce, but different from map reduce is the job intermediate output and results can be saved in memory. Spark comes with more than 80 operators to take full advantage of memory computing to achieve real-time processing of massive amounts of data, and its program uptime is nearly hundreds of times shorter than Hadoop mapreduce, as shown in Fig. 1.

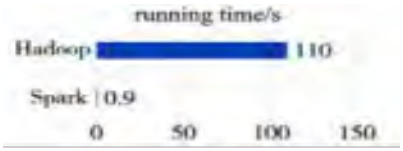

Fig. 1 The comparison diagram of run time

Spark seamlessly integrates components such as spark core, sparks streaming, Spark SQL, Mllib, and Graphx, where spark Core provides memory computing, spark streaming primarily handles real-time applications, spark SQL provides timely queries, MLLIB provides machine learning and GRAPHX provides graph processing, and provides a stack of data solutions.

2.2.2 Spark GraphX.

Graphx is a distributed graph processing system, which uses the data supported by spark directly for distributed computing, and provides and implements a rich graph computing interface, including data graph construction, data graph structure operation, vertex operation, edge operation and other methods, which provide APIs (application Programming Interface, application programming Interface) is also compatible with Pregel and Graphlab interfaces, with powerful graph computing advantages. GRAPHX integrates graph calculation and data calculation into a system, and a combination of the basic operations of data parallel computing and graph calculation provided by the framework can achieve a number of graph calculation algorithms suitable for the current needs. Compared with other distributed graph computing frameworks, it has a flow operation that can complete a whole set of graph calculations very quickly and effectively.

The structure of graph of GraphX can be represented by G (P)={V,E,P}, where V represents the vertex set, E represents the edge set, P represents the data or label collection of vertices and edges, and its basic data structure includes vertices (Vertex), Edges (edge), and Edge triples (Edgetriplet), Many operations are performed on the edge triples. The overall code structure of GRAPHX is shown in Figure 2.

Fig. 2 The overall code structure of GRAPHX

The distributed or parallel processing of the GRAPHX implementation graph is to split the graph into many sub-diagrams by means of point segmentation, and then calculate the sub-diagrams separately, and the calculation can be carried out in stages by iteration.

3. An algorithm of Betweenness Based on Width traversal of Graph on Spark GraphX

The principle of Spark Graphx implemente the betweenness algorithm based on the width traversal graph is described as follows:

1) For each node source s, the width traversal graph starts from s, and the node ID, the source node ID, the access state of the node, the node's precursor node collection, and the number of shortest paths from the node to the source node are recorded. where, there are three access states for nodes, I,u,v,i represents the initial node, U indicates that the node has not been accessed, and V indicates that the

node has been accessed. As shown in Figure 3, the first traversal, select Node A as the source node, calculate the shortest distance between all nodes in A's adjacency table to A, and the number of shortest paths from these nodes to A, on the second traversal, calculate the shortest distance from all nodes of the adjacency table (B,C,E) to (B,C,E), and the number of shortest path to (B,C,E), and so on, until all nodes of the graph are accessed.

2) After the graph is traversed, each node of the graph has information about the above record. Count the number of times each node is used as a precursor node, and add up the number of shortest paths per node to the source node.

3) Use the nodes in the diagram as the source node in turn, and accumulate the number of the node appears in the middle of the shortest path and the number of shortest path of the network, and the $\lambda_k(i,j)$ and $\lambda(i,j)$ of each node K in the formula (1) are obtained, and the betweenness centrality of each node in the graph can be obtained by formula (1).

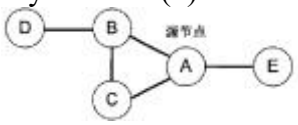

Fig. 3 Example of a network

4. Experiments and Analysis

The algorithm of betweenness centrality based on graph width traversal in this paper is compared with the algorithm of betweenness centrality based on backtracking mechanism put forward by Ulrik brands, and compares the results of key nodes of networks identified by the two algorithms and the algorithm running time. It is chosen to compare the algorithm proposed in this paper with the algorithm of betweenness centrality based on backtracking mechanism proposed by Ulrik brands because both methods are accurate calculations of the formula of the betweenness centrality, and the method proposed by Ulrik brands is the fastest method recognized in the current algorithm for accurately calculating the betweenness centrality.

4.1 Analysis of Experimental Results of Small-scale Data Set.

4.1.1 Datasets.

This article is experimenting on two small networks, the Zachary ' s Karate Club network: The Friendship and social network between 34 members of a university Karate club in the United States in the 70; Football network: 22 members of the football team who competed in the world Championships in Paris in 1998 signed a network of contracts with 35 countries.

4.1.2 Experimental Results and Analysis.

In this paper, the experimental environment for identifying key nodes of small-scale networks is Spark's pseudo-distributed mode, 1G memory and one CPU core. Using the method proposed in this article and the method proposed by Ulrik brands to sort the above two network nodes separately, the result of sorting Top-15 is shown in Table 1~2, and the algorithm execution time is shown in Table 3:

Table 1 Sorting of Zachary network nodes

the method proposed by Ulrik brands	the method proposed in this paper
12	10
10	18
18	12
13	13
14	14
8	8
28	35
5	24
35	5
24	29
29	32
26	25
34	26
22	11
11	34

Table 2 Sorting of Football network nodes

the method proposed by Ulrik brands	the method proposed in this paper
34	1
1	34
33	33
3	3
32	32
2	9
9	2
14	14
28	20
6	6
7	7
20	28
4	24
24	31
31	4

Table 3 Execution time of Two algorithms on Two networks

networks	the method proposed by Ulrik brands（s）	the method proposed in this paper（s）
Zachary network	37	20
Football network	35	17

As can be seen from Table 1~2, the network key nodes identified by the two methods are consistent, and as can be seen from table 3, in the same operating environment, the running time of the algorithm of betweenness centrality based on graph traversal proposed in this paper is half shorter than that of the algorithm of betweenness centrality based on backtracking mechanism presented by Ulrik brands.

4.2. Analysis of the Results of Large-scale Real Social Network.

4.2.1 Datasets.

This article is experimenting on two real-world large-scale networks, Usari97 network: the network represented by the route between the 332 airports, and Email-eu-core network: The e-mail generation networks of large European research institutions.

4.2.2 Experimental Results and Analysis.

In this paper, the experimental environment for calculating betweenness centrality of nodes of large-scale network is the fully distributed mode of Spark, two executors, each executor has 1G memory, 1 CPU core, and the number of partition of per dataset is set to 2.

Table 4 Execution time of Two algorithms on Two networks

networks	the method proposed by Ulrik brands（min）	the method proposed in this paper（min）
USair network	98	53
Email network	255	136

As can be found from Table 4, the running time of the algorithm of betweenness centrality based on graph traversal proposed in this paper is half shorter than that of the algorithm of betweenness centrality based on backtracking meson proposed by Ulrik brands.

5. Summary

In this paper, an algorithm of betweenness centrality based on graph traversal is proposed, which obtains the total number of shortest paths of the network and the number of bars each node appears on the shortest path through the width traversal of the graph, and then accurately calculates the betweenness centrality of the nodes by using the calculation formula of betweenness centrality, and implements the algorithm on the Spark GRAPHX distributed Graph computing platform. Experiments show that the method proposed in this paper identifies the key nodes in line with the method proposed by Ulrik brands, and the running time is shortened by half, and it can be well applied to large-scale networks. This paper provides a reference for the research of betweenness centrality of nodes of network.

References

[1]. Lü LY, Chen D B, Ren X L, Zhang Q M, Zhang Y C, Zhou T 2016 Phys. Rep. 650 (1).

[2]. Lü T Y, PU X F, XIE W Y. Study on Controllability of Complex Network Based on Propagation Immunity [J]. Acta Physica Press2012, (17): 141-149.

[3]. WANG G Z, CAO Y J, BAO Z.A new local evolution model for power network [J] .Acta Physica Sinica. 2009, 58 (6): 3597-3602.

[4]. Bader D A, Kintali S, Madduri K, et al. Approximating Betweenness Centrality[C]// International Conference on Algorithms & Models for the Web-graph. 2007.

[5]. Brandes U. A Fast Algorithm for Betweenness Centrality [J]. The Journal of Mathematical Sociology, 2001, 25.

[6]. Sariyüce AE, Saule E, Kaya K, Çatalyürek ÜV. Shattering and compressing networks for centrality analysis. In: Proc. of the Computer Science.2012.

[7]. Lee M J, Lee J, Park J Y, et al. QUBE: a quick algorithm for updating betweenness centrality.[C]// International Conference on World Wide Web. ACM, 2012.

[8]. Lee M J, Choi S, Chung C W. Efficient algorithms for updating betweenness centrality in fully dynamic graphs [J]. Information Sciences, 2016, 326:278-296.

[9]. Qian Jun, Wang Chaokun, Guo Gaoyang. Community Based Node Betweenness Centrality Updating Algorithms in Dynamic Networks [J]. Journal of Software, 2018.

[10]. Newman M E J. Fast algorithm for detecting community structure in networks [J]. Phys Rev E Stat Nonlin Soft Matter Phys, 2004, 69(6 Pt 2):066133.

[11]. Zhang Guoqiang, Zhang Guoqing. An Algorithm for Internet AS Graph Betweenness Centrality Based on Backtrack [J].Journal of Computer Research and Development, 2006, 43(10):1790-1796.

[12]. Kintali S. Betweenness centrality: Algorithms and lower bounds [J]. Journal of Computing Research Repository, 2008, abs.0809.1906.

[13]. Deng Xiaolong,Li Yuxiao. Efficient node betweenness approximation computation method for large graph in emergency management [J]. System Engineering Theory and Practice, 2015, 35(10):2531-2543.

[14]. Yang Jianxiang, Wang ChaoKun, Wang Meng, et al. Algorithms for Local Betweenness Centrality of Fully Dynamic Multi-Dimensional Networks [J]. Chinese Journal of Computers, 2015(9):1852-1864.

[15]. Bader D A, Madduri K. Parallel Algorithms for Evaluating Centrality Indices in Real-world Networks[C]// International Conference on Parallel Processing. IEEE Computer Society, 2006.

[16]. Yang Cheng. Research and implementation of social network analysis system on MapReduce [D]. Beijing University of Posts and Telecommunications, 2010.

[17]. Freeman L C. Centrality in social networks [J]. Social Networks, 1978, 1.

[18]. Zaharia M, Chowdhury M, Franklin M, et al. Spark: cluster computing with working sets[C]// Usenix Conference on Hot Topics in Cloud Computing. USENIX Association, 2010.

[19]. Gonzalez J E, Xin R S, Dave A, et al. GraphX: Graph Processing in a Distributed Dataflow Framework[C]// Usenix Conference on Operating Systems Design & Implementation. USENIX Association, 2014.G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.