Study of Recognition Algorithm for Core Node in KAD Network Based on BP Model

Li Qiang, Tang Bo, and Yang Jie

Abstract—**In view of the core node recognition in the KAD (Kademlia), a model based on BP is presented to determine whether a node is core node in real time. According to the result of the measurement for KAD, some attribute characteristics of core nodes in the network have been extracted and normalized to obtain an attribute set with higher separable degree. An algorithm in MatLab is designed to train the BP network until the results are limited in a predetermined error range. In addition, the trained BP model is adopted to determine the tested nodes, and the results of the experiment show that the method can judge degrees of importance of nodes in real time, and the recognition accuracy rate is up to about 70%.**

Index Terms—**Back-prorogation (BP), KAD network, core node, recognition.**

I. Introduction

Sorting of network nodes is a basic and important issue in a complex network and is extensively applied to data mining, network analysis, network prediction, network security and control and other purposes. As intensified researches into complex network indicate, a great number of real networks are neither regular nor completely random. Therefore, productive evaluation and measurement of the importance of network nodes is not only a primary issue in network data mining, but also a research focus on complex network, social network, Internet search and system science [1]. KAD is a new-generation DHT network [2] derived from 10 years of development of the P2P technology. Its advantages such as decentralization, mensurability, easy expansion and high fault tolerance rapidly popularize it in file sharing, instant communication, distributed storage, cloud storage and other areas. This design without central server makes different nodes relatively equal, where they can be both service provider and service requester. As the KAD network develops, it is used by millions of concurrent users, while the massive number of users makes it very difficult to control and supervise the network. Numerous network measurement experiments indicate that the loads on the nodes in the DHT network are substantially imbalanced [3]-[5]. The nodes in the KAD network that provide more routing services or downloading services for other nodes are known as core (important or key) nodes. If this imbalance can be used to discover and recognize the nodes that play a core role in the network, this will be a great contribution to the supervision on the KAD network.

Manuscript received January 12, 2016; revised April 1, 2016.

Li Qiang and Tang Bo are the Institute of Communication and Signal, CREEC, Chengdu 610031, China.

Yang Jie is with the Institute of Science and Technology, CREEC, Chengdu 610031, China.

II. Related Works

Link-based node sorting is a core task in network analysis, where the nodes are sorted based on the importance of the link between two nodes shown in a graph. For now, the domestic and foreign researchers use four typical methods for node sorting [6]. First method: evaluate the importance of a node using the network-only static parameters such as node centrality, concentration centrality, sub-graph centrality and network traffic centrality. The traditional static social analysis method has a significant disadvantage that it ignores the important factors that cause network status change [7], which results in information deficiency. For example, this method ignores the connections between different individuals, the mutual effects between related individuals and time-related change trend [8]. Second method: use the graph partitioning method, i.e. identify the division points in the graph and treat them as core nodes, for these nodes can divide the graph into several sub-graphs [9]. Third method: use the node sorting concept in search engine to discover and sort the core nodes. This method is based on the random walk model [10]. Fourth method: use an algorithm based on statistical method or data mining, e.g. the classical models including frequent item set [11], [12], hidden space model [13] and two-step clustering model. For the data mining algorithm, the most typical method is the frequent item set. However, all these methods have a significant disadvantage that node identification is hysteretic, i.e. it costs considerable time/space complexity to determine the importance of a node, and, put another way, an unknown node cannot be identified at real time. Thus the usefulness of these methods is rather limited in the application scenarios that strictly require real-time results.

III. Code Node Evaluation Algorithm

In the KAD network, a core node is one that plays an important role in the basic services of the network. Such nodes indeed provide more access services and routing services of other nodes. Their definitions are given below [14]:

Definition 1 (routing hotspot): in a zone *Zx*, each node has n resource sets. Assume the probability of the node a being accessed by other nodes is *Pa* and *a* is accessed Ta times by other nodes in the unit time *T*, then $Ta = Pa \bullet n$. If the given threshold value of access count is *t* and $Ta > t$, *a* is called query hotspot node (routing hotspot).

Definition 2 (keyword hotspot): in a zone *Zx*, assume the index of the nodes in that zone is *Ia* and the access count of each node is *Ca*. If the given threshold value of the access count of a keyword is *c* and *Ca* > *c*, *Ia* is called keyword hotspot. If there are totally n indexes, the access count of the node is the sum of the access counts of all the indexes:

$$Ta = \sum_{a=1}^{n} (Ia \cdot Ca) \quad (1)$$

Definition 3 (core node): in a zone *Zx*, assume the node *A* is a routing hotspot or a keyword node. If, in the given time *T*, the access count *R* of node *A* is greater than the given threshold value c, node *A* is a core node.

Neural network is a cross-disciplinary field which simulates the human brain thinking pattern to reveal the rules hidden in mass data. It has been extensively applied in various fields, with many great achievements produced. We use the BP network to learn information from the currently available sample nodes, with the expectation of discovering valuable rules, to identify unknown nodes in a more real-time and more accurate manner. Fig. 1 presents the entire process of using BP neural network for training and testing. The key steps are sample characteristics extraction/processing and network training.

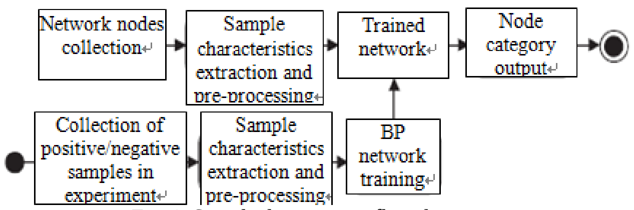
Network nodes collection
Sample characteristics extraction and pre-processing
Trained network
Node category output
Collection of positive/negative samples in experiment
Sample characteristics extraction and pre-processing
BP network training

Fig. 1. Sample data training flow diagram.

In Fig. 1, characteristics extraction is to select a group of most ideal subsets from one characteristic set to fulfill the training objective. To make the processing easier, you need to normalize the characteristic data in advance to make them vary in the range of [0, 1]. Next, we generalize and improve the characteristic attributes available in KAD.

1) *R'c*: effectiveness rate of returned node

Upon each query in KAD, if the node is not the target node, the proximate nodes in a routing tables will be returned to the requester. According to the KAD routing algorithm, when the neighbour node receives the request, it first selects a specified number of nodes from the corresponding K-bucket. If the number of the nodes in that bucket is less than the specified number, the proximate K-bucket will have to provide nodes too. Hence, as long as the number of nodes in the routing table is not less than the specified number, the specified number of nodes will be returned. If every neighbour node is online, the number of routing nodes provided in the next hop will always be the same every time. Therefore, quantity computation does not produce ideal separability. We use the effectiveness rate of returned node instead. The calculation formula is shown below:

$$R'c = Ln / Bn \quad (2)$$

where: *Bn* represents the total number of routing nodes returned by the node *I* in the next hop and *Ln* the number of the currently online nodes. To obtain the value of *Ln*, execute a Ping to determine the number of remaining online nodes after the neighbour node has returned the information. In the information returned from the neighbour node, this value represents the proportion of the valid nodes in the total number of returned nodes.

2) *S'c*: average xor distance

In KAD, the count of hops, as a measurement criterion, does not accurately indicate the distance between two nodes, so we use the average xor distance instead. This value represents the extent to which the returned result approaches the target node after a query on the node and indicates the role the node plays during the query. If the node I returns totally n routing nodes, the average distance from node I can be calculated by the following formula:

$$S'c = \sum_{i=0}^{n} \frac{Xor(r_i, d)}{Xor(s, d)} / n \quad (3)$$

where: *ri* represents the returned node from *I*, and s and d respectively represent the request node and target node. It is easy to infer from this formula that the greater the returned distance, the closer the node is to the target node after the query on the node 1.

3) *T't*: time for processing the result of query

The greater this value, the higher the processing performance of the queried node. Usually, the time interval between the moment of sending the message and the moment of receiving the message is deemed as the round trip time. To make the processing easier, set a fixed TTL cardinal number as the reference object. A value less than this value is deemed approximately equal to this value. We set TTL to be 2,000 ms here. This value is to be determined from the following formula. Where: Mt is the maximum value among the return time values. A bigger fixed value may also be set:

$$T't = T_a / M_t \quad (4)$$

4) *R'i*: node return-query quantity ratio

This value represents the proportion of the number of the files returned from the target node in the total number of returned results. It can be calculated by the following formula:

$$R'i = Ri / Rc \quad (5)$$

where: *Rc* represents the total number of files returned from all nodes in the query. However, this value is meaningful only if a node returns results. It is impossible to guarantee that every node tested will return stored results, thus it is necessary to process the *R'i* that has been determined from the routing via multiple nodes, to determine the *R'i* value of each node. The essential concept is that the nodes are arranged based on their *S'c* values. *S'c* represents the extent to which the recommended routing node approaches the target node. The greater the distance from the node recommended by node I, the closer the request node is to the target node and the more the *R'i* values allocated to node I. This value is calculated by the following formula:

$$R'i = \begin{cases} Ri / Rc,\ I \text{ is the target node} \\ S'c \times Ri / Rc, I \text{ is the routing node} \end{cases} \tag{6}$$

5) *F'd*: node return-query quality ratio

This value represents the extent to which query result file name matches the query word. It is calculated by the following formula:

$$F'd = \sum_{i=0}^{n} Fi / c \tag{7}$$

Fi represents the extent to which the returned file matches the query keyword; *n* represents the number of all files returned from the node; c represents the sum of the file indexes returned from different nodes. The greater this value, the better the returned value in the query matches the requirement of the keyword.

This value is also meaningful for only a node that returns results, therefore, similar to the processing of the preceding characteristic value, this value is to be converted in the following formula:

$$F'd = \begin{cases} \sum_{i=0}^{j} Fi / c,\ I \text{ is the target node} \\ S'c \times \sum_{i=0}^{j} Fi / c, I \text{ is the routing node} \end{cases} \tag{8}$$

The aforementioned process generates a normalized characteristic set <*R'c*, *S'c*, *N'm*, *R'i*, *F'd*>. Every value in this set varies in the range of [0, 1].

For the purpose of training, you need to collect both positive and negative samples for learning and testing. The tool used to collect training samples is the customized software KFetch [l4] which obtains snapshoot of the network topology. We use the concentration centrality parameter in the social network and the random walk algorithm for node evaluation and select the core nodes and non-core nodes identified by both algorithms. The test commenced at 20:00 on February 4, 2012; sampling lasted for 40 min. The existing core nodes are used as the positive samples, where 60% of these nodes are used as training samples and the other nodes are used as testing samples. Thus the training sample set and testing sample set are obtained.

IV. Experiment Process and Result Analysis

For different purposes, you can select a proper subset out from the aforementioned characteristic sets and use it for the training. The KAD message types indicate there are four types of basic RPC (Remote Process Command). Queries include node query and keyword query. The two types of messages are used in different scenarios, thus their characteristic value subsets are also different. Map these two types of message commands to the aforementioned characteristic sets and select the suitable characteristic attributes to establish the corresponding characteristic subset.

Mark the characteristic set related to Find_Node as Cn and select its corresponding characteristics <*R'c*, *S'c*, *N'm*>. In a query aimed to a node, there are only two possible returned results: getting the target or not getting the target. Therefore, it is relatively easier to process the results. Mark them as *C'n* =<*R'c*, *S'c*, *N'm*>. Mark the characteristic set related to Find_Value as Vn and its corresponding characteristics include <*R'c*, *S'c*, *N'm*, *R'i*, *F'd*>. This characteristic set contains the query result index and index result quality, which allows us to more comprehensively evaluate the results of query. Mark them as *Vn* =<*R'c*, *S'c*, *N'm*, *R'i*, *F'd*>.

Use Find_Node to collect the characteristic attributes of the node. Divide the zone into parts and select one random node ID from each part as the target ID to be queried. We do so to more accurately determine the distances from a node to other positions. Select one neighbour node for the ID of each of the test servers, to generate the client ID. Before commencing the test, make sure there are sufficient nodes and nodes have been added to your routing table. In this experiment, the parameter is set to be 8. First, collect the attributes via the selected core nodes. Proceed with the test for 25 min. Obtain the characteristic values of the samples used for all the nodes to be trained and tested.

Group the normalized sample data as learning samples and training samples, in the format of *L* = *T* =<*R'c*, *S'c*, *N'm*, *E*>. *E* is the expected value of the node, i.e. the probability of the node being a core node. The expected output result is a probability value, thus, during sorting of importance degrees, the most important node cannot be treated as equal to the generally important nodes, for this will degrade the accuracy of the result. Hence, the core nodes obtained from the aforementioned algorithms need to be converted accordingly. Specifically, pick out the core nodes identified by both algorithms and sort them as per their importance degrees. The probability value range for the specified result is [0.6, 1]. In other words, the core nodes are divided into five equally sized zones and the nodes in each zone correspond to a given expected probability value. The negative samples are also to be processed this way. Send these groups of training samples to the network for learning. In this experiment, the learning error is set to be 0.001. The number of neurons in the hidden layer is 1.4 times the number in the input layer, i.e. it is set to be 5. The transfer function for the input layer is set to be an S type tangent function tansig and the transfer function for the output layer is set to be an S type logarithmic function logsin. The negative-gradient weight correction method is adopted here. In order to make coding easier, both learning process and testing process in this experiment are accomplished with the help of the neuron component in the MatLab[15] tool kit. The list of the processes is given below. The input data are given in the form of vectors whose values correspond to the variable P. The expected output value is also assigned in the form of a vector to the variable T.

1) net_ 1 =newf((minmax ([P]), [5, 1], {'tansignpurelin'}, 'traingdm');
2) inputWeights=net_1 .IW {1, 1};
3) inputbias=net_1 .b{1};
4) layerWeights=net_1.LW{21};
5) layerbias=net_1.b{2};

```
6)  net1 .trainParam.show=1 000;
7)  net1 .trainParam.lr=0.05;
8)  net1 .trainParam.mc=0.009;
9)  net_1.trainParam.epochs=5 000;
10) net_1.trainParam.goal=1E-3;
11) [net_1tr]=train(net_1, [P], [T]);
12) sim(net_1, [P]);
```

Among the codes listed above, minmax(P) represents the value range that uses the maximum value and minimum value of the input data as the training data; net_1 is the output network generated after the training.

Through this multistep network training, we learn the sample data fitting that meets the error requirement. The experiment results are shown in Fig. 2.

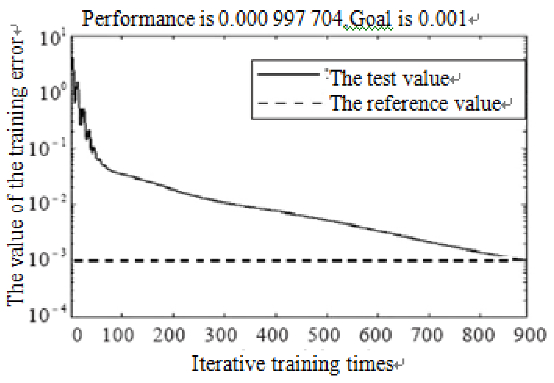

Fig. 2. Diagram of Find_Node characteristics learning fitting effect.

According to Fig. 2, if learning has been done up to 889 times, the error is limited within the specified range. By comparing the output probability of the trained network to the expected value, if the error is less than 0.2, the experiment is deemed successful. Then the recognition rate of positive samples is 78.3% and the recognition rate of negative samples is 67.1%. We test the trained network model in the actual KAD, where we use the KFetch tool to identify the node relations and then extract the attribute sets of the online nodes and incorporate them into the model to identify the nodes. The experiment proceeds for 30 min every time and it is repeated 5 times. Then collect and analyze the statistics on the experiment results. Meanwhile, compare the recognition results to the core nodes identified by the concentration centrality method and random walk algorithm. The comparison indicates that the average recognition rate of core nodes achieved by the BP model is 63.8%. The recognition rate in the actual network may be lower than the value from the training. However, this method is still proven to be feasible and practical.

V. Conclusion

In this paper, we use measurement experiment to collect the attribute characteristics of KAD nodes and use certain query commands to establish the characteristic subsets used in the experiment. According to the experiment result, recognition rate still needs to be enhanced to acclimate to the dynamic nature of the network nodes. However, this real-time identification method does solve the hysteresis of node identification, thus it has the potential for further application. To enhance accuracy of recognition, we will undertake more intensive researches into node attributes in the future. Conclusively, the performance of the BP network depends on two factors: the quantity of the attribute characteristics of the selected node and the quality of the selected attribute characteristics. To seek useful attribute characteristics of a node, we need more intensive analysis on the nodes in the KAD network. Then we can extract the useful attribute characteristics of the nodes to amplify the attribute sets. Besides, we can try other recognition modes such as simulated annealing algorithm, Tabu search algorithm and genetic algorithm [16] to transform, calculate and extract the available attribute characteristics to generate better separable attribute characteristic values, so as to substantially enhance recognition accuracy of core nodes.

References

[1] J.-J. He and R.-F. Li, "Node sorting based on modified random walk model," *Computer Engineering and Applications*, 2011, vol. 47, no. 12, pp. 87-89.

[2] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer informatics system based on the XOR metric," in *Proc. the 1th International Workshop on P2P Systems*, 2002, pp. 53-65.

[3] J. Jiang and Q.-N. Deng, "Nonuniform distribution in eMule system," *Microelectronics and Computer*, 2007, vol. 24, no. 10, pp. 153-156.

[4] W. Xiong, D.-Q. Xie *et al.*, "Self-adaption load balancing of a structured P2P," *Journal Software*, 2009, vol. 20, no. 3, pp. 661-663.

[5] Z.-Y. Li and G.-G. Xie, "P2P load balancing algorithm based on DHT," *Computer Research and Development*, 2006, vol. 43, no. 9, pp. 1579-1580.

[6] C.-G. Zhou, P.-C. Qu *et al.*, "DSNE: A new dynamic social network analysis algorithm," *Journal of Jilin University: Engineering Edition*, 2008, vol. 38, no. 2, pp. 408-411.

[7] H. Cai *et al.*, "Algorithm research on community mining from dynamic social network," *Journal of Jinlin University*, 2008, vol. 26, no. 4, pp. 380-382.

[8] T. Y. Berger-Wolf and J. Saia, "A framework for analysis of dynamic social networks," in *Proc. the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006, vol. 12, pp. 523-528.

[9] N. He *et al.*, "Overview on exploring of important nodes in complex network," *Computer Science*, 2007, vol. 34, no. 12, pp. 1-4.

[10] J. Guo *et al.*, "Analysis and application of random walk circuit and parallelization improvement," *Computer Engineering and Applications*, 2010, vol. 46, no. 18, pp. 199-201.

[11] Z.-J. Deng *et al.*, "Study on the algorithm for exploring the set of most frequent items in P2P network," *Computer Application Research*, 2010, vol. 27, no. 9, pp. 3490-3492.

[12] W.-J. Song *et al.*, "Core node identification based on graph based frequent item set," *Computer Science and Exploration*, 2010, vol. 4, no. 1, pp. 82-85.

[13] P. Sarkar, "Dynamic social network analysis using latent space models," in *Proc. the ACM SIGKDD Explorations Newsletter*, 2005, pp. 31-35.

[14] J. Wang, *Key Technology Research and Realization Based on KAD Network Supervision*, Cheng Du: Sichuan University, 2012.

[15] FECIT Technological Product Research Center: Neural Network Theory and MATLAB7 Realization, MATLAB Applied Technology, Beijing: Electronic Industry Press, 2005, pp. 4-90.

[16] Z.-Q. Bian and X.-G. Zhang, *Pattern Recognition*, Beijing: Tsinghua University Press, 2006, pp. 176-208.