ICIC Express Letters
Part B: Applications
Volume **7**, Number **11**, November **2016**
 ISSN 2185-2766
pp. **2381–2387**

CRITICAL NODE MEASURING METHOD IMPROVED BASED ON PAGERANK AND BETWEENNESS

Lei Wang[1,2,3,*], Jun Dong[1,3] and Jiadong Ren[1,3]

[1]College of Information Science and Engineering
Yanshan University
[3]The Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province
No. 438, Hebei Ave., Qinhuangdao 066004, P. R. China
[*]Corresponding author: wangl216@163.com; donycosmos@163.com; jdren@ysu.edu.cn

[2]E&A College
Hebei Normal University of Science and Technology
No. 360, Hebei Ave., Qinhuangdao 066004, P. R. China

Received April 2016; accepted July 2016

Abstract. *In software execution network, PageRank and betweenness methods are used to determine the importance of nodes. The experiment results show that the differences between nodes are not strong and cannot reflect the software nodes' importance degree. In order to solve this problem, this paper adds the entropy weight method to the calculation of node weights, and puts the weights of nodes into the computing of two measuring methods above. Finally, we get the ranking result of nodes' importance which means we can find out the critical nodes from top ones. Experimental results indicate that the method is efficient for critical node measuring.*
Keywords: Software execution network, Critical node measuring, PageRank, Betweenness

1. **Introduction.** How to use quantitative analysis to evaluate the nodes' importance in large-scale software network, and find out the critical nodes in the network is a fundamental problem for both software quality and software security research recently. There are many methods on the study of critical node measuring. The evaluating metrics can be integration of influence, position, and other factors. A lot of metrics and methods of critical node measuring are submitted.

Betweenness [1] was utilized to measure the importance of nodes, and the nodes with larger betweenness are more important in the network. Degree metric [2] was proposed to measure the importance of nodes in network, namely, if a node has larger in-degree, it will be suggested to undertake important task in network. Study of Xiao and Xiao [3] showed that incomplete global information has different impacts on an intentional attack in different circumstances, while local information-based attacks can be actually highly efficient. Gao and Li [4] used complexity to compare software network and corresponding random networks, and found the important nodes in network with PageRank algorithm. Masuda and Kori [5] extended Laplacian-based centrality and adopted the idea of PageRank to introduce global connectivity between all pairs of nodes with certain strength. A new page scoring algorithm [6] based on centrality measures and PageRank algorithm is proposed to score Web pages in an effective manner. Ugander et al. [7] found the number of connected subgraphs among neighbor nodes decides the importance of nodes. Bhattacharya et al. [8] defined a measure called NodeRank that assigns a numerical weight to each node in a graph, to measure relative importance of that node in software. TOPSIS [9] was utilized to aggregate the multi-attribute to obtain the evaluation of node importance of each node. Via assigning degree-dependent weights onto links associated with the ground nodes, Li et al. [10] proposed weighted LeaderRank to identify influential

spreaders on social networks. Wei et al. [11] used an edge weighting method by adding the degree of its two end nodes to construct weighted networks, and proposed a weighted k-shell decomposition method to identify the node importance in complex networks.

There are two basic methods to measure critical nodes from the researches mentioned above: PageRank method and betweenness method. The two methods measure the importance of nodes from different angles. However, the experiment results show that the nodes with similar structure cannot be distinguished by PageRank and betweenness methods. The researches also show that nodes with greater influence may normally crash the software system. So it is important to measure the importance of nodes, and ensure the measuring results are not the same. This paper proposes critical node parameters index (*CN_Value*) to measure the importance of nodes based on PageRank method and betweenness method. In order to make the measure values different from each other, we put entropy value into the calculation of node weights.

The rest of the paper is organized as follows: Section 2 gives some definitions; critical nodes mining algorithm and node weights computing method are discussed in Section 3; experimental results and analysis are presented in Section 4; Section 5 concludes our study.

2. **Definitions.** We can form a software execution graph model by taking the elements which maybe statement, predicate, basic block, function and so forth as nodes, the execution path sequence as edges. Graph as an expression of dependency can reduce duplicated expressions among elements and demonstrate the strength of element dependence intuitively. In this paper, we use the basic block as the unit of a software dynamic execution graph. The basic definitions are as follows.

Definition 2.1. *Software execution graph. The term* $G = \langle V, E \rangle$ *is used to denote a graph* G. $V = v_1, v_2, \ldots, v_n$ *is the set of vertices, where* v_i $(1 \leq i \leq n)$ *is a basic block node. And* $E = e_1, e_2, \ldots, e_{n*n}$ *is the set of directed edges where* $e_k =< v_i, v_j >$ $(1 \leq i, j \leq n)$ *is a directed edge between two nodes.*

Definition 2.2. *PageRank model in software execution graph.*

$$NR(A) = \left(\sum_{i=1}^{m} \frac{NR(T_i)}{C(T_i)} \right) \times w(A). \tag{1}$$

$\sum_{i=1}^{m} \frac{NR(T_i)}{C(T_i)}$ *is the sum of components that nodes which point to node* A *account for its indegree.* $w(A)$ *is the weight of node* A.

In software execution process, the node with larger outdegree has greater influence, namely, its PageRank value is greater.

Definition 2.3. *Betweenness model in software execution graph.*

$$NT_i = \left(\frac{2 \sum\limits_{j<k} g_{ik}(i)/g_{ik}}{n(n-1)} \right) \times w_i. \tag{2}$$

$\sum_{j<k} g_{ik}(i)/g_{ik}$ *is the betweenness value of node* i, *and* g_{jk} *is the number of the shortest paths from node* j *to node* k, *while* $g_{jk}(i)$ *is the number of the shortest paths from node* j *to node* k *which passes node* i. w_i *is the weight of node* i.

That the node i *has greater* NT_i *value means it plays a more important role in software execution network.*

Definition 2.4. *CN_Value parameters index. It is used to measure the node importance.*

$$CN\_Value[N] = \alpha \times NR[N] + \beta \times NT[N]. \tag{3}$$

NR is PageRank algorithm model which is used for sorting the node in the software execution network, and it is mentioned in Formula (1). NT is betweenness algorithm model which is mentioned in Formula (2). α and β represent the weights of NR and NT respectively, and $\alpha + \beta = 1$. Therefore, CN_Value is proportional to NR and NT values of nodes. In order to embody fairness, this paper uses the entropy weight method strategy to determine the weight relations of nodes automatically.

3. **Critical Node Measurement Based on Entropy Weight Method.** In this paper, we build up a software execution model firstly, then select the correct and error behavior graphs, and analyze the differences between the two graph models to acquire critical node set. Due to the differences of nodes dependence, execution sequence and importance degree, each node in the node set has different critical levels. We acquire a new measure index *CN_Value* by combining entropy, PageRank and betweenness methods. The index is used to compute the critical levels of nodes, namely, we sort the nodes by *CN_Value* and find out the critical node which is the most influential in software execution.

3.1. **Establishment of software execution graph set.** In software testing process, each test case has different execution sequences which correspond to a graph. The process of building the basic graph set has been described as follows.

(1) Node set V is built by scanning basic block which is covered by path.

(2) Edge set E is constructed by means of adding the child nodes for each node, and it expresses the connection between nodes.

(3) Graph model G is created by using node set V and edge set E.

(4) We put a large number of graph sets of test cases together to form the graph data set $Gset$. $Gset$=$\{G_1, G_2, \ldots, G_n\}$.

(5) In order to establish critical node set, $Gset$ is divided into two parts: $CorrectGset$ and $ErrorGset$. There are a lot of error graphs in the $ErrorGset$, and correspondingly, the correct graphs are included in $CorrectGset$.

3.2. **Building of the node set.** In order to form the critical node set, the $Cgraph$ and the $Egraph$ models are built by using partition, statistics and screening methods, and then the comparative analysis between two models is used to find the differences of two graph models. The critical node set is formed by the nodes which can cause the differences.

There is a principle to select the error graph model to do comparative analysis: when the test result failed, it must contain the critical code block that causes error in the execution path.

$$Egraph = MIN(Gcov), \quad (G_i \in ErrorGset). \tag{4}$$

Gcov records the cover node numbers of all the error graphs in $ErrorGset$. $Egraph$ model is built by mining the error graph with minimum coverage in $ErrorGset$.

$Cgraph$ model is built via traversing the $CorrectGset$ to find the similar graph with $Egraph$ model. We calculate the similar graphs of two graphs firstly, and then sort the calculation result descending by the size of similarity.

$$Cgraph = CorrectGset(Gsim(i).sim < k), \quad (1 < i < n). \tag{5}$$

Gsim(i).sim records index number and the corresponding similarity. Comparing similarity with the threshold k, $Cgraph$ is built through $CorrectGset$.

$$Cmis = F(Egraph, Cgraph). \tag{6}$$

$Cmis$ is the maximum common subgraph of two graph models, namely the similar parts. F is a function for gaining the maximum common subgraph between $Egraph$ and $Cgraph$.

The critical node set includes two parts: the complementary set of $Cmis$ and some nodes tend to decide whether its child nodes can be implemented. The mining process has been described in Algorithm 1.

Algorithm 1 Mining the critical node set

Input: $Egraph$, $Cgraph$

Output: $Nset$

(1) $Cmis = \phi$, $Nrest = \phi$, $Niso = \phi$, $Nset = \phi$ //$Niso$ is the node set in $Cmis$
(2) $Cmis = F(Egraph, Cgraph)$
(3) $Nrest = U(Egraph, Cmis)$ //$Nrest$ is the complementary set of $Cmis$
// U is a function for node complement set of G_2 in G_1
(4) for i from 1 to n
(5) if $Nrest(i)$.parent $\in Cmis$ // .parent expresses the parent node of a node
(6) $Niso = Niso \cup Nrest(i)$.parent
(7) end if
(8) end for
(9) $Nset = Nrest \cup Niso$ //$Nset$ is the set of total nodes
(10) Weight($Nset$)
//Weight() is a function for sorting the final set of nodes by node weight
(11) output($Nset$)

Line 1 of the algorithm initializes the $Cmis$, $Nrest$, $Niso$ and $Nset$ as empty sets. In line 2, the maximum common subgraph node set $Cmis$ of $Egraph$ and $Cgraph$ models is recorded. We use function U to mine the node set $Nrest$ which is the Complementary set of $Cmis$ in $Egraph$. The node set in $Niso$ is given in line 4 to line 8. When the parent node of a node belonging to $Nrest$ set is contained in $Niso$ set, the parent node is combined with $Niso$ set. In line 9, the set of total nodes, $Nset$, is got through merging both $Nrest$ and $Niso$ sets. In line 10, the nodes of $Nset$ are sorted by the node weight which is measured by CN_Value, and the set $Nset$ is returned in line 11 finally.

3.3. **The node weight calculating strategy.** In Algorithm 1 line 10, we need to sort the nodes by node weight. PageRank method and betweenness method are used to comprehensive judgment of critical degree of the nodes in software execution graph. In order to make the results more objective, we lead entropy weight method into the measure calculation. The model establishment is shown as follows.

1) Using entropy method strategy to determine the index weights.

Suppose there is m index which is applied to evaluating n objects, the process of determining the weight vector which is using entropy method is described as follows.

a. Determine the entropy of j index.

$$S_j = -K \sum_{i=1}^{n} P_{ij} \ln P_{ij}, \quad (j = 1, 2, 3, \ldots, m). \tag{7}$$

$K = \frac{1}{\ln n}$, $P_{ij} = \frac{1+b_{ij}}{\sum_{k=1}^{m}(1+b_{ik})}$.

b. Entropy vector W is

$$W = (w_j)_{1\times m} = \left(\frac{1 - S_j}{m - \sum_{j=1}^{m} S_j} \right)_{1\times m}. \tag{8}$$

2) According to Formula (3), CN_Value is calculated to weight the nodes in $Nset$.

3) We sort the nodes descending by CN_Value, and take top k nodes as critical nodes.

4. Experimental Results and Analysis.

4.1. **Datasets.** To check the practical feasibility of our method, we use the famous Siemens benchmark test suite in the field of software testing. Siemens contains seven kinds of programs, including print tokens, print tokens2, replace, schedule, schedule2, tcas and tot_info. Each program has been implanted with error. In this paper, we take print tokens as an example to describe the critical node measurement.

4.2. **Results and analysis.** Figure 1 shows the analysis of node importance with weighted betweenness, weighted PageRank and entropy weight methods. From the figure, the results show that the division of the critical nodes and non-critical nodes are basically the same, but there are a few differences. The results between entropy weight method and PageRank method are similar except the sequence of nodes 0 and 60, 59 and 82. This is because the proportion of node weights which is assigned to PageRank method is greater when we calculate the node weights with entropy weight method. So the sorting results between entropy weight method and weighted PageRank method are more consistent.

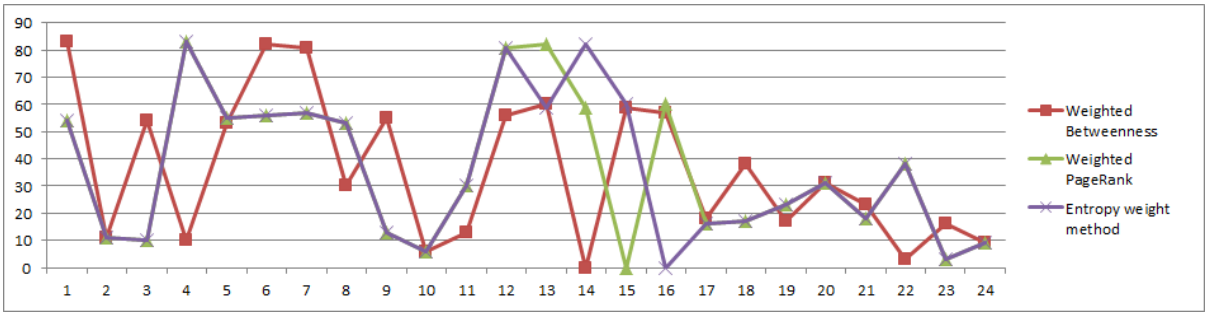

FIGURE 1. Infection percentage

Table 1 is the node importance measurement values which are gained by using weighted betweenness, weighted PageRank and entropy weight methods respectively. As seen from the table, the results of weighted betweenness method have small differences from each other, and it is not easy to compare the differences within five valid value expressive ranges. Instead, the results of weighted PageRank method are large and easy to compare the differences. The disadvantage of both is some nodes have the same measurement values, such as node no. 6 and 55, 56 and 60 in the second column and node no. 16 and 17, 55 and 56 in the third column. In Table 1, although the sorting results obtained from three methods are basically the same, the values with entropy weight method are not only easy on comparison, but also have different values from each other. This is because the proportion of node weights is different between PageRank method and betweenness method according to Formula (3). In summary, the measurement of entropy weight method has stronger distinction between nodes, so it is a more effective node importance measuring method.

5. Conclusions.

We propose a critical node measuring method based on PageRank method and betweenness method. In order to ensure the objectivity, entropy weight method is used to integrate the values obtained from two methods above, and then sort the nodes descending by the important values. The experiment results show that this method makes the measurement value of each node different, and determines the critical nodes effectively. we will make further efforts to improve and optimize this method, and apply it in the software fault location process.

TABLE 1. Evaluation of node important measurement using weighted betweenness, PageRank and entropy weight methods

Nodes NO.	Weighted Betweenness	Weighted PageRank	Entropy Weight Method
0	0.00955	2.42858	1.09341
1	0.00475	0.85	0.38435
3	0.0048	0.99662	0.4502
6	0.00975	2.43791	1.1022
9	0.00477	0.9775	0.44174
10	0.01869	6.06035	2.72441
11	0.0198	7.03621	3.16256
13	0.0097	2.45971	1.10739
16	0.00477	1.0581	0.48056
17	0.00497	1.0581	0.47794
18	0.00508	1.00131	0.45244
23	0.00482	1.00871	0.46391
30	0.0098	2.43137	1.0976
31	0.00495	1.00871	0.45576
38	0.005	1.0002	0.45194
53	0.01005	2.53238	1.14262
54	0.0197	7.16412	3.22011
55	0.00975	2.77462	1.24882
56	0.0096	2.77462	1.24869
57	0.0093	2.75543	1.24018
59	0.0095	2.42859	1.09379
60	0.0096	2.42857	1.09346
81	0.00985	2.42941	1.09592
82	0.0099	2.42882	1.09352
83	0.02111	4.85729	2.18403

Acknowledgment. This project is supported by the Natural Science Foundation of Hebei Province, P. R. China, under Grant (No. F2014203152) and (No. F2015203326). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] L. C. Freeman, Centrality in social network conceptual clarification, *Social Network*, vol.1, no.3, pp.225-239, 1979.

[2] X. F. Wang, Complex network: Topology, dynamics and synchronization, *International Journal of Bifurcation and Chaos*, vol.12, no.5, pp.885-916, 2002.

[3] S. Xiao and G. Xiao, On intentional attacks and protections in complex communication networks, *Global Telecommunications Conference*, pp.1-5, 2006.

[4] S. Gao and C. Li, Complex network model for software system and complexity measurement, *2009 World Congress on Computer Science and Information Engineering*, pp.624-628, 2009.

[5] N. Masuda and H. Kori, Dynamics-based centrality for directed networks, *Physical Review E*, vol.82, no.5, pp.514-539, 2010.

[6] S. Qiao, J. Peng, T. Li, H. Li, T. Li and C. Wang, Hybrid page scoring algorithm based on centrality and PageRank, *Journal of Southwest Jiaotong University*, vol.45, no.3, pp.456-460, 2011.

[7] J. Ugander, L. Backstrom, C. Marlow and J. Kleinberg, Structural diversity in social contagion, *Proc. of the National Academy of Sciences of the United States of America*, vol.109, no.16, pp.5962-5966, 2012.

[8] P. Bhattacharya, M. Iliofotou, I. Neamtiu and M. Faloutsos, Graph-based analysis and prediction for software evolution, *Proc. of International Conference on Software Engineering*, pp.419-429, 2012.

[9] Y. Du, C. Gao, Y. Hu, S. Mahadevan, Y. Deng, L. Lü and T. Zhou, A new method of identifying influential nodes in complex networks based on TOPSIS, *Physica A: Statistical Mechanics and Its Applications*, vol.399, pp.57-69, 2014.

[10] Q. Li, T. Zhou, L. Lü and D. Chen, Identifying influential spreaders by weighted LeaderRank, *Physica A: Statistical Mechanics and Its Applications*, vol.404, no.24, pp.47-55, 2014.

[11] B. Wei, J. Liu, D. Wei, C. Gao and Y. Deng, Weighted k-shell decomposition for complex networks based on potential edge weights, *Physica A: Statistical Mechanics and Its Applications*, vol.420, pp.277-283, 2015.