ICIC Express Letters
Part B: Applications
Volume **7**, Number **9**, September **2016**

ICIC International ©2016 ISSN 2185-2766
pp. **1879–1884**

MINING CRITICAL NODES IN SOFTWARE EXECUTION NETWORK BASED ON COMPLEX NETWORK

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 February 2016; accepted May 2016

Abstract. *It is significant for measuring the importance of nodes accurately to improve software stability and robustness in software network. A software execution directed network takes function as a node and relationship of function as an edge in this paper. To find the critical nodes in the network, a novel method is proposed to measure the nodes' importance by means of depth search mining in software execution. According to the principle of cascading failure, a novel critical nodes metrics FID is defined for nodes measuring and sorting. Critical nodes mining (CNM) algorithm is put forward for calculating FID value of each node and sorting. We choose top-ranking nodes as critical nodes which play an important role in software execution process. Experimental results show that CNM algorithm can measure the critical nodes accurately in software network.*
Keywords: Software execution network, Complex network, Critical nodes

1. **Introduction.** In complex networks, cascading failures usually occur when one part of the system fails. It will lead to the nearby nodes to take up the slack for the failed component. This may cause the other nodes failing. Cascading failures also occur in software execution network. The node which can cause more nearby nodes failing plays an important role in software network, and it is considered as a critical node. Therefore, it is significant for measuring the importance of nodes accurately.

Betweenness [1] was utilized to measure the importance of nodes, and the larger betweenness it has, the more important status it has in network. In order to predict the behavior of percolation on random graphs which was under general types of breakdown or interference, node degree [2] was used to measure the importance of nodes in network, and the nodes with larger degree are considered as critical ones. Degree metric [3] 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. However, the results obtained by these methods mentioned above are not very accurate. Study of Xiao and Xiao [4] showed that protect a small amount of critical nodes can improve the robustness significantly, and proved that some critical nodes missing would reduce the efficiency of purposeful attack. Nodes reachability metric [5] was proposed to measure the importance of nodes in network. It is more accurate than the degree and betweenness measures, but the time complexity is not reduced. Kitsak et al. [6] believed that nodes importance was related to its position in global network. They adopted k-shell decomposition analysis to obtain ranking index of node importance, and results were more accurate than by using degree and betweenness. "Complexity" [7] is defined as measure metric to compare software network and corresponding random networks, and find the important nodes in network

with PageRank algorithm. Laplacian-based centrality was extended [8] and adopted the idea of PageRank to introduce global connectivity between all pairs of nodes with certain strength. NodeRank [9] was defined to assign a numerical weight to each node in a graph, and measure relative importance of that node in software. The critical nodes ranking results are not unique, which were measured by PageRank and its extended algorithms. ClusterRank algorithm [10] is proposed to identify influential nodes in very large-scale directed networks and it outperforms some benchmark algorithms such as PageRank and degree. Multiple Attribute Fusion (MAF) method [11] is proposed to identify influential nodes, and it gains more information propagation efficiency in different types of networks. Wei et al. [12] 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.

Critical nodes have higher importance degree and greater impact in software execution process. CNM algorithm based on complex network is proposed to mine the critical nodes in software network. In CNM algorithm, we define FID as the measuring metrics of nodes' importance, and sort the nodes by the FID value. Top-k ranking nodes can be regarded as critical nodes.

The remaining paper is organized as follows. Section 2 introduces the foundational definitions. Process of constructing software execution directed networks and CNM algorithm are given in Section 3. Section 4 presents our experimental results and analysis. Section 5 contains concluding remarks.

2. Definitions.

Software functions can be decomposed into reusable elements (such as packages, classes, libraries, interfaces, objects, methods, and compilation units). A software topology network is formed in which we take these elements as nodes and the relationship between the nodes as edges.

The term $G = (V, E)$ is used to denote a graph G, where $V = \{v_1, v_2, \ldots, v_n\}$ represents a set of nodes, $E = \{e_{ij} = (v_i, v_j) : v_i, v_j \subset V\}$ represents a set of edges, and e_{ij} is defined as follows:

$$e_{ij} = \begin{cases} 1 & v_i \to v_j \\ 0 & \text{others} \end{cases}$$

Definition 2.1. *Failure infection: In nodes set V, $\forall v_i$, $v_j \in V$. If there is an edge e from node v_i to node v_j, when a failure happens at node v_j, it may be also infected to node v_i, and node v_i fails too.*

Definition 2.2. *Critical node: According to cascading failures, if node v_j fails, the nodes which can arrive to node v_j by one or more directed edges may also fail. Node v_j is called as critical node.*

Definition 2.3. *Critical nodes metrics (FID): When node v_i fails, FID(i) is the proportion of failure nodes numbers and total nodes numbers in the software network, as shown in Formula (1).*

$$FID(i) = \frac{V_f(i)}{V} \tag{1}$$

$V_f(i)$ represents the numbers of failure nodes which are caused by node v_i in network. V represents the numbers of total nodes in network. We regard FID as the metrics of critical node measure in software execution network.

Definition 2.4. *Level(i). Nodes are classified by FID values. The importance grade of node v_j is called as Level(i). When FID value is greater, then the Level is higher, and the node is more important to the network. The nodes with the same FID value are at the same level.*

3. **Method of Critical Nodes Mining.** The program execution sequence is extracted to create the software execution directed network firstly, and then CNM algorithm is used to calculate the node FID value which helps us to select the critical nodes. In the end, we sort the nodes belonging to the FID value descending and take top-k nodes as critical nodes.

3.1. **Constructing software execution directed network.** In order to accurately measure the importance of nodes in software network, a model is proposed to construct the software execution directed network under Linux environment. The process is described as follows.

a. Using GNU compiler tool chain to collect the traces of function calls when software executes, and put the results in trace.txt file.

b. Using Pvtrace to analyze trace.txt file and generate graph.dot file. In order to describe clearly, Graphviz is used to display visualization of calls relationship between functions.

c. Developing a tool which used java to realize development to convert graph.dot to nodes.txt and edges.txt.

3.2. **Critical node mining algorithm.** CNM algorithm is used to measure the importance of nodes after software execution network generated. We traverse the nodes in depth which can arrive to node v_i by one or more directed edges, and accumulate the numbers. The FID value of node v_i is counted according to Formula (1), and sort in descending order. Finally, we take top-k nodes as critical nodes from the results.

Algorithm 1 Critical Node Mining Algorithm

Input: G = $\langle V, E \rangle$

Output: FID[G.no], Top(k, Level)

```
(1)  Link[G.no] = 0;  Label[G.no] = false;
(2)  Visited[G.no] = false;  InitQueue(Q);
(3)  for (i = 0; i ≤ G.no; i++){
(4)   if (Label[i] = false){
(5)    sum = 0;
(6)    Label[i] = true;
(7)    Visited[G.no] = false;
(8)   if (Visited[i] = false) {
(9)     Visited[i] = true;
(10)     InQueue(Q,i);
(11)     While (Queue_empty(Q) != NULL){
(12)       OutQueue(Q,i);
(13)       for (j = DirectTo(G,i) → OtherDirectTo(G,i))
(14)         if (Visited[j] = false){
(15)           ++sum;
(16)           Visited[j] = true;
(17)           OutQueue(Q,j);
                }
            }
          }
        }
(18)    Link[i] = sum;
      }
(19) FID[G.no] = Link[G.no]/ G.no;
(20) Level (FID[G.no]);
(21) output (FID[G.no], Top(k, Level)).
```

Queue structure is used to traverse the directed graph G in depth. When node v_j arrives to node v_i by a path, sum++ (lines 3-17), and the number of nodes of the path is counted. Then, we can calculate FID value of node v_i according to Formula (1) (lines 18-19). Finally, we take top-k nodes as critical nodes which are sorted by FID value (lines 20-21).

4. Experimental Results and Analysis.

4.1. Datasets.

To check the practical feasibility of our method, we use three software datasets – a version of gzip, tar and cflow. The version numbers are gzip-1.6, tar-1.27 and cflow-1.4. The numbers of functions and function calls relations of each software which are tracked in software execution are shown in Table 1.

Table 1. The numbers of functions and call relations in each software

software datasets	numbers of functions	numbers of call relations
gzip-1.6	48	59
tar-1.27	101	121
cflow-1.4	116	194

4.2. Results and analysis.

We can get the software execution directed network according to the method mentioned in Section 3.1. Then, CNM algorithm is used to mine the important functions of gzip, tar and cflow network graphs. In order to reflect the results more clearly, a comparative analysis is used among Indegree, NodeRank and CNM algorithms.

Table 2. Top-5 levels of cflow software measured by CNM algorithm

V_f	FID	Function name
47	0.4052	deref_linked_list
46	0.3966	linked_list_append
36	0.3103	hash_symbol_hasher, hash_symbol_compare, symbol_is_function
31	0.2672	unlink_symbol
30	0.2586	linked_list_destroy, static_free, lookup, delete_symbol

In cflow software, function deref_linked_list and linked_list_append are used to initialize respectively. As shown in Table 2, function deref_linked_list is called by 47 functions directly or indirectly; likewise, function linked_list_append is called by 46 functions in cflow software execution process. If function deref_linked_list or linked_list_append fails, it can lead to 40.52%, or 39.66% of the functions in network to break down respectively, which causes software on the brink of collapse.

Table 3 shows the top-10 critical nodes of gzip software which is measured by Indegree, NodeRank and CNM methods. In Indegree method, the nodes from no.2 to no.10 have the same indegree value. It is not easy to distinguish the importance of each node. The measurements between NodeRank and CNM methods are almost the same. In NodeRank methods, function read_buffer is considered as the most important node in the network. However, the function read_buffer can cause seven functions to fail. It is less than function copy_block. In CNM algorithm, if the function copy_block has fault, the fault can be transmitted to nine functions, such as ct_init, gzip, treat_file, main, gen_codes, built_tree, flush_block, built_bl_tree and deflate_fast. So the value of FID(copy_block) is 0.1875, namely, if function copy_block gets failure, 18.75% nodes in the network may be caused to fail. Thus, CNM algorithm is better than Indegree and NodeRank methods.

TABLE 3. Critical nodes ranking of gzip software using Indegree, NodeRank and CNM methods

NO.	Indegree	NodeRank	CNM
1	read_buffer	read_buffer	copy_block
2	strlwr	copy_block	gen_codes
3	remove_output_file	_moddi3	read_buffer
4	display_ratio	pqdownheap	pqdownheap
5	gen_codes	bi_reverse	gen_bitlen
6	copy_block	display_ratio	bi_reverse
7	init_block	file_read	display_ratio
8	file_read	ct_tally	scan_tree
9	build_tree	write_buf	build_tree
10	bi_reverse	scan_tree	init_block

TABLE 4. Top-5 levels of tar software measured by CNM algorithm

V_f	FID	Function name
12	0.1176	checkpoint_run, read_header
11	0.1078	_gnu_flush_read, represent_uintmax
10	0.0980	gnu_flush_read, from_header, assign_string
9	0.0882	tar_stat_close, flush_read
8	0.0784	flush_archive

As shown in Table 4, FID of function read_header is 0.1176 which could lead 12 functions to break down. The value of FID(tar_stat_close) is 0.0882, and the value of FID(flush_archive) is 0.0784, but the Indegree and NodeRank strategies cannot measure the importance of function read_header, tar_stat_close, as well as flush_archive. Targeted supports should be provided to these functions in software testing and maintenance; otherwise, they may lead to software failure if these functions break down.

Take top-5 nodes of three software networks which are sorted by FID value. In order to show software infection rates better, we gather statistics of V_f value and calculate its average $\overline{V_f}$, namely, if these functions are under attack, $\overline{V_f}/N \times 100\%$ of functions in software network will be led to failure. The specific infection percentage of three softwares is shown in Figure 1. $\overline{V_f}$(cflow) = 34.66%, $\overline{V_f}$(gzip) = 15.83%. $\overline{V_f}$(tar) = 11.09%. It is helpful to find the failure nodes as soon as possible, and improve software stability and robustness in network.

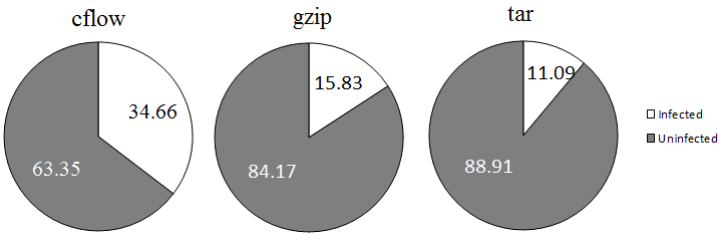

FIGURE 1. Infection percentage

5. **Conclusions.** In order to predict the consequences if one or more nodes fail, we measure the importance of nodes in the software execution network. This paper presents a critical nodes mining (CNM) algorithm based on complex network. FID is defined to identify the critical level of nodes, and the critical nodes are sorted by using FID values.

Top-ranking nodes play important roles in the software network. Ultimately, experimental analysis shows that CNM algorithm can accurately measure the importance of nodes and predict their failure range. However, the results also show that there may be more than one node in the same level. So how to distinguish from each other is one of the most important researches in the future.

Acknowledgment. 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] D. S. Callaway, M. E. J. Newman, S. H. Strogatez and D. J. Watts, Network robustness and fragility: Percolation on random graphs, *Physical Review Letters*, vol.85, no.25, pp.5468-5471, 2000.

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

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

[5] B. A. N. Travençolo and L. da F. Costa, Accessibility in complex networks, *Physics Letters A*, vol.373, no.1, pp.89-95, 2008.

[6] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley and H. A. Makse, Identification of influential spreaders in complex network, *Nature Physics.*, vol.6, no.11, pp.888-893, 2010.

[7] 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.

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

[9] 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.

[10] D. Chen, H. Gao, L. Lü and T. Zhou, Identifying influential nodes in large-scale directed networks: The role of clustering, *Plos One*, vol.8, no.10, 2013.

[11] L. Zhong, C. Gao, Z. Zhang, N. Shi and J. Huang, Identifying influential nodes in complex networks: A multiple attributes fusion method, *Lecture Notes in Computer Science*, vol.8610, no.4, pp.11-22, 2014.

[12] 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 & Its Applications*, vol.420, pp.277-283, 2015.