ICIC Express Letters
Volume **10**, Number **7**, July **2016**

 ISSN 1881-803X
pp. **1683–1689**

IN-RANK: AN ENTROPY-BASED INFORMATIVE NODES MINING METHOD IN COMPLEX SOFTWARE NETWORK

Jiadong Ren[1,2], Yanling Li[1,2,*], Hongfei Wu[1,2]
Yang Liu[1,2] and Peng Zhang[1,2]

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

Received December 2015; accepted March 2016

Abstract. *It is a fundamental issue to find a small subset of influential individuals in a complex network such that they can spread information to the largest scope of nodes in the network. Informative functions in complex software network can lead maximum information propagation scope, and mining such function nodes is important to understand the system's topology structure and information transfer flows. In this paper, a novel top-k informative nodes mining approach based on global information in directed-weighted complex software network is proposed. Firstly, we map functions and call relationships between them as a Directed-Weighted Function Call Network (DWFCN). Secondly, an algorithm path-compute based on Software Executing Path Sequence (SEPS) generated by Depth-First-Search strategy is used to compute the software executing paths and get nodes' information Accessible Set (AS). Thirdly, algorithm Entropy-Compute is proposed to calculate each source node's information entropy in each information transfer flow process. Finally, an Informative Node Rank (IN-Rank) algorithm is put forward to mine most informative nodes. Experimental results show that our new method is accurate and effective.*

Keywords: Software network, Path sequences, Information entropy, Influential nodes

1. **Introduction.** Node's importance analysis now is a hot topic in complex networks. Identifying the most influential "spreaders" in a network is an important step to optimize the use of available resources and ensure the more efficient spread of information [1]. Software network as a kind of complex system displays lots of complex characteristics, if failure or malicious attack occurs in influential nodes, cascading failure will be caused in software systems, which brings about rebellious security issues, so mining influential functions is important for software system's structural analysis and regular maintenance.

Different measures of node's importance were proposed in recent years. Freeman [2] argued that degree centrality indexes the node's activity; betweenness centrality exhibits the node's potential for network control and closeness centrality reflects its communication independence or efficiency. Method of degree centrality is simple, but it lacks relevance. Although betweenness centrality and closeness centrality can effectively identify influential nodes, they are incapable of being applied in large-scale networks due to the high computational complexity. Borgatti [3] conceptualized a typology of centrality measure based on ways that traffic flows through the network, this method considers the global information and performs well, and it is suitable for traffic network but not software network. Li et al. [4] introduced a weighted software network model to represent the structure of object-oriented software and defined an indicator IC to measure the importance of nodes. It needs to get the Extra-Class method reachable set, and this process is excessively time-consuming. Bhattacharya et al. [5] defined a measure called NodeRank

that assigns a numerical weight to each node in a graph, to measure relative importance of node in software, but they did not consider function call into account. He et al. [6] proposed two new measures NMS and RNMS to efficiently evaluate function's importance, but they all need to adjust parameters.

Because paths from a particular node to its destination nodes can be built as sequences, some sequence mining methods were proposed to analyze node's centrality. Tutzauer [7] proposed an entropy-based centrality measure for traffic which propagates by transfer flows along path sequences, this method is classically efficient, but its looped graph model does not work well in software network which is hardly with any loops. Nikolaev et al. [8] developed the idea of Tutzauer, and presented a new, high-utility entropy centrality measure based on a discrete Markovian transfer process in local paths, which is well performed in social network.

Unlike mentioned above, our paper defines the information of network in a different way and widens the application of information entropy into software network. As failure of functions may spread to other functions and causes unmanageable fault in software network, important functions have greater probability to travel fault far in multiple potential directions, namely, have a larger entropy value. We first quantify function calls in software as information transfer process and propose a novel method to measure importance of functions based on information entropy in software network. After extracting function calls as path sequences, we calculate each source node's accessible set, and then algorithm Entropy-Compute and IN-Rank are used to get informative nodes.

The remaining paper is organized as follows. Section 2 gives some basic definitions. Detailed description of our approach and related algorithms are given in Section 3. Experiments in Section 4 show algorithm performances on two open-source software. Conclusion and future work are mentioned in Section 5.

2. Preliminaries.

In a software system, collaborations and calling relationships along paths reflect the system's control flow, and informative functions can be found by analyzing the control flow. In order to accurately measure node's influential degree in spreading information, we propose a model to construct software directed-weighted function call network. First we give some basic definitions in the following.

Definition 2.1. *DWFCN (Directed-Weighted Function Call Network). A DWFCN is defined as* $DWFCN = (V, E, W)$. *Nodes set* V *represents functions and directed edges set* E *represents call relationships between functions, edge* $< u, v >$ *means function* u *calls function* v. W *is the corresponding weights set of* E, *where* $w_{u,v} = \frac{a_{uv}}{k_u^{out}}$ *is probability that a random call at function* u *goes to* v, $a_{uv} = 1$ *if node* u *points to* v, *otherwise,* $a_{uv} = 0$, k_u^{out} *denotes the out-degree of* u.

Definition 2.2. *SEPS (Software Executing Path Sequence). A software executing path sequence is a list of functions denoted as* $< f_1 \to f_2 \to \ldots \to f_{n-1} \to f_n >$, *in which* $f_n \in V$, f_n *is an available node of* f_1 *from sub-path* $< f_2 \to \ldots \to f_{n-1} >$.

Definition 2.3. *AS (Accessible Set). Accessible set is the set of all path-reachable nodes traversed along SEPSs or sub-path of SEPSs from a specific source node.*

Definition 2.4. *INTP (Information Transfer Probability). Suppose call relationship* $a \to b \to c$, *if* a *calls* b *with probability* p_{ab}, b *calls* c *with probability* p_{bc}, *information flows from* a *to* c *will be with probability* $p_{ac} = p_{ab} \times p_{bc}$.

Information Transfer Probability from source node d *to destination node* s *is defined as follows*

$$INTP_{sd} = \sum_{p=1}^{n} Intp_{sd} \tag{1}$$

where n is count of different connected paths from d to s, $Intp_{sd} = \prod_{<i,j>\in E(SEPS)} w_{i,j}$ is Information Transfer Probability on one path from d to s, and E (SEPS) represents all edges in a SEPS.

Example 2.1. *Figure 1 shows a simple DWFCN, $AS(c) = \{d, e, f\}$, from path sequences $\{c \to d, c \to e \to d\}$, $\{c \to e\}$, $\{c \to d \to f, c \to e \to f, c \to e \to d \to f\}$ respectively. We can get $INTP_{cd} = w_{cd} + w_{ce} \times w_{ed} = 3/4$, $INTP_{ce} = w_{c,e} = 1/2$, $INTP_{cf} = w_{c,d} \times w_{d,f} + w_{c,e} \times w_{e,f} + w_{c,e} \times w_{e,d} \times w_{d,f} = 1$.*

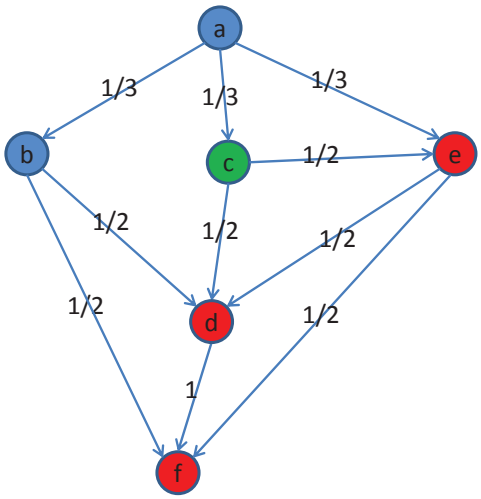


FIGURE 1. A simple DWFCN

Definition 2.5. *IESN (Information Entropy of Source Node) A node's information entropy is defined as follows*

$$IESN(i) = -\sum_{j=1}^{n} P_{ij} \log(P_{ij}) \tag{2}$$

where $n = |AS(i)|$, P_{ij} is the proportion of i's total INTP volume that points to j, namely

$$P_{ij} = \frac{INTP_{ij}}{\sum_{j=1}^{n} INTP_{ij}} \tag{3}$$

Definition 2.6. *IN (Informative Node) Function node i is an informative node if $IESN(i) \geq p$, p is a user-defined threshold.*

Informative Node has a larger ability of diffusing information than nodes with a lower IESN.

3. **IN-Rank: Mining Informative Nodes in Directed-Weighted Software Network.** Firstly, functions and relationships between them will be extracted from software source code, and then we map them as DWFCN. Secondly, algorithm Path-Compute based on path sequences obtained by Depth-First-Search (DFS) is used to get AS of each node. Thirdly, algorithm Entropy-Compute is proposed to obtain every source node's information entropy. At last, informative nodes will be mined by IN-Rank.

Algorithm 1 Path-Compute

Input: Directed-Weighted Function Call Network (DWFCN)
Output: F($i \rightarrow j$, $INTP_{ij}$)//$j \in AS(i)$, $INTP_{ij}$ is the probability that i's information spreads to j

```
1: call Algorithm 2 to get SEPS-DB
2:   for each path in SEPS-DB
3:     initialize node i=parh.firstNode, temp=1.0, tempSource=i, seps=i, INTP_ij=0.0
4:       for (i.next!=null)
5:         j=i.next;
6:         seps=seps+j;
7:           if (seps has not been computed)
8:             INTP_ij+=w_tempSource,j*temp temp=w_tempSource,j*temp
9:           end if
10:        tempSource=j; j=j.next
11:      end for
12:    GetFunction(path-i)
13:  end for
14: return F (i → j, INTP_ij)
```

Algorithm 2 GetSEPS-DB

Input: Directed-Weighted Function Call Network (DWFCN)
Output: SEPS-DB

```
1: initialize SEPS=rootnode
2:   if (rootnode.child is not null) then
3:     for each child in set of rootnode.children
4:       SEPS=SEPS+child
5:       GetSEPS-DB(child)
6:     end for
7:    else
8:     SEPS-DB.add(SEPS)
9:  end if
10: return SEPS-DB
```

Algorithm Path-Compute firstly calls GetSEPS-DB to get all SEPS (line 1). Then each path is computed to get every node's accessible destination node and corresponding INTP value recursively (line 2-line 13), and function F($i \rightarrow j$, $INTP_{ij}$) is obtained to keep this result (line 14). Algorithm GetSEPS-DB generates all SEPS recursively from DWFCN based on DFS strategy.

Algorithm 3 Entropy-Compute

Input: A node i in DWFCN
Output: IESN(i)

```
1: initialize IESN(i)=0.0
2:   for each node j in F (i → j, INTP_ij)
3:     calculate P_ij=INTP_ij / sum_{j=1}^{|F|} INTP_ij //calculate P_ij, make it meet the Entropy calculation conditions
4:     calculate information I=−P_ij log P_ij
5:     IESN(i)=IESN(i)+I
6:  end for
7: return IESN(i)
```

Algorithm Entropy-Compute reveals how the information entropy of a source node is calculated (line 1-line 5) based on Formulas (2) and (3).

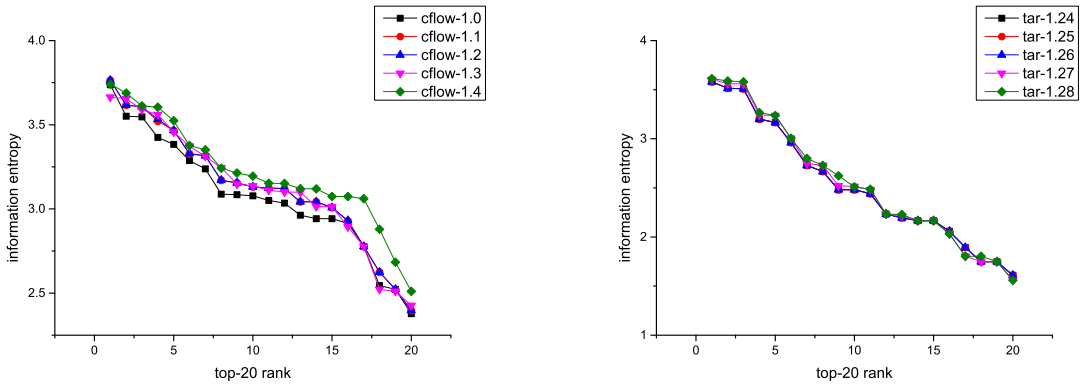
Algorithm 4 Informative Node Rank (IN-Rank)

Input: Directed-Weighted Function Call Network (DWFCN)
Output: A sorted list storing the nodes' informative entropy

```
Path-Compute
    for each node i F(i → j, INTP_ij)
      temp=Entropy-Compute(i)
      insert (i, temp) to list L
    end for
  sort list L in descending order
return L
```

Example 3.1. *Taking Figure 1 for example, in the DWFCN, $AS(c) = \{d, e, f\}$, according to Algorithm 1, we can get $F(c \to d) = 3/4$, $F(c \to e) = 1/2$, $F(c \to f) = 1$. Then according to Formula (3), $P_{cd} = 1/3$, $P_{ce} = 2/9$, $P_{cf} = 4/9$, $IESN(c) = 0.460724$ according to Algorithm 3.*

4. **Experiment Analysis.** Experiment will be tested on two open-source software cflow and tar, which are available from http://gnu.april.org/software/cflow and http://gnu.april.org/software/tar/tar.html. We choose the latest five versions of each software. Experiment is conducted on 64 bit Windows 7 ultimate, Xeon CPU E5-2603 @1.80GHz, 8G Memory and Ubuntu14.04. We will compare our method with SN-KNN [9] and Degree.



(a) Entropy rank of *cflow* (b) Entropy rank of *tar*

FIGURE 2. Entropy rank of cflow and tar

Figure 2 shows the version tendency of source node's information entropy of cflow and tar. We can get some implicit information from Figure 2.

- The curves have similar variation tendency and top-k information entropy values of different versions, which meets the fact that new version will not change too much considering the system's stability.
- Each new version has a slight higher entropy value than former one; it indicates that new software's information flow is a little more active.
- More attention should be paid on functions that have a higher entropy value such as top-10, because a higher entropy value means a higher possibility of spreading information far.

Table 1 shows part of rank results of cflow versions and Table 2 shows part of rank results of tar versions, and some patterns can be found.

- The rank of functions in each software version is steady with little range. In Table 1, function parse_typedef's rank ranges from 9 to 13, and function yyparse always ranks 6.

- Due to the steady rank rules, we may predict that there will not be a great rank change of certain function nodes in the next new version. For example, in Table 2, function dump_file0 and create_archive will still be the most two important nodes in the next version, and create_archive may rank NO.1.

Table 1. Rank of each cflow version by IN-Rank

function_name	$V-1.0$	$V-1.1$	$V-1.2$	$V-1.3$	$V-1.4$
main	1	1	1	1	1
func_body	2	2	2	4	3
maybe_parm$_l$ist	3	3	3	2	2
parse_dcl	4	4	4	3	4
parse_variable_declaration	5	5	5	5	5
yyparse	6	6	6	6	6
expression	7	7	7	7	7
dcl	8	8	8	9	10
parse_typedef	9	12	12	12	13
fake_struct	10	9	9	10	9

Table 2. Rank of each tar version by IN-Rank

function_name	$V-1.24$	$V-1.25$	$V-1.26$	$V-1.27$	$V-1.28$
*dump_file*0	1	1	1	2	2
create_archive	2	2	2	1	1
*dump_dir*0	3	3	3	3	3
main	4	4	4	3	4
dump_regular_file	5	5	5	5	5
dump_hard_link	6	6	6	6	6
start_header	7	7	7	8	8
dump_file	8	8	8	7	7
_open_archive	9	9	9	9	9
dump_dir	10	10	10	10	11

Table 3. Top 15 informative functions of tar-1.28

function_name	*IN-Rank*	*SN-KNN*	*Degree*
create_archive	1	3	3
*dump_file*0	2	4	1
*dump_dir*0	3	6	5
main	4	1	6
dump_regular_file	5	7	4
dump_hard$_l$ink	6	8	7
dump_file	7	2	12
start_header	8	10	2
_open_archive	9	9	8
close_archive	10	18	9
dump_dir	11	5	13
write_header_name	12	12	24
open_archive	13	9	8
write_eot	14	26	14
write_short_name	15	19	21

We compare top-15 functions in our method (IN-Rank) with SN-KNN and Degree using tar-1.28 in Table 3. We can find some information in the following.

- Results of IN-Rank and SN-KNN are 80% similar because they both take out-degree into account, but SN-KNN is a semi-global method only considering node's neighbor and neighbor's neighbor, so function dump_file ranks 2 in SN-KNN just ranks 7 in IN-Rank, because the transfer path from dump_file stops after moving two steps.
- There is no correlation between the importance rank and the out-degree of a function node. For example, in Table 3, function start_header has a very high out-degree but lower ability of spreading information than dump_file with a lower degree.

5. **Conclusions and Future Work.** Although many efforts have been made to measure node's importance in spreading information, till now we see no researches towards defining entropy measure that give better applicability and performance for software. In order to accurately and globally reflect function's information diffuse ability, this paper proposes a novel approach with information entropy by path calculation to measure the importance of functions in software function call network. First we map process of software calling to Directed-Weighted Function Call Network. Functions are defined as nodes, relationships of calls between functions are abstracted as edges, and call probability of function is set to weight. Then path sequences are extracted to calculate each source node's information entropy based on processed weight. Finally, algorithm IN-Rank is proposed to mine the most informative nodes with high ability of diffusing information. Experimental results indicate that the proposed approach is effective. Our future work will focus on tracing software's dynamic execution process to find more patterns about influential nodes.

Acknowledgment. This work is supported by the National Natural Science Foundation of China under Grant No. 61170190, No. 61472341 and the Natural Science Foundation of Hebei Province China under Grant No. F2013203324, 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] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley and H. A. Makse, Identification of influential spreaders in complex networks, *Nature Physics*, vol.6, pp.888-893, 2010.

[2] L. C. Freeman, Centrality in social networks: Conceptual clarification, *Social Networks*, vol.1, pp.215-239, 1979.

[3] S. P. Borgatti, Centrality and network flow, *Social Networks*, vol.27, no.1, pp.55-71, 2005.

[4] D. Li, B. Li and W. Pan, Ranking the importance of classes via software structural analysis, *Future Communication, Computing, Control and Management*, vol.141, pp.441-449, 2012.

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

[6] H. He, J. Wang and J. Ren, Measuring the importance of functions in software execution network based on complex network, *International Journal of Innovative Computing Information and Control*, vol.11, no.2, pp.719-731, 2015.

[7] F. Tutzauer, Entropy as a measure of centrality in networks characterized by path-transfer flow, *Social Networks*, vol.29, no.2, pp.249-265, 2007.

[8] A. G. Nikolaev, R. Razib and A. Kucheriya, On efficient use of entropy centrality for social network analysis and community detection, *Social Networks*, vol.40, pp.154-162, 2015.

[9] J. Ren, H. Wu and T. Yin, A novel approach for mining importance nodes in directed-weighted complex software network, *Journal of Computational Information Systems*, vol.11, no.8, pp.3059-3071, 2015.