Computer Communication & Collaboration (Vol. 5, Issue 2, 2017)
ISSN 2292-1028(Print) 2292-1036(Online)
Submitted on March 25, 2017
DOIC: 2292-1036-2017-02-001-85

The Network Complexity of Small-scale Software Systems

Dong Yan*, Junsheng Jin

College of Mechanical Engineering, Shanghai University of Engineering Science,

201620 Shanghai, China

E-mail: * yd_email@aliyun.com

Abstract: Large-scale complex networks in information technology area have been shown to display a scale-free distribution and a small-world property. However, the network complexity of small-scale software systems are studied seldom. This paper focuses on collaborative relationships among custom classes in small-scale software systems to study their network complexity. It is found that small-scale software systems written in Java as networks follow a stretched exponential distribution by empirical analysis. But collaborative relationships among custom classes don't have a distinct small-world property. In addition, the scale of a realistic system has a minor influence on network complexity.

Keywords: Small-scale Systems, Stretched Exponential Distribution, Small-world Property

1. Introduction

Software architecture refers to the fundamental structures of a software system, the discipline of creating such structures, and the documentation of these structures. Each structure comprises software elements, relations among them, and properties of both elements and relations, along with rationale for the introduction and configuration of each element [1]. Complexity measures of software structures have been carried out in software engineering in the last decades.

Watts and Strogatz first found that many biological, technological and social networks are no completely regular networks or completely random networks but small-world networks with a high clustering and a small characteristic path length [2]. Many networks such as the semantic metadata distribution [3] and the clustering formation and routing in wireless network [4] can be characterized by small-world pattern. Another outstanding contribution in complex networks is the scale-free power-law distribution of evolution of networks, namely the BA model, which is brought forth by Barabasi and Albert when they studied the world wide web [5]. Consequently, it is found that collaboration graphs of software written in C/C++ also reveal scale-free and small-world properties [6]. For a software system written in Java as a directed network, the relationships that a class imports other classes tends to an exponential distribution, while the relationships that a class is imported by other classes and the compound import relationships of the former two reveal the power-law behavior [7]. However, Laherrere and Sornette proposed that the stretched exponential distribution describes very well the connectivity of many networks in nature and economy [8]. It is found that the Linux Gentoo network [9], the user posting behavior in online social networks [10] and the object reference ranks of media workloads [11] follows stretched exponential distributions instead of power-law distributions. The wealth distribution of the richest persons taken from the "rich lists" provided by business magazines like Forbes is consistent with some rivals like the

log-normal or stretched exponential distribution [12].

The existing research contributions have shown that network complexity of large-scale software systems involves the scale-free distribution of connectivity and the small-world property of collaborability among classes [2-7,13]. But are small-scale software systems identical in network complexity with large-scale systems? In addition, most programmers generally face small-scale software modules belonging to a large-scale system. Therefore, this paper chooses to investigate the network complexity of some small-scale open-source Java software systems or modules.

2. Kits of a Java system

Java is a popular general-purpose computer programming language that is concurrent, class-based, object-oriented (OO), and specifically designed to have as few implementation dependencies as possible [14]. A Java software system consists of many collaborative classes, such as Java Development Kit (JDK), third party development kits and custom classes as shown in Figure 1. The custom classes implement all kinds of custom functions or tasks, which are developed by programmers. Software engineers generally depict organization relationships of custom classes through class diagrams. The stereotype keyword 'import' is introduced into class files to specify the class-level relationships of a Java system. This paper focuses on the class-level relationships of custom classes. Furthermore, three open-source Java software systems, 'webauction', 'OnlineStore' and 'Chess', in addition to three modules, 'internetbeans', 'jelly' and 'lucene' are chosen to investigate the network complexity of small-scale software systems. Scales of six systems are shown in Table 1.

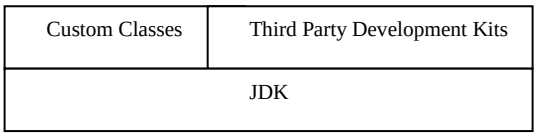

Figure 1 Kits of a Java software system

Table 1 Scales of six Java software systems or modules

	webauction	OnlineStore	Chess	internetbeans	jelly	lucene
Number of nodes	32	36	60	100	199	303
Average degree	6.19	5.58	13.15	5.68	7.36	10.73

3. Analysis of network complexity

3.1 Degree distributions of network connectivity

At first, the degree distributions of realistic small-scale software systems are investigated. In general, if classes among a system are abstracted as nodes, and if relations among classes are abstracted as edges, then the software system is abstracted as a network. The degree of each class is the number of its edges linking to other classes. The frequency of a degree is the number of the nodes with the same degree. The probability of a degree is the proportion of its frequency to the number of all nodes. Therefore, a probability distribution can be thought of as providing the probability of occurrence of different possible degrees.

Probability distributions are generally divided into two classes: a discrete probability

distribution and a continuous probability distribution. Figure 2 shows the discrete probability distributions of six different software systems mentioned above. Obviously, six graphs don't resemble each other, so that it seems impossible that they are fitted by the same function. But it would be better if a continuous probability distribution function can describe the probability of any given degree value. Fortunately, it is found that the stretched exponential distribution can be used to solve this problem. Moreover, the discreteness of degree distributions decreases along with the growth of system scales.

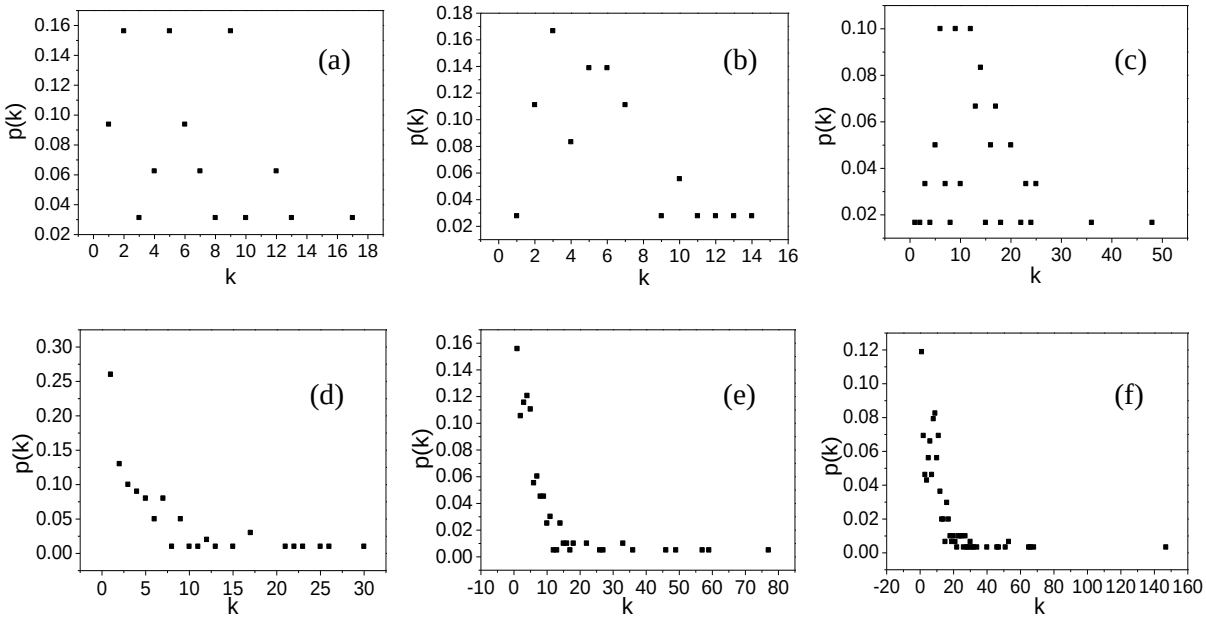

Figure 2 Discrete degree distributions of six Java software systems/modules: (a) webauction, (b) OnlineStore, (c) Chess, (d) internetbeans, (e) jelly, (f) lucene.

3.2 Stretched exponential distribution

In 1998, Laherrère and Sornette proposed the stretched exponential distribution,

$$p(k) = c\frac{k^{c-1}}{k_0^c}\exp(-(k/k_0)^c). \quad (1)$$

It is an alternative model for probability distribution functions. The exponent c quantifies the fatness of the tail of a stretched exponential distribution, the smaller the exponent, the fatter the tail. The parameter k_0 is the characteristic scale of a mean value. To fit properly graphs of Figure 2, we need to bin the data into exponentially wider bins by the cumulative distribution function (CDF) $P_c(k)=\int_k^{+\infty} p(k)dk$, as shown in Figure 3. The CDF of the stretched exponential distribution $p(k)$ can be written as

$$P_c(k)=\exp(-(k/k_0)^c),\ (0<P_c(k)<=1), \quad (2)$$

which further leads to

$$\ln(-\ln(P_c(k)))=c\ln(k)-c\ln(k_0),\qquad (0<P_c(k)<1). \quad (3)$$

As $P_c(k) = 1$, there is a hop in Eq. (3), so $P_c(k) = 1$ is disregarded. If there are enough statistical data from an example, the effect of that disregard can be ignored.

Let $Y = \ln(-\ln(P_c(k)))$, $X = \ln(k)$, and $B = -c\ln(k_0)$. Then $Y = cX + B$, such that $Y \sim X$. Consequently, the scattered points in each of plots in Figure 2 can be fitted by the CDF, that's Eq. (3). Figure 3 shows the cumulative degree distributions of six examples on log-loglog scales. The actual values of parameters c and x_0 obtained by fitting cumulative distributions of six examples are shown in Table 2, where parameter R is the relative coefficient, and SD is the standard deviation. The research contributions in the last decades have shown that the connectivity of large-scale software systems follows the scale-free distribution [6-7], whereas that of small-scale Java software systems does the stretched exponential distribution.

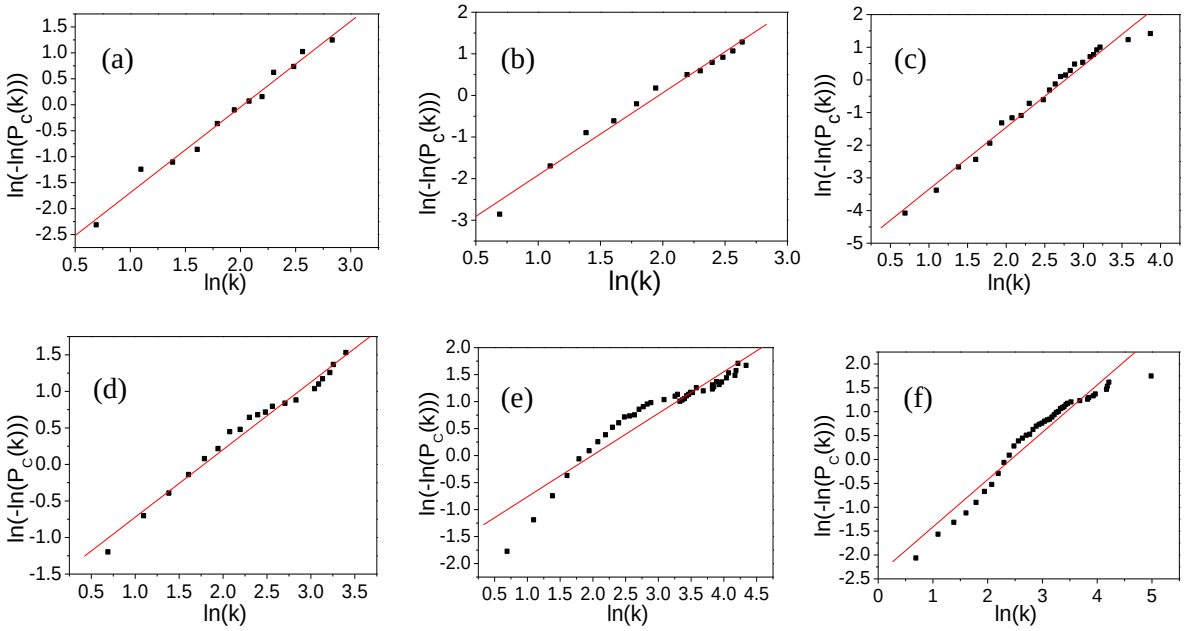

Figure 3 Log-loglog scale graphs of the cumulative degree distributions of six Java software systems/modules mentioned above.

Table 2 Parameter values obtained by fitting cumulative distributions

	c	k_0	*R*	*SD*
webauction	1.65105	7.580167	0.99176	0.14142
OnlineStore	1.9789	7.16078	0.9914	0.17092
Chess	1.89701	15.887642	0.98974	0.22202
internetbeans	0.92445	5.934534	0.99027	0.1021
jelly	0.7721	7.300399	0.95123	0.23488
lucene	0.99082	11.303 273	0.96825	0.23318

3.3 Analysis of small-world property

The small-world property of a network is quantified by its clustering coefficient C and characteristic path length L. The clustering coefficient C_i of a node i with degree k_i in the network is the fraction

between the actual number e_i of edges between its neighbors and the total number of possible edges between those neighbors, that's

$$C_i = \frac{2e_i}{k_i(k_i - 1)}. \quad (4)$$

The clustering coefficient C of the whole network is the average value of the clustering coefficients of all nodes in the network, thus $C = n^{-1}\sum_i C_i$, where n is the number of all nodes.

The characteristic path length L_i of node i is defined as the average value of steps along the shortest paths for all possible pairs of node i and other nodes. Thus

$$L_i = \frac{1}{n-1}\sum_j l_{ij}, \quad (5)$$

where l_{ij} is the shortest path length between nodes i and j in the network. The characteristic path length L of the network is the average value of characteristic path lengths of all nodes, that's $L = n^{-1}\sum_i L_i$.

The small-world property is quantified by comparing clustering coefficient C_{real} and path length L_{real} of a realistic network to an equivalent random network with same degree on average [15]. The clustering coefficient of a random network with scale n and average node degree $\langle k \rangle$ is given by $C_{rand} = \frac{< k >}{n}$, while its characteristic path length is $L_{rand} = \frac{\ln(n)}{\ln(\langle k \rangle)}$. If $C_{real} \gg C_{rand}$ and $L_{real} < L_{rand}$, the network can be considered as a small-world network.

Table 3 shows the actual values of clustering coefficients and characteristic path lengths for six realistic Java systems and their equivalent random networks. Obviously, $C_{real} < C_{rand}$ for systems 'Chess' and 'webauction', and $L_{real} > L_{rand}$ for six systems do not satisfy $C_{real} \gg C_{rand}$ and $L_{real} < L_{rand}$. Thus six realistic small-scale software systems do not satisfy small-world property owing to their small clustering coefficients or large characteristic path lengths.

Table 3 Clustering coefficients and characteristic path lengths of realistic systems, compared with their equivalent random networks

	C_{real}	*Operator*	C_{rand}	L_{real}	*Operator*	L_{rand}
webauction	0.1489	<	0.19	2.7491	>	1.9
OnlineStore	0.2311	>	0.16	3.6241	>	2.08
Chess	0.1798	<	0.22	3.2462	>	1.59
internetbeans	0.2266	>	0.0568	2.8283	>	2.651
jelly	0.2875	>	0.03697	2.825	>	2.652
lucene	0.2953	>	0.0354	2.774	>	2.4075

According to Table 3 and Figure 4, clustering coefficients of realistic systems and characteristic path lengths of their equivalent random networks tend to increase with their scales, whereas characteristic path lengths of realistic systems and clustering coefficients of their equivalent random

networks tend to decrease with scales. Therefore the scale of a realistic system has a minor influence on its small-world property, although it is mainly dominated by evolution mechanisms.

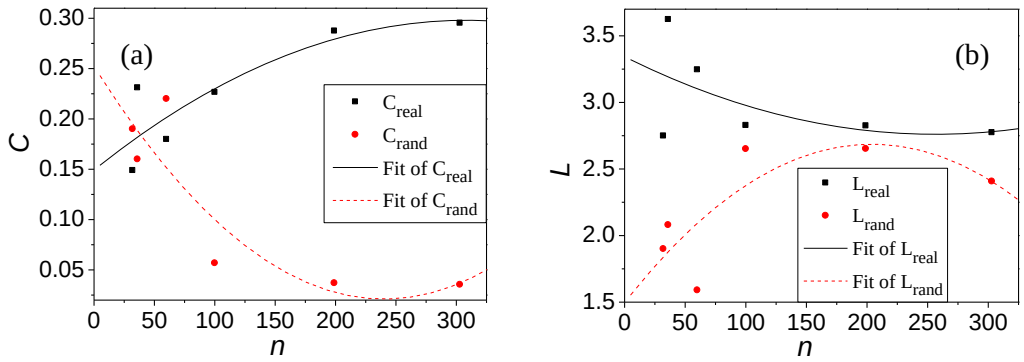

Figure 4 Clustering coefficients (a) and characteristic path lengths (b) of realistic systems versus their scales, compared with their equivalent random networks.

4. Conclusions

In this paper, we have examined realistic small-scale Java software systems to discover their network complexity. The investigation shows that small-scale Java software systems follows the stretched exponential distribution on network connectivity, but does not reveal a small-world property. Obviously, small-scale software systems are different from large-scale ones on network complexity which generally reveal a scale-free behavior and a small-world property. The probable linking relations among classes in a small-scale software system can be simulated, forecasted and estimated by the stretched exponential distribution. In addition, the scale of a realistic system has a minor influence on its network complexity, such as the degree distribution and the small-world property.

References

[1]. Wikipedia (2017), "Software architecture", [Online] Available at: https://en.wikipedia.org/wiki/Software_architecture (April 12th, 2017).

[2]. Watts, D.J., and Strogatz S.H. (1998), "Collective dynamics of 'small-world' networks", *Nature*, 393(4): 440-442.

[3]. Leist, A., and Hawick K.A. (2009), "A Small-World Network Model for Distributed Storage of Semantic Metadata", *7th Australasian Symposium on Grid Computing and e-Research (AUSGRID 2009)*,Wellington, New Zealand, 2009:49-56.

[4]. Dong, Z., Wang, Z., Xie, W., Emelumadu, O., Lin, C.B., and Rojas-Cessa R.(2016), "An Experimental Study of SmallWorld Network Models for Wireless Networks", *Journal of Cyber Security*, 4(4):259–278, doi: 10.13052/jcsm2245-1439.442.

[5]. Barabási, A.L., and Albert R. (1999), "Emergence of scaling in random networks", *Science*, 286(15): 509-512.

[6]. Myers R. (2003), "Software systems as complex networks: Stricture, function, and evolvability of software collaboration graphs", *Physical Review E*, 68(4): 046116(15).

[7]. Yan, D., Qi, G.N., and Gu X.J. (2006), "The complexity nature of large-scale software systems", *Chinese Physics*, 15(11): 2489-2495.

[8]. Laherrere, J., and Sornette D. (1998), "Stretched exponential distributions in nature and economy: "fat tails" with characteristic scales", *Euro. Phys. J. B*, 2(4): 525-539.

[9]. Zheng, X., Zeng, D., Li, H., and Wang F. (2008), "Analyzing open-source software systems as complex networks", *Physica A*, 387(24): 6190–6200.

[10]. Saburo, S., Akihiro, K., and Kohei A. (2009), "Analyzing Patterns of User Content Generation in Online Social Networks", *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, 2009: 369-378 .

[11]. Guo, L., Tan, E., Chen, S., Xiao, Z., and Zhang X. (2008), "The Stretched Exponential Distribution of Internet Media Access Patterns", *Twenty-Seventh ACM Symposium on Principles of Distributed Computing*, PODC 2008, Toronto, Canada, August. DBLP, 2008:283-294.

[12]. Brzezinski M. (2013), "Do wealth distributions follow power laws? Evidence from "rich lists"", *Physica A: Statistical Mechanics & Its Applications*, 406(406):155-162.

[13]. Li, H., Hao, L., and Chen R. (2016), "Multi-Level Formation of Complex Software Systems", *Entropy*, 18(5):178(25), doi:10.3390/e18050178.

[14]. Wikipedia (2017), "Java(programming_language)", [Online] Available at: https://en.wikipedia.org/wiki/Java_(programming_language) (April 8th, 2017).

[15]. Humphries, M.D., Gurney, K., and Prescott T.J. (2006), "The brainstem reticular formation is a small-world, not scale-free, network", *Proc. Roy. Soc. B*, 273(1585): 503–511.