Modeling Open Source Software Bugs with Complex Networks

Cheng Nie[1], Daniel Zeng[1,2], Xiaolong Zheng[1], Fei-Yue Wang[1], Huimin Zhao[3]
[1] The Key Lab of Complex Systems and Intelligence Science, Institute of Automation, Chinese Academy of Sciences, China
[2] Department of Management Information Systems, The University of Arizona, USA
[3] Sheldon B. Lubar School of Business, University of Wisconsin-Milwaukee, USA

***Abstract*—A major concern in open source software development is regarding bugs emerging during the life circle of a software project. Understanding the topological characteristics of interrelated bugs can provide useful insights into software project management and potentially facilitate the development of new complex network models. In this paper, we analyze the bug network in Gentoo Linux, one of the most popular Linux distributions. We model software bugs as nodes and the blocking relationships among them as edges. As the resulting Gentoo bug network cannot be adequately explained by some commonly-used complex network models, we propose a new model, which can better explain this network.**

I. INTRODUCTION

Complex networks research has attracted a lot of attention from researchers in a variety of fields [1-2]. Examples of complex networks that have been studied include the World Wide Web [3-7], citation networks [8], and collaboration networks among movie actors [9], scientists [10] , or authors of Wikipedia [11]. Some recent work has also applied network analysis to such domains as music [12] and transportation systems [13-14].

Software systems represent an important kind of man-made systems, which could be investigated using the framework of complex network analysis [15-16]. Due to the popularity of open source software and the free access to the software source code, much work has been conducted on open source software from the perspective of software engineering [17-23]. However, research on analyzing open source software from the network point of view is still rather limited.

As software systems grow more complex, bugs become more common and more difficult to fix. Programmers may even have to spend more time and effort finding and fixing bugs than writing new code. The National Institute of Standards and Technology concluded that software bugs cost the US economy an estimated $59 billion annually, which constitutes 0.6 percent of the gross domestic product [24].

Understanding the relationship among software bugs would potentially lead to better software project management practice and increased software reliability. In this paper, we analyze the bug network in Gentoo Linux, one of the most popular Linux distributions, treating bugs as nodes and the blocking relationships among the bugs as edges. Based on findings from some empirical studies on the Gentoo bug network, we develop a new model to simulate the growth process of this network. Experimental results show that our model can better explain the Gentoo bug network than some commonly-used existing models.

The rest of the paper is organized as follows. In Section II, we present some background of open source software development. In Section III, we provide a survey of some related work. In Section IV, we describe the data we have collected and the preprocessing step. Next, we show our empirical study results in Section V. We then present our new model and computational study in Section VI. Finally, we summarize our findings and discuss possible future research directions in Section VII.

II. BACKGROUND

Open source software projects need users and developers to help improve the quality of the developed software so that every user can enjoy bug-free packages. Generally, an open source software project needs two systems to manage the development process: a *version control system* and a *bug tracking system*. Version control systems, such as CVS and SVN, are used to share source code files among developers and users. Bug tracking systems, such as Bugzilla and ITracker, are used to track the bugs that emerge during the life circle of a project. During the development process, a large amount of data is recorded in the version control systems and bug tracking systems. Such data provide valuable resources for scientific research.

During the development of an open source software package, the source code is constantly changing. The general procedure for bug resolution can be explained informally as follows. First, a bug is reported into a bug repository when a user finds the software is not performing correctly or when a user thinks the software can be enhanced by adding some new features. The former situation results in a traditional bug, meaning a software flaw, while the latter situation indicates an enhancement request [25]. Here we use the term "bugs" to refer to traditional bugs as well as enhancement requests. The reported bug will then emerge on the bug-to-fix list of the developers, who will contribute to fix the bug. Developers would commit the new source code to the version control system after they get the bug fixed. An interesting phenomenon during the debugging process is that some reported bugs are interrelated with each other. For example, when debugging, a developer may find that a bug cannot be fixed until another bug has been fixed first. The developer would then add a blocking relationship between these two bugs, i.e., bug A blocking bug B means that fixing bug B requires fixing bug A first.

The Gentoo Linux project uses CVS and SVN at the same time to manage the source code files, while using Bugzilla as the bug tracking system. We can get information about the bugs and the blocking relationships among them from the Gentoo Bugzilla system [26].

978-1-4244-7119-5/10/$26.00 ©2010 IEEE

III. Related Work

The "open" characteristic of open source software has facilitated profound research in software related areas. Software engineers are interested in the possibility of automatic bug fixing [17] and the error proneness of a software component or an individual developer [18]. Viewing bug reports as text messages has also provided some insights into software engineering, including predicting the time needed to fix a bug [21], duplicate bug detection [22], and locating potential bugs [23].

Analyzing open source software systems from the perspective of complex networks is an emerging field, which has not yet received wide-spread attention. Myers investigated the software collaboration graphs in several open source software projects and found them to form scale-free and small-world networks [15]. Zheng *et al.* analyzed the package dependency network in open source software and discovered that the probability of a new package connecting to an existing one depends not only on the degree but also on the age of the existing package [16]. A major distinction of our research from these past studies is that we apply network analysis in another important aspect, namely, open source software bugs.

IV. Data Preprocessing

Gentoo Bugzilla [26] is the bug tracking system of the Gentoo Linux project, storing all the reported bugs of the project. The first bug emerged in the repository on January 4, 2002. From then on, the bug repository keeps on growing with more and more bugs reported by geographically dispersed developers and users all over the world. As of January 2009, when we collected the data, 220,576 bugs had been reported. Information about each bug is composed of mainly two parts, an XML file and an activity log. A bug's XML file records the bug *ID*, the *reporter* of the bug, the *time* when it is reported, and a *description*. The activity log of a bug records *who* has changed *what* and *when*.

Reported bugs are connected if developers found relationships among them, for example, bug A cannot be fixed until bug B has been fixed first. We treat bugs as nodes and their relationships as undirected edges in a complex network. Fortunately, we can acquire the exact time of nodes' and edges' arrival by analyzing the XML files and activity logs. Of the 220,576 bugs in the collected dataset, 7,386 (3.3%) bugs with 7,884 edges formed the largest connected component, which constitutes our focus of study. We refer to this network as the *Gentoo bug network*.

V. Empirical Study

A. Growth of nodes and edges

The nodes and edges kept on growing during the 84 months from January 2002 to January 2009. We plot the number of edges versus the number of nodes in Fig. 1. Each point in Fig. 1 corresponds to (N_i, M_i), where N_i is the number of nodes and M_i is the number of edges in the i-th month.

The ratio between the number of edges and the number of nodes in the Gentoo bug network can be viewed as roughly a constant. Thus, we can safely assume that the edges' increase resulted from the same number of new nodes is about the same (this will be further discussed in Section VI).

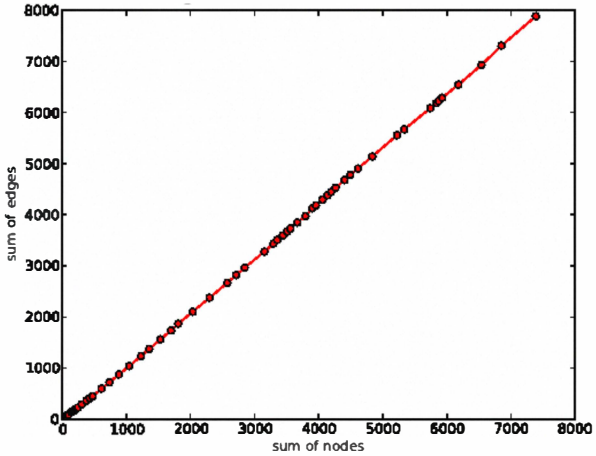


Fig. 1. Increase in the number of edges versus increase in the number of nodes in the Gentoo bug network.

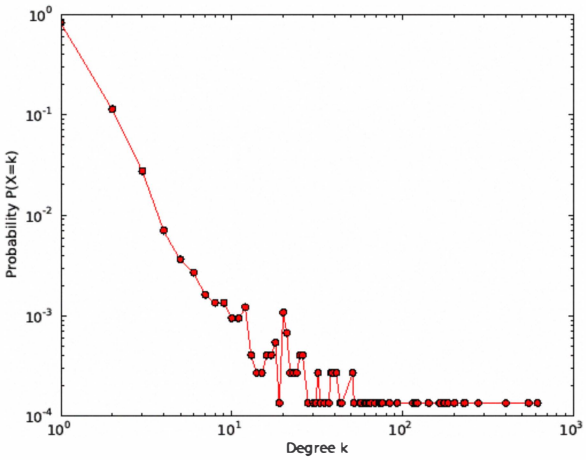


Fig. 2. The degree distribution of the Gentoo bug network on a log-log scale.

B. Degree distribution

Degree distribution $P(k)$ is also useful in characterizing complex networks. It denotes the probability that a randomly selected node has k edges. Power-law distribution ($P(k) \sim k^{\alpha}$) is a dominant feature of many real-world networks [6]. Such networks are said to be scale-free. We plot the degree distribution of the Gentoo bug network on a log-log scale in Fig. 2. It shows a heavy-tail distribution with a lot of fluctuations in the tail. This signifies that the Gentoo bug network does not follow a power-law distribution often found in other related work.

C. Joint degree distribution

The joint degree distribution $p(d_1, d_2)$ captures information of which nodes are connected to which others. It is defined as the relative frequency with which the two nodes at the end of an arbitrarily picked edge have a given pair of degrees. As the bug network is undirected, we first sort the nodes in all edges in the network so as to have the form (v_1,v_2), where the degree of node v_1 is no more than that of v_2. Then, we get the joint degree distribution by the following definition [27]:

$$p(d_1, d_2) = \begin{cases} \frac{1}{2} \times \frac{\#(d_1,d_2)}{M} & d_1 \neq d_2 \\ \frac{\#(d_1,d_2)}{M} & d_1 = d_2 \end{cases} \quad (1)$$

where M denotes the total number of edges and $\#(d_1, d_2)$ is the number of edges with degree pair (d_1, d_2).

Fig. 3 shows the joint degree distribution of the Gentoo bug network. There is a noticeable tendency for high-degree nodes to be connected to low-degree nodes, while nodes with similar degrees do not tend to be connected with one another in the Gentoo bug network.

D. Clustering Coefficient

The degree k_i of a node i measures the number of neighbors of this node, and the clustering coefficient of a node C_i is defined as $2e_i/(k_i*(k_i-1))$, where e_i is the number of edges among the k_i neighbors of node i. The clustering coefficient C of a network is the average of C_i over all nodes in the network. Note that the clustering coefficient for a node with degree zero or one is set to zero. The clustering coefficient of a random graph (i.e., a graph in which nodes are randomly connected) with 7,386 nodes and 7,884 edges would be about 0.00029 [1]. However, the clustering coefficient of the Gentoo bug network is 0.02199, which is 76 times that of a random graph with the same number of nodes and edges. This shows that the Gentoo bug network significantly deviates from a random graph.

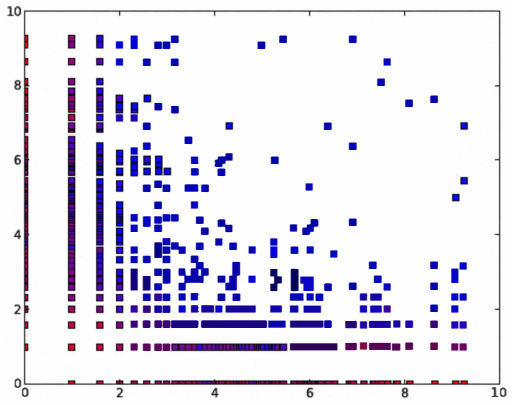


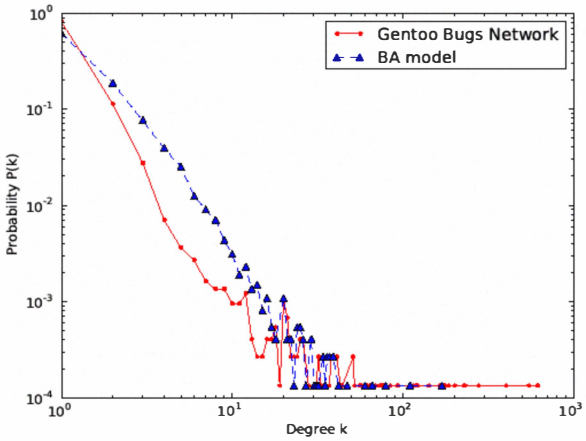
Fig. 3. Image representation of the joint degree distribution. Colors range from blue (low frequency) to red (high frequency), with white indicating areas with no data.

VI. Model Development

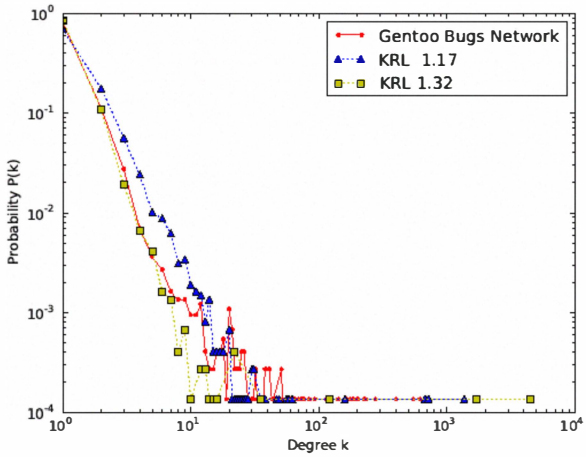
As shown in the empirical analyses presented in the previous section, the Gentoo bug network has a large clustering coefficient and a heavy-tail degree distribution. To explain the evolution of the Gentoo bug network, we start with reviewing some existing network growth models.

A. Fit of existing models

The random graph model proposed by Erdos and Renyi, referred to as the ER model, has a Poisson degree distribution and the clustering coefficient is just 1/76 of the Gentoo bug network [28]. The WS model, proposed by Watts and Strogatz, would possess a high clustering



(a)



(b)

Fig. 4. The degree distributions predicted by two models and that of the Gentoo bug network. (a) The BA model. (b) The KRL model.

coefficient. However, it loses its efficacy in explaining the incremental growth of bugs in the Gentoo bug network because the number of nodes is fixed at the very beginning [9]. As Barabasi and Albert stated in [6], most man-made networks are naturally connected in a scale-free manner. The model they proposed, referred to as the BA model, is based on two mechanisms, incremental growth and preferential attachment. Starting from m_0 nodes with no edge, the network grows by adding one new node with m edges ($m \leq m_0$) at each time step. The probability that an existing node would be connected to the new node is proportional to the existing node's degree, which could be expressed by $\Pi_i = k_i / \sum_j k_j$. Krapivsky, Rendner, and Leyvraz (KRL) extended the BA model into a more flexible version by adding a parameter into the preferential attachment probability: $\Pi_i = k_i^\alpha / \sum_j k_j^\alpha$, where α is a tunable parameter [7].

As stated in Section V.A, the edges' increase resulted from the same number of new nodes can be viewed as about the same. However, due to the sparsity of the Gentoo bug network, the resulting edges' increase for each new node is only around 1.07. For the incremental growth in BA and KRL models, in which each new node arrives with m new edges, the number m should be an integer. To make the BA and KRL models applicable to the Gentoo bug network, we introduce the mechanism proposed in [16]. Each node arrives with m edges, where m is drawn from the set {1, 2}

with corresponding probability $\{p_1, p_2\}$. Suppose the number of new nodes with one edge is N_1 and the number of nodes with two edges is N_2. Then, $N_1 + N_2 = N$ and $N_1 + 2N_2 = M$, where N is the number of nodes and M is the number of edges. Then, $p_1=N_1/N$=0.93258 and $p_2=N_2/N$=0.06742.

Fig. 4 shows the degree distributions predicted by the BA and KRL models. As we can see in Fig. 4 (a), the actual degree distribution decreases faster than the simulation result of the BA model. Moreover, the largest degree predicted by the BA model is just one third that of the actual value. The clustering coefficient averaged over 50 times of simulation is 0.00039, which is far smaller than the Gentoo bug network's 0.02199. Fig. 4 (b) shows the results of KRL simulations when α equals to 1.17 and 1.32 respectively. In the case of 1.17, the maximum degree predicted is close to the actual value. However, the absolute value of the slope of the degree distribution is smaller than the actual value and the clustering coefficient is 0.0023, which is only about one tenth that of the Gentoo bug network. When α equals to 1.32, although the clustering coefficient of 0.0197 is close to that of the Gentoo bug network, the degree distribution deviates a lot from that of the Gentoo bug network when degree k is larger than 5. In addition, the maximum degree of this model exceeds the actual maximum degree substantially.

B. *Proposed model modification*

As we noted in the empirical analyses in Section V, nodes with similar degrees do not tend to connect with each other. In the incremental growth procedure of traditional BA and KRL models, a new node has a fixed degree and the degree difference between the new node and the existing nodes would be included in the degrees of the existing nodes implicitly. However, since the incremental growth has been modified and a new node here may bring with it one or two edges, we believe that including the degree difference would refine the simulation result. We therefore introduce such a factor into the model by modifying the probability of preferential attachment to $\Pi_i = k_i^{\alpha} d_i^{\beta} / \sum_j k_j^{\alpha} d_j^{\beta}$, where d_i is the absolute value of the degree difference between the new node and an existing node i.

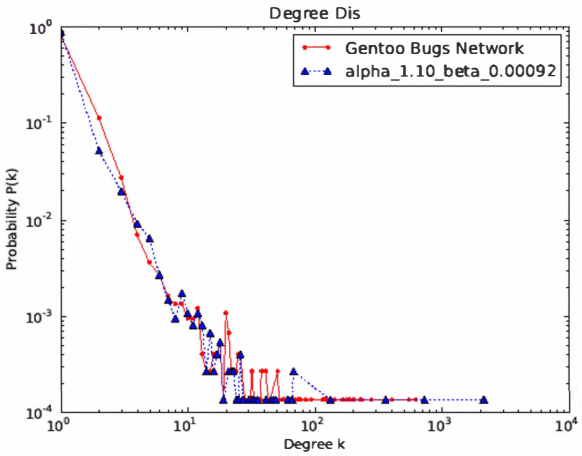


Fig. 5. Simulation result of the new model when α is 1.10 and β is 0.00092.

The simulation result when α is 1.10 and β is 0.00092 is shown in Fig. 5. The degree distribution of this model is similar to that of the Gentoo bug network in general. Moreover, the clustering coefficient is 0.02193, which is quite close to the actual value of 0.02199 with a tiny deviation of 0.27%. We tested various settings of α and β in experiments, and the result shown here seems to have a reasonably good tradeoff between the match of clustering coefficients and the match of degree distributions. Although there are still some problems in the simulation result (for example, the largest degree predicted by our model is still larger than the actual value,) our model outperforms the BA and KRL models in general.

VII. CONCLUSION

In this paper, we analyzed open source software bugs with complex networks, treating bugs as nodes and the relationships among bugs as edges. Our empirical analyses show that the Gentoo bug network has a heavy-tail degree distribution and a large clustering coefficient. Besides analyzing the topological characteristics of the Gentoo bug network, we found that the traditional BA and KRL models are not so satisfactory in explaining the high clustering coefficient and heavy-tail degree distribution of this network. Based on the fact that nodes with similar degrees in the Gentoo bug network do not tend to connect with each other, we introduced the factor of degree difference by modifying the probability of preferential attachment. Our experiments show that the Gentoo bug network can be better explained using the new model.

Besides the bug network we analyzed, there are many other interesting phenomena that exist in the development process of open source software projects. An interesting topic for future research would be to investigate the co-evolution of different networks, for example, the developer network, the package network, and the software bug network. This would be beneficial to software engineering as well as complex system analysis and modeling. Another possible future research direction would be to analyze more open source software projects to help build a better model for explaining the open source software development process.

ACKNOWLEDGMENT

The authors would like to thank Youzhong Wang, Yuan Luo, Zhidong Cao, Huiqian Li, and Fen Xia for valuable discussions and suggestions. This work was supported by the National Natural Science Foundation of China under grants No. 60875049, 60921061, and 90924302, and by the Chinese Academy of Sciences under grants No. 2F07C01 and 2F09N06.

REFERENCES

[1] M. Newman, "The structure and function of complex networks," *SIAM Rev.*, vol. 45, pp. 167-256, 2003.

[2] R. Albert and A. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, pp. 47-97, 2002.

[3] A. Barabasi, R. Albert, and H. Jeong, "Diameter of the World Wide Web," *Nature*, vol. 401, pp. 130-131, 1999.

[4] J. Liu, Y. Dang, Z. Wang, and T. Zhou, "Relationship between the in-degree and out-degree of WWW," *Physica A: Statistical Mechanics and its Applications*, vol. 371, pp. 861-869, 2006.

[5] B. Tadic, "Dynamics of directed graphs: the world-wide Web," *Physica A: Statistical Mechanics and its Applications*, vol. 293, pp. 273-284, 2001.

[6] A. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science,* vol. 286, pp. 509-512, 1999.
[7] P. Krapivsky and S. Redner, "Organization of growing random networks," *Physical Review E,* vol. 63, p. 66123, 2001.
[8] M. Wang, G. Yu, and D. Yu, "Effect of the age of papers on the preferential attachment in citation networks," *Physica A: Statistical Mechanics and its Applications,* vol. 388, pp. 4273-4276, 2009.
[9] D. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," *Nature,* vol. 393, pp. 440-442, 1998.
[10] M. Newman, "The structure of scientific collaboration networks," *Proceedings of the National Academy of Sciences,* vol. 98, pp. 404-409, 2001.
[11] U. Brandes, P. Kenis, J. Lerner, and D. van Raaij, "Network Analysis of Collaboration Structure in Wikipedia," in *Proc. 18th International World Wide Web Conference*, Madrid, Spain, 2009, pp. 731-740.
[12] X. F. Liu, C. K. Tse, and M. Small, "Complex network structure of musical compositions: Algorithmic generation of appealing music," *Physica A: Statistical Mechanics and its Applications,* vol. 389, pp. 126-132, 2010.
[13] Y. Hu and D. Zhu, "Empirical analysis of the worldwide maritime transportation network," *Physica A: Statistical Mechanics and its Applications,* vol. 388, pp. 2061-2071, 2009.
[14] L. Lacasa, M. Cea, and M. Zanin, "Jamming transition in air transportation networks," *Physica A: Statistical Mechanics and its Applications,* vol. 388, pp. 3948-3954, 2009.
[15] C. Myers, "Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs," *Physical Review E,* vol. 68, p. 46116, 2003.
[16] X. Zheng, D. Zeng, H. Li, and F. Wang, "Analyzing open-source software systems as complex networks," *Physica A: Statistical Mechanics and its Applications,* vol. 387, pp. 6190-6200, 2008.
[17] A. Andrea, "On the automation of fixing software bugs," in *Proc. the 30th international conference on Software engineering*, Leipzig, Germany, 2008, pp. 1003-1006.
[18] Z. Andreas, "Where Do Bugs Come From?," *Electron. Notes Theor. Comput. Sci.,* vol. 174, pp. 55-59, 2007.
[19] H. Israel, M. G. Daniel, M. G.-B. Jesus, and R. Gregorio, "Towards a simplification of the bug report form in eclipse," in *Proc. the 2008 international working conference on Mining software repositories*, Leipzig, Germany, 2008.
[20] G. Tibor, F. Rudolf, and S. Istvan, "Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction," *IEEE Trans. Softw. Eng.,* vol. 31, pp. 897-910, 2005.
[21] W. Cathrin, P. Rahul, Z. Thomas, and Z. Andreas, "How Long Will It Take to Fix This Bug?," in *Proc. the Fourth International Workshop on Mining Software Repositories*, 2007.
[22] W. Xiaoyin, Z. Lu, X. Tao, A. John, and S. Jiasu, "An approach to detecting duplicate bug reports using natural language and execution information," in *Proc. the 30th international conference on Software engineering*, Leipzig, Germany, 2008, pp. 461-470.
[23] H. David and P. William, "Finding bugs is easy," in *Proc. the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, Vancouver, BC, CANADA, 2004, pp. 92-106.
[24] G. Tassey, "The economic impacts of inadequate infrastructure for software testing," National Institute of Standards and Technology, Gaithersburg, MD May 2002.
[25] G. Canfora and L. Cerulo, "Impact analysis by mining software and change request repositories," in *Proc. 11th International Symposium on Software Metrics*, 2005, pp. 259-267.
[26] *Gentoo Bugzilla.* Available: http://bugs.gentoo.org/
[27] E. D. Kolaczyk, "Statistical Analysis of Network Data," ed New York: Springer, 2007.
[28] P. Erdos and A. Renyi, "On random graphs," *Publications Mathematics,* vol. 6, pp. 290-297, 1959.