Signatures of small-world and scale-free properties in large computer programs

Alessandro P. S. de Moura,[1] Ying-Cheng Lai,[2,3] and Adilson E. Motter[2]
[1]*Instituto de Física, Universidade de São Paulo, Caixa Postal 66318, 05315-970 São Paulo, Brazil*
[2]*Department of Mathematics and Statistics, Arizona State University, Tempe, Arizona 85287, USA*
[3]*Department of Electrical Engineering, Department of Physics, Arizona State University, Tempe, Arizona 85287, USA*
(Received 18 February 2003; published 15 July 2003)

A large computer program is typically divided into many hundreds or even thousands of smaller units, whose logical connections define a network in a natural way. This network reflects the internal structure of the program, and defines the ''information flow'' within the program. We show that (1) due to its growth in time this network displays a *scale-free* feature in that the probability of the number of links at a node obeys a power-law distribution, and (2) as a result of performance optimization of the program the network has a *small-world* structure. We believe that these features are generic for large computer programs. Our work extends the previous studies on growing networks, which have mostly been for physical networks, to the domain of computer software.

DOI: 10.1103/PhysRevE.68.017102 PACS number(s): 89.75.Hc, 87.23.Ge, 89.20.Hh

Large computer programs nowadays are becoming increasingly more complex. Such a program can easily contain hundreds of thousands or even millions of lines of code. In order to make these programs manageable, the code is split into many small files that are linked together in a coherent but quite sophisticated fashion. A large computer program can thus be regarded as a complex network. But what are the characteristics of such a network?

Some basic features about large computer programs are the following. First, they are *dynamic* in that they continue to evolve in time. For instance, the beginning versions of a program may be relatively simple and small in size. With time the application demand increases, resulting in continuous expansion of the program in many aspects. Thus, the underlying networks may be regarded as *growing networks*. Second, there exists a number of ''key'' components of the program which are linked to many other components (such as subroutines). As new components developed for new applications are added to the program, they are more likely to be linked to the key components of the program. That is, the network develops according to the rule of *preferential attachment*. As argued by Barabási *et al.* in their seminal work [1,2], growth with preferential attachment is one possible dynamical mechanism responsible for the network to exhibit the scale-free characteristic, i.e., a power-law scaling for the probability distribution of the number of links at a node.

For a dynamically growing network, however, at a given time, one can also view it as ''static'' and ask for the topology of the connections between the nodes. Most networks occurring in the nature are large, as they usually contain a huge number of nodes, but they are sparse in the sense that the average number of links per node is typically much less than the total number of nodes. Sparse networks can be characterized as *regular*, *random*, and *small world*. Most regular network possess the property that if two nodes are connected to a common third node, then there is a high probability that the two nodes are connected between themselves. That is, the networks has a high degree of *clustering*. However, in general it takes many steps to move between two arbitrary nodes in the network, i.e., the *shortest possible path* to go from one node to another can be long (in a statistical sense). A high degree of clustering and a large value for the average shortest path are thus the two defining properties of most locally connected regular networks. At the opposite end are random networks [3]: due to the sparsity and random connections, such networks have extremely low degree of clustering and small average shortest path. Regular and random networks had been the main focus of research on network structure and dynamics. It was pointed out in Ref. [4] that there exists a physically realizable range of network topology for which the degree of clustering can be almost as high as that of a regular network, but the average shortest path can be almost as small as that of a random network. These are small-world networks. Structurally, a small-world network differs from a regular one in that there exist a few random links between distant nodes in the former. Watts and Strogatz argued that the small-world configuration is expected to be found commonly in large, sparse networks of the real world. Indeed, examples of small-world networks identified so far occur in almost every branch of science, which include nervous system [4,5], epidemiological invasions [6,7], business management [8], electrical power grid [9], Internet and World Wide Web [10–13], social networks [14,15], metabolism [16], scientific-collaboration network [17,18], Ising model in physics [19], religion and economic growth network [20], polymer networks [21], gene network [22], and linguistics [23].

In this paper, we investigate the network properties of large computer programs and present results for four widely used computer programs, whose codes are publically available and can be downloaded from the Internet. They are (1) the Linux kernel, the core program of the Linux operating system; (2) ''Mozilla,'' the open source version of the web-browser Netscape; (3) ''XFree86,'' the Unix X-Window graphics package; and (4) ''Gimp,'' an image manipulator program for Unix. We study the structure of these programs and develop a natural way to construct the networks underlying these programs. We provide a strong evidence that the networks are scale-free and small worlds. While both the scale-free and small-world features have been demonstrated

 ©2003 The American Physical Society

in many *physical* (or ''hardware'' type of) networks such as the Internet, the World Wide Web, and actor collaboration networks [1,4,24–26], our work demonstrates that these features also govern the network dynamics and topology in the *software* domain of computer science.

The programming language of choice for encoding large complex programs is C (and its offspring C++). In order to make a program manageable, the code is split into many small files. These files are of two kinds: *source* files and *header* files. The source files (usually with names terminating in ''.c'' or ''.cpp'') contain the actual code, whereas the header files (with termination ''.h'') have definitions of variables, constants, data structure, and other information needed by the source files. A large program typically consists of thousands of source and header files. If a source file needs the information contained in a header file, that file is ''included'' in the source file with an ''#include'' clause. For example, if the source file ''main.c'' needs some data structure defined in ''sys.h,'' it contains a statement such as ''#include <sys.h>,'' whereby the contents of ''sys.h'' are made accessible to ''main.c.''

A network can now be defined from the set of source and header files as follows. The nodes of the network are header files, and two nodes are said to be connected if the corresponding header files are both included in the same source file. The connected header files are thus functionally related (they ''work together'' to help the source file in which they are both included do its job). By using a simple program that automatically scans every source file to see which header files each one of them includes, we generate the network corresponding to each of the four large programs aforementioned. We note that a few header files included in the source files belong to external libraries, and are not part of the program itself. While generating the networks, we ignore such files. Also, we only consider the largest connected component of the network, which includes over 90% of all nodes in all four cases.

We first present results concerning the scale-free feature of the computer-code networks. Let k be the variable that measures the number of links at different nodes in the network. For a network that contains a large number of nodes, k can be regarded as a random variable. Let $P(k)$ be the probability distribution of k. A scale-free network is characterized by the following algebraic scaling behavior in $P(k)$:

$$P(k) \sim k^{-\gamma}, \tag{1}$$

where γ is the scaling exponent. As pointed out in Refs. [1,2,25], many real networks, such as the Internet, the World Wide Web, and the network of movie actors, appear to be scale-free with the value of the exponent ranging from 2 to 3. The theoretical model proposed in Ref. [2] suggests the following two basic features in the network dynamics, which determine the algebraic scaling law: growth and preferential attachment. For growth, one can start with a small number m_0 of vertices and at every time step add a new vertex with m edges to the network, where $m \leqslant m_0$. For preferential growth, one can choose the probability that a new link is to be added to the ith node to be proportional to the number of links already existing in that node. The scaling law (1) can be derived from these two conditions [2]. Figure 1 shows the scaling behavior of $P(k)$ for three of the computer programs that we consider here, where panels (a)–(c) correspond to the Linux kernel, XFree86, and Mozilla, respectively. [The total number of nodes in the network associated with Gimp is too small to allow for the statistical quantity $P(k)$ to be computed.] For large k, a robust algebraic scaling behavior is present in all the three cases, where the scaling exponents are $\gamma_{Linux} \approx 2.8$, $\gamma_{XFree86} \approx 2.9$, and $\gamma_{Mozilla} \approx 1.9$. These results suggest that large computer programs can be regarded as scale-free growing networks [27].

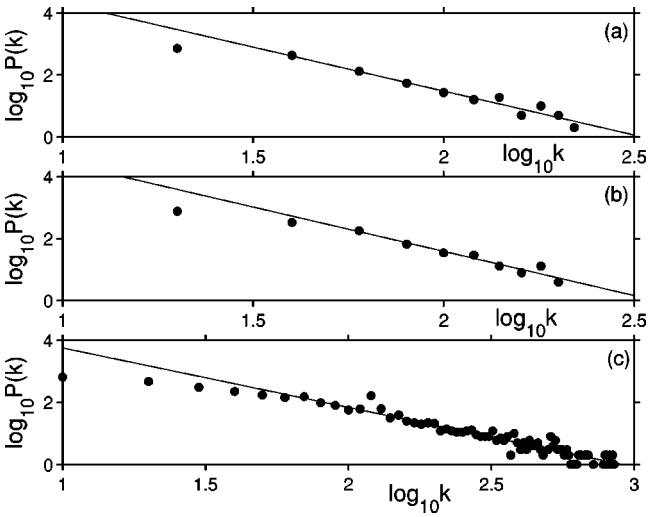

FIG. 1. Algebraic scaling behavior of the non-normalized probability $P(k)$ of the underlying networks for widely used computer programs: (a) the Linux kernel, (b) XFree86, and (c) Mozilla.

We next turn to the small-world feature of the large computer-program networks. For a given program, once the underlying network is built up, we can calculate the quantities that characterize their statistical properties; these are shown in Table I for each program. We see that the average number of links per node μ in all networks is much smaller than the total number of nodes N, which means that the networks are *sparse*, a necessary condition for the notion of small-world network to be meaningful. The quantities of

TABLE I. Results for the networks corresponding to the four programs we have studied. N is the total number of nodes; μ is the average number of links per node; C is the clustering coefficient; C_{rand} is its value for an equivalent random network; L is the average shortest path; and L_{rand} is the same quantity for the corresponding random network.

Program	N	μ	C	C_{rand}	L	L_{rand}
Linux kernel	1448	41.4	0.88	0.03	2.11	1.93
Mozilla	3803	76.6	0.81	0.02	2.49	1.87
XFree86	1465	33.0	0.81	0.02	2.56	2.05
Gimp	403	43.9	0.83	0.11	2.28	1.56

interest to us are the *average shortest path L*, which is the average over all pairs of nodes of the number of links in the shortest path connecting the two nodes; and the *clustering C*, which is the probability that two nodes *a* and *b* are connected, given that they are both connected to a common third node *c*. If *C* is close to 0, the network is not locally structured; if *C* is close to 1, the network is highly clustered.

A random network with given N and μ (with $N \gg \mu$) is characterized by having small values of L and C. In particular, for $N \to \infty$ and μ fixed, the average shortest path in the largest connected component approaches the logarithmic behavior of a Moore graph [3],

$$L_{rand} \approx \frac{\ln N}{\ln \mu}, \tag{2}$$

and the clustering coefficient approaches zero [4],

$$C_{rand} \approx \mu / N. \tag{3}$$

On the other hand, regular networks are typically highly clustered, but at the price of having very large L. The small-world networks lie in between these two extremes. They have large clustering, $C \gg C_{rand}$, and small average shortest path, $L \approx L_{rand}$, where C_{rand} and L_{rand} are the respective statistical quantities for a random network with the same parameters N and μ. From Table I, we see that the networks corresponding to all four programs we have studied are small-world networks. This result seems to be typically true for any large enough program. Therefore, we conclude that the logical structure of large programs can be described by small-world networks.

Notice that each source file corresponds to a totally connected subgraph in the network, since every header file included in a source file is connected to every other header file included in that same source file. Thus the network consists of several clusters (corresponding to the source files) interconnected by header files that are included in more than one source file. The clustering effect of the source files is the same as movies in the actors' network (the ''Kevin Bacon network''). Because of this, it is perhaps not surprising that C is large for our program networks. The fact that L is small, however, is not obvious and is due to the nodes between otherwise distant clusters, caused in turn by header files included in more than one source file.

We have also investigated the influence of very highly connected nodes on the network, and how the networks' statistical properties change if those highly connected nodes are removed. In order to do this, we define a new network from each of the four original programs by removing all the nodes with a number of links larger than $N/4$. The new networks will, of course, have smaller N and μ, and a larger L. We now calculate C and L for these new networks. The results are displayed in Table II. We see that these networks still have the small-world property, in all cases. In fact, we have verified that the further removal of highly connected nodes always preserves the small-world property of the resulting networks, up to the point where we remove too many nodes, and the resulting networks are too small to define meaningful statistics. This shows that the small-world property in these networks is a robust phenomenon, and does not depend on the presence of a few highly connected nodes in the tail of the algebraic distribution (1).

TABLE II. Results for the networks constructed from the ones used in Table I by deleting all the nodes with a number of links larger than $N/4$.

Program	N	μ	C	C_{rand}	L	L_{rand}
Linux kernel	1397	20.8	0.85	0.01	2.85	2.34
Mozilla	3760	68.0	0.80	0.02	2.72	1.93
XFree86	1435	30.8	0.80	0.02	2.79	2.09
Gimp	241	24.9	0.74	0.10	2.55	1.66

Finally, we observe that a network that contains full information about both header and source files can be defined. The result is a bipartite network [29] which has two types of nodes (one corresponding to the header files and the other to the source files) and links that run only between nodes of different kinds, as defined by the ''#include'' clause. The networks analyzed so far correspond to the projection of this bipartite network onto the space of header files. A similar projection with respect to the space of source files produces a network, whose nodes are source files and links are between source files that include a common header file. The network of header files and the network of source files share similar properties. In particular, both evolve according to a preferential growth and both exhibit the small-world feature.

In summary, we have shown that large computer programs correspond to growing networks that generally possess the small-world and scale-free properties. As computer softwares for various modern applications are becoming increasingly more complex, it is important to study and understand their topological structure for improved efficiency and improved performance. In particular, even for large computer programs the flow of information within the program is expected to be quite efficient because, as we have shown, in spite of the size of the program the average shortest path in the underlying network is very small. Also, some of the nodes of these networks appear to be much more connected than the average, which means that the corresponding files in the program are required for a large number of applications, making them relatively more important. This in turn, together with the very fact that different parts of the program (different applications) make use of a limited number of files, is expected to help the maintenance and debugging of the programs. In debugging, for example, the first files to be checked should be the most connected ones. We emphasize that our viewpoint that sophisticated computer softwares can be considered as networks is relevant because the network features identified in this paper are expected to be generic and universal.

This work was supported by the AFOSR CIP (Critical Information Protection) Program under Grant No. F49620-01-1-0317.

[1] A.-L. Barabási and R. Albert, Science **286**, 509 (1999).
[2] A.-L. Barabási, R. Albert, and H. Jeong, Physica A **272**, 173 (1999).
[3] B. Bollobás, *Random Graphs* (Academic Press, London, 1985).
[4] D.J. Watts and S.H. Strogatz, Nature (London) **393**, 440 (1998).
[5] K.E. Stephan, C.C. Hilgetag, G.A.P.C. Burns, M.A. O'Neill, M.P. Young, and R. Kotter, Philos. Trans. R. Soc. London, Ser. B **355**, 111 (2000).
[6] M.J. Keeling, Proc. R. Soc. London, Ser. B **266**, 859 (1999).
[7] A.L. Lloyd and R.M. May, Science **292**, 1316 (2001).
[8] B. Kogut, Strategic Manage. J. **21**, 405 (2000).
[9] D. J. Watts, *Small Worlds* (Princeton University Press, Princeton, 1999).
[10] R. Albert, H. Jeong, and A.-L. Barabási, Nature (London) **401**, 130 (1999).
[11] R. Albert, H. Jeong, and A.-L. Barabási, Nature (London) **406**, 378 (2000).
[12] A.-L. Barabási, R. Albert, and H. Jeong, Physica A **281**, 69 (2000).
[13] A.-L. Barabási, Phys. Today **14**(7), 33 (2001).
[14] B. Skyrms and R. Pemantle, Proc. Natl. Acad. Sci. U.S.A. **97**, 9340 (2000).
[15] B. Kogut and G. Walker, Am. Sociol. Rev. **66**, 317 (2001).
[16] D.A. Fell and A. Wagner, Nat. Biotechnol. **18**, 1121 (2000).
[17] M.E.J. Newman, Phys. Rev. E **64**, 016131 (2001).
[18] M.E.J. Newman, Phys. Rev. E **64**, 016132 (2001).
[19] M. Gitterman, J. Phys. A **33**, 8373 (2000).
[20] U. Blum and L. Dudley, J. Evol. Econ. **11**, 207 (2001).
[21] A.A. Gurtovenko and A. Blumen, J. Chem. Phys. **115**, 4924 (2001).
[22] N. Guelzim, S. Bottani, P. Bourgine, and F. Kepes, Nat. Genet. **31**, 60 (2002)
[23] A.E. Motter, A.P.S. de Moura, Y.-C. Lai, and P. Dasgupta, Phys. Rev. E **65**, 065102 (2002).
[24] S.H. Strogatz, Nature (London) **410**, 268 (2001).
[25] R. Albert and A.-L. Barabási, Rev. Mod. Phys. **74**, 47 (2002).
[26] N.E.J. Newman, SIAM Rev. **45**, 167 (2003).
[27] An idealized scale-free network can be generated by a *linear* preferential attachment rule: $\Pi_i \sim k_i$, where k_i is the number of existing links at node i and Π_i is the probability for this node to acquire a new link. In this case, the connectivity distribution $P(k)$ can be shown [1,2] to be strictly algebraic with the scaling exponent $\gamma=3$. Most realistic networks, such as those studied in this paper, exhibit the scale-free feature only to certain extent. Moreover, in the range where $P(k)$ appears to be algebraic, the scaling exponent usually deviates from the ideal value of 3. There have been various models to address these realistic issues [28] (see also Ref. [29]). A feature of most growing models of scale-free networks is that $P(k)$ does not scale algebraically for k smaller than the average number of links per node, in agreement with the deviations for small k observed in Fig. 1.
[28] See, for example, L.A.N. Amaral, A. Scala, M. Barthélémy, and H.E. Stanley, Proc. Natl. Acad. Sci. U.S.A. **97**, 11 149 (2000); P.L. Krapivsky, S. Redner, and F. Leyvraz, Phys. Rev. Lett. **85**, 4629 (2000); S.N. Dorogovtsev, J.F.F. Mendes, and A.N. Samukhin, *ibid.* **85**, 4633 (2000); R. Albert and A.-L. Barabási, *ibid.* **85**, 5234 (2000); S.N. Dorogovtsev and J.F.F. Mendes, Phys. Rev. E **63**, 025101 (2001); **63**, 056125 (2001); G. Bianconi and A.-L. Barabási, Europhys. Lett. **54**, 436 (2001); Phys. Rev. Lett. **86**, 5632 (2001); P.L. Krapivsky and S. Redner, Phys. Rev. E **63**, 066123 (1999); P.L. Krapivsky, S. Redner, and F. Leyvraz, Phys. Rev. Lett. **85**, 4629 (2000); P.L. Krapivsky, G.J. Rodgers, and S. Redner, *ibid.* **86**, 5401 (2001); B. Tadić, Physica A **293**, 273 (2001).
[29] M.E.J. Newman, S.H. Strogatz, and D.J. Watts, Phys. Rev. E **64**, 026118 (2001).