STUDYING COMPLEX NETWORKS WITH CXNET

Árpád Horváth

Óbuda University, Alba Regia University Centre
H-8000 Székesfehérvár, Budai út 45., Hungary
Email: horvath.arpad@arek.uni-obuda.hu

Abstract

Recent investigations in the field of complex networks include the analysis of distributions of connections in particular networks and the clustering properties of networks. In this presentation we analyse the dependency network of the software packages of the Ubuntu distribution of GNU/Linux. We developed the CXNet module for the Python programming language to analyse this network. Using this module the students can easily investigate these properties of the package dependency network.

I. Introduction

Networks are often used as a synonym of the graphs in the field of sociology and some other fields of the science. Networks contains some kinds of entities. A pair of entity can be connected or not. The entities are called nodes, vertices or points, the connections are called as edges.

Complex networks are very large networks with a structure difficult to describe in a compact form. The aim of the science of the complex networks is to study the properties of real networks. The earliest investigations of networks happened in the field of sociology by studying the acquaintance network of people.

There is a lot of networks in the fields of engineering, such as the World Wide Web, the Internet, whose investigation has brought networks in the

Table 1: Real networks

nodes	edge exists if...	type
people	they have met each other	undirected
web pages	there is link from one to other	directed
routers of Internet	there is wire between them	undirected
scientific papers	citation	directed
proteins	they participate in the same interaction	undirected
words	they are synonyms according to a given dictionary	undirected
mathematicians	authors of the same paper	undirected
actors	actors of the same movie	undirected

forefront of investigations recently. In biology and medicine the network of protein interactions, the food chain or to forecast the spreading of a disease the acquaintance and sexual networks are important. Some examples can be found in the Table 1.

Properties of many networks and methods of the investigations have been summarized in several papers [1, 2].

II. Definitions

Networks can be directed or undirected. In directed networks the nodes at the two endpoints of an edge do not have the same role in the connection.

The function and the growth of a network is influenced by the degree distribution of the network. The k_i degree of the i-th node is the number of nodes connecting this node. The in-degree and out-degree are defined in a directed network similarly. In-degree includes only the in-connections, where the arrow leads to the given node, and out degree includes only the out-connections.

The p *degree distribution* is a function. $p(k)$ is the probability of a randomly chosen node has the degree k, or

$$p(k) = Prob(\text{degree of a randomly choosen node} = k). \qquad (1)$$

In directed networks we can investigate the distributions of in-degree and out-degree as well. Many real networks belong to the *scale-free networks*, whose degree distribution decreasing as a power-law function for large degrees. In this networks there are nodes with degrees orders of magnitude larger as the average degree. It is difficult to analyse the plots of degree distributions of scale-free networks, because of the large statistical fluctuation at large degrees. In such cases the plot of the *cumulative degree distribution* is useful. The $P(k)$ value of the cumulative degree distribution gives the probability of a randomly chosen node having degree *larger* then k:

$$P(k) = Prob(\text{degree of a randomly choosen node} \geq k). \tag{2}$$

In real networks the clustering is usually large. The clustering coefficient was introduced to measure this property. For a given node it shows the ratio of the number of existing edges among its neighbours and the number of the all possible edges among the neighbours. It can only be defined for undirected networks, so the directed networks need to transformed to undirected ones to investigate their clustering coefficients. There can be at most $k_i(k_i-1)/2$ edges between the k_i neighbours of the ith node, in the case they form a complete graph as subgraph, so the C_i *clustering coefficient of node i* is

$$C_i = \frac{2E_i}{k_i(k_i-1)}, \tag{3}$$

where E_i is the actual number of the edges between the neighbours of the edge. The C *clustering coefficient of the network* can be defined as the arithmetic mean of the clustering coefficients of the nodes:

$$C = \frac{1}{N}\sum_{i=1}^{N} C_i \tag{4}$$

III. The properties of the package dependency network

We have developed a program to analyze the package dependency network of the GNU/Linux operational system.

The GNU/Linux system has a lot of distributions with their own software packages. These software packages have two prevalent formats. One

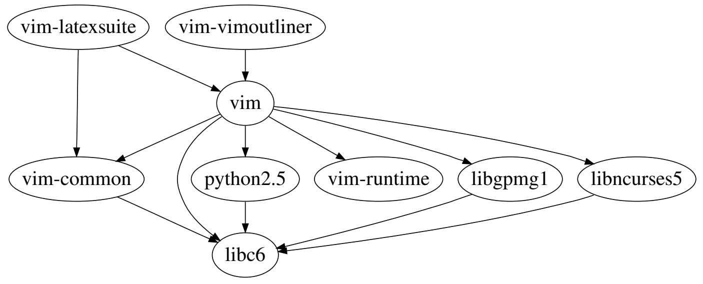

Figure 1: A part of the package dependency network: the surroundings of the package of the Vim text editor. This plot includes the package with the highest degree, the libc6 package, the standard library of the C programming language.

of them are the *deb* packet format developed by the Debian distribution team [5] and used many other distributions, including the popular Ubuntu. The packages can be reached on optical disks (CD, DVD) or via repositories on the World Wide Web. One package can depend on other packages, which means that without them it is not able to function properly. These dependences are stored in other files than software packages, so we need to download only these files to create the network of the packages. These dependences can be treated by the APT package managing tool. This tool can search the dependences of a package, and install this package with all the other packages it needs or with a removal of a package all the dependent packages to remove. The packages compose a directed network. We defined the direction of the edges to direct form the dependent package to the package it depends on. A part of the package dependency network can be seen in Fig. 1, the package of the Vim text editor with its neighbours. We can observe, that we need the version of 2.5 of the Python programming language (python2.5 package) and the standard library of the C programming language to install the vim package and the vim package needs to install other packages as well.

In this article we investigated the package dependency network of the

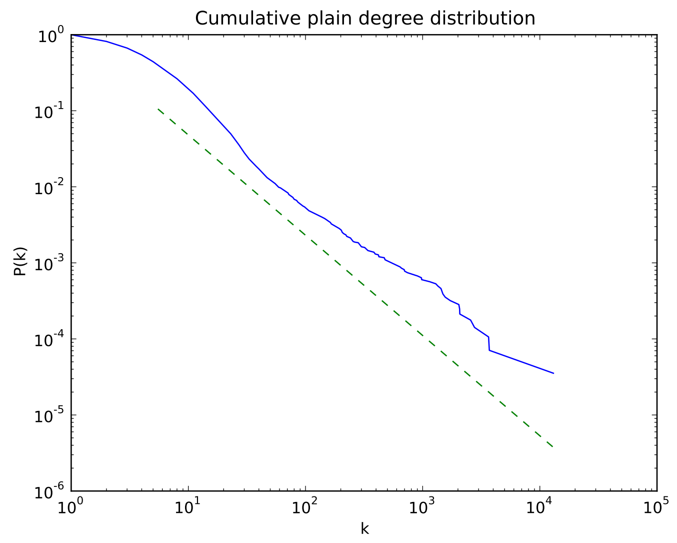

Figure 2: The cumulative degree distribution of the package dependency network (continuous), and the power-law function with the exponent $-1,19$ given by the maximum likelihood method (dashed). $\left(\gamma = 2.19 \pm 0.14\,^{+0.1}_{-0.12}\right)$

Jaunty (9.04) version of Ubuntu distribution dated November 3, 2009. The stored files can be found on the web page
`http://mail.roik.bmf.hu/cxnet/netdata/`

In this network the number of nodes and edges are $N = 27554$ and $M = 126540$ respectively. The network is not connected, but 93.14% of the nodes belongs to one, the largest component. The diameter of the largest component is 13, so any two nodes can be reached from one to the other through 13 edges.

The cumulative degree distribution of the network can be seen in Fig. 2. We can see without any calculation, that the scale-free model better fits the distribution than the random graph model.

We can estimate the absolute value of the exponent, $\gamma > 0$ of the power-law distribution assumed to be valid for the network with the maximum likelihood method [6]:

$$\gamma = 1 + N \left[\sum_{k_i > k_{\min}} \ln \frac{k_i}{k_{\min}} \right]^{-1} , \tag{5}$$

where $k_{\min}$ means the minimum value of degrees above which we assume the power-law behaviour. This estimate is valid for continuous power-law distributions, but with a small modification it can also be used for discrete power-law distributions [7]:

$$\gamma = 1 + N \left[\sum_{k_i > k_{\min}} \ln \frac{k_i}{k_{\min} - 0.5} \right]^{-1} \qquad k_{\min} \in \mathbb{Z} , \tag{6}$$

which we use below. The estimate for its standard deviation is

$$\sigma = \frac{\gamma - 1}{\sqrt{N}} . \tag{7}$$

Eqn: (6) gives the exponent and eqn. (7) its standard deviation exactly, if the values really come from a power-law distribution, which we have not proven in our case. A rigorous analysis of the distribution needs deeper statistical knowledge [7], we think that the proof of the power law nature is worth to be done only with students with such an interest.

Instead of proving the power-law behaviour, we propose to plot the dependence of γ on the minimal degree ($k_{\min}$) as in the Fig. 3. As the plot shows the statistical uncertainty is very small for small values of $k_{\min}$, but this is not the real uncertainty of the exponent. As we can see, the value of the exponent becomes stable around $k_{\min}$= 240, indicating a true power-law behaviour only for large degrees. Thus the exponent can be interpreted only for $k_{\min}$ values above 240. At the uncertainty of the exponent beside the γ statistical error we may include the fluctuation of the γ values: the systematic uncertainty. With $k_{\min}$= 240 we obtain $\gamma = 2.19$ and $\sigma = 0.14$ from the equations (6) and (7). The systematic uncertainty is calculated from the differences of the values from the value at 240.

$$240 \leq k_{\min} \leq 600 \qquad \Rightarrow \qquad 2.07 \leq \gamma \leq 2.29$$

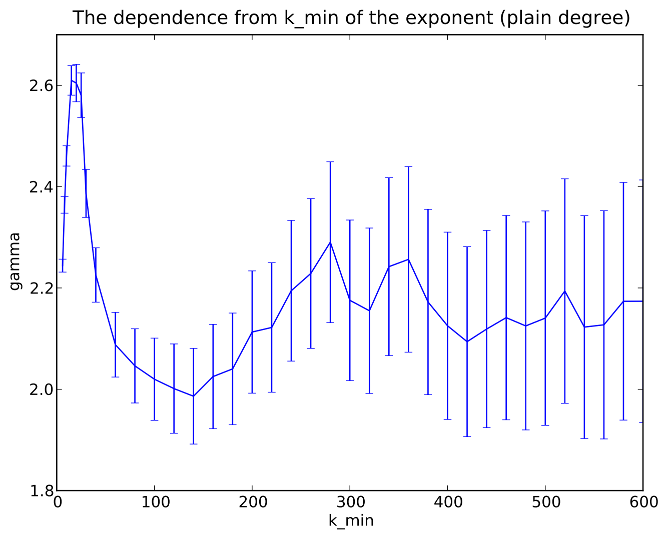

Figure 3: The calculated values of the exponent as a function of $k_{\min}$.

The largest difference in the negative direction is 0.12, in the positive direction is 0.10, so

$$\gamma = 2.19 \pm 0.14 \begin{matrix} +0.1 \\ -0.12 \end{matrix}$$

Using cumulative distribution, we obtain the exponent $-\gamma + 1 = -1.19$. Plotting the power law function with that exponent and the cumulative distribution we can check, that the cumulative distribution is really nearly parallel to the power law function $k^{-1.19}$.

The average of the in-degree is equal to the ratio of the number of edges and the number of nodes

$$\langle k_{\text{in}} \rangle = M/N = 4.592.$$

However the largest in-degree is 11868, more than 2500 times larger.

The clustering coefficient is defined for undirected networks, so we have to transform our directed network into undirected. The clustering coefficient

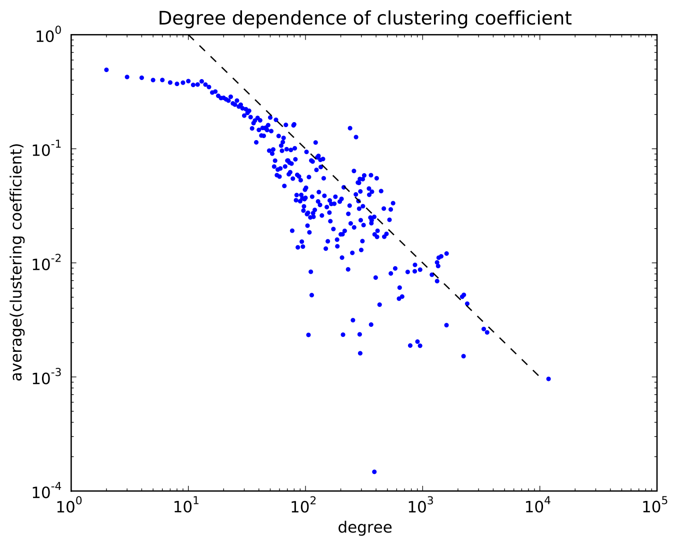

Figure 4: The dependence of the average clustering coefficient on the degree (dots) and the power law function with the exponent -1 (dashed line)

of the network is 0.308. We can compare it with the values coming from network models. If the network were a random graph, the p probability of edges, as well as the C clustering coefficient would be

$$p = \frac{M}{N(N-1)/2} = C = 0.000333. \tag{8}$$

In the Barabási–Albert model the mean degree is $2m$. To compare the clustering coefficient of the dependency network to that of the Barabási–Albert model, we first calculated the mean degree of the package dependency network, and we have set m the rounded value of the half of the mean degree. As the mean degree is

$$\langle k \rangle = 2M/N = 9.1849, \text{ we have choosen } m = 5$$

for the parameter of the corresponding Barabási–Albert model. With the parameters $N = 27554$, $m = 5$ we obtained a network with 137745 edges and clustering coefficient 0.0032. The parameters of the two models give us a clustering coefficient 2–3 orders of magnitude smaller, than that of the package dependency network. The clustering coefficient of the hierarchical model, discussed in the previous section, is $C_h = 0.743$ [4], which is in the same order of magnitude as the coefficient of the dependency network.

We plot the clustering coefficient of the nodes with given degree as a function of degree (Fig. 4). The dashed line represents the power law function with the exponent -1. The function does not contradict the degree dependency of the other networks [4]. Our network can be called hierarchical network.

IV. Complex networks in the education

One aim of our program development was that students could analyze a real network created by the computer. In our center in Székesfehérvár there are two computer rooms, where students can use Linux distributions with *deb* packages (Debian and Ubuntu). Our CXNet module, uses the `apt` module of Python to get the dependences, and the `IGraph` or `NetworkX` module [8] to create and analyze the package dependency network and to compare with the models, and the `pylab` module of the `matplotlib` package to plot functions. There is some Hungarian video tutorial and document helping to learn the usage of the Python language and the NetworkX module at `http://mail.roik.bmf.hu/cxnet`. The English and Hungarian documentation of the CXNet module can be reached from here too and a syllabus for a 10-week-course in Hunarian.

Unfortunately the usage of IGraph and NetworkX is quite different. NetworkX has been written entirely in Python. However the NetworkX module is easier to use and install, we prefer IGraph to NetworkX for its speed. The IGraph module [9] uses libraries written in C and have been developed to handle very large networks.

The Pylab module allows us to plot functions as the degree distribution. The functions of Pylab is similar to the functions of the MATLAB language. We recommend to use the IPython interactive shell with the option `-pylab`. So the plotting of functions and use of mathematical functions are easier as with the standard Python shell.

CXNet have several components worth to mention:

1. The `debnetwork` function can create the *deb* package dependency network as an `igraph.Graph` or `networkx.DiGraph` instance.
2. The `DegreeDistribution` is a class of CXNet. It can create cumulative or binned plots of the distribution from the graph objects created by `debnetwork` with the Pylab module, and calculate the γ exponent.
3. There is a function to download network data from the web. These networks include archived dependency networks as well for performing the analysis on systems not using deb packages.
4. There is a class for creating multifractal networks. Further development is necessary to create a tool for adjusting the parameters of the multifractal to creating networks with pre-defined properties [10].
5. The `network_evolution` is a standalone module. With this one can carry out simulations of disease spreading on a growing network, make measurements on that network during the evolution, and store the results in a standard binary format of Python. Measuring the diameter took a lot of time with NetworkX. With IGraph this time decreased significantly.

We introduced the teaching of complex networks as a part of an optional course. In this course the students learn the methods to analyze networks, the models of the networks and the properties of these networks, and they compare it to real networks. We developed a syllabus for this course. It is for a course in a computer room. However it includes only 10 weeks because some extra weeks are necessary to get acquinted with the Python language and some aspects of the object oriented programming.

Our experience is that the students can perform exercises at home such as to find the p probability above which a giant component appears in Erdős–Rényi graphs, and how the value of p depends on the number of nodes N, or to write a program of the modified Barabási–Albert models, and the analysis of the network obtained.

V. Conclusion

We studied the package dependency network, and we found that it is a scale-free hierarchical network, sharing some properties with other real networks. We found that the Python language with some modules (IGraph or NetworkX) suits for this analysis, these are handy for teaching this field.

We think, that teaching of the complex networks is possible and useful in some areas of the higher education, such as informatics, physics, sociology, engineering and biology, as well as in courses of informatics of secondary education.

This research was supported by the TÁMOP 4.2.1./B-09/1/KONV-2010-0007 project.

References

[1] R. Albert and A. Barabasi, *Statistical mechanics of complex networks*", Rev. Mod. Phys. **74**, 1 (2002)

[2] M. E. J. Newman, *The structure and function of complex networks*, SIAM Review **45**, 2 (2003).

[3] A.-L. Barabási, *Linked: The New Science of Networks.* (Perseus, Cambridge, MA, 2002).

[4] E. Ravasz and A. Barabasi, *Hierarchical organization in complex networks*, Phys. Rev. **E67**, 026112 (2003).

[5] `http://www.debian.org/`

[6] M. E. J. Newman, *Power laws, pareto distributions and zipf's law*, Contemporary Physics **46**, 223 (2005).

[7] A. Clauset at al, *Power-law distributions in empirical data,* `http://arxiv.org/abs/0706.1062`

[8] A. Hagberg at al., *NetworkX: High productivity software for complex networks,* `http://networkx.lanl.gov/`

[9] G. Csárdi and T. Nepusz, *Igraph*, `http://igraph.sourceforge.net/`

[10] G. Palla et al.: *Multifractal network generator*, Proceedings of the National Academy of Sciences **107**, 17 (2010).