

Multifractal analysis of the software evolution in software networks

Meili Liu(刘美丽), Xiaogang Qi(齐小刚), and Hao Pan(潘浩)

Citation: Chin. Phys. B, 2022, 31 (**3**): 030501. DOI: 10.1088/1674-1056/ac1b8a
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Dynamics of high-frequency modulated waves in a nonlinear dissipative continuous bi-inductance network

S M Ngounou and F B Pelap
Chin. Phys. B, 2021, 30 (**6**): 060504. DOI: 10.1088/1674-1056/abd751

Empirical topological investigation of practical supply chains based on complex networks

Hao Liao(廖好), Jing Shen(沈婧), Xing-Tong Wu(吴兴桐), Bo-Kui Chen(陈博奎), Mingyang Zhou(周明洋)
Chin. Phys. B, 2017, 26 (**11**): 110505. DOI: 10.1088/1674-1056/26/11/110505

An image encryption scheme based on three-dimensional Brownian motion and chaotic system

Xiu-Li Chai(柴秀丽), Zhi-Hua Gan(甘志华), Ke Yuan(袁科), Yang Lu(路杨), Yi-Ran Chen(陈怡然)
Chin. Phys. B, 2017, 26 (**2**): 020504. DOI: 10.1088/1674-1056/26/2/020504

H_∞ synchronization of the coronary artery system with input time-varying delay

Xiao-Meng Li(李晓蒙), Zhan-Shan Zhao(赵占山), Jing Zhang(张静), Lian-Kun Sun(孙连坤)
Chin. Phys. B, 2016, 25 (**6**): 060504. DOI: 10.1088/1674-1056/25/6/060504

A novel color image encryption algorithm based on genetic recombination and the four-dimensional memristive hyperchaotic system

Xiu-Li Chai(柴秀丽), Zhi-Hua Gan(甘志华), Yang Lu(路杨), Miao-Hui Zhang(张苗辉), Yi-Ran Chen(陈怡然)
Chin. Phys. B, 2016, 25 (**10**): 100503. DOI: 10.1088/1674-1056/25/10/100503

Multifractal analysis of the software evolution in software networks

Meili Liu(刘美丽), Xiaogang Qi(齐小刚)†, and Hao Pan(潘浩)

School of Mathematics and Statistics, Xidian University, Xi'an 0710071, *China*

(Received 10 June 2021; revised manuscript received 25 July 2021; accepted manuscript online 7 August 2021)

As the scale and complexity have been increased in software systems, developers place more emphases on software engineering and system designs. Software architecture is evolved with update of softwares, and it plays a fundamental role in determining quality of software systems. Multifractal characteristics of software networks can reflect software quality. In this paper, we construct a software network from the dependencies between object classes, and gain a deep understanding of software through network analysis. To be specific, multifractal analysis of the software network is performed based on a modified box-covering algorithm that yields fewer boxes. We verify that software with different functions and dependencies is multifractal. Further, different versions of the software are compared to discover the evolution of the software architecture. The results show that the singularity of class dependencies decreases as the software is updated. This trend leads to a more specific division of functions between software modules. One of the visible advantages of our work is that it allows the characterization of software structures at the code level. The methodology and results of this paper provide new insights into the evaluation and design of large-scale software systems.

Keywords: software, multifractal, box-covering algorithm

PACS: 05.45.Df, 05.10.–a **DOI:** 10.1088/1674-1056/ac1b8a

1. Introduction

Software systems are gradually increasing in terms of size and complexity due to the rapid development of the Internet, and developers are paying more attention to software engineering and system designs. Software is created to solve practical problems, but problems and requirements are always changing.[1] To this end, software needs to constantly adapt and develop to accommodate changes in the real world. Architecture of software has a profound impact on quality of software, as Torres pointed out in Ref. [2]: the architecture of software system is the foundation for high availability and fault tolerance of software. Software architecture is widely recognized as the backbone for any successful software system.

The widespread use of software systems has attracted researchers to develop a large body of knowledge about software. Software consists of a large number of object classes and the dependencies between them.[3] When software is referred to as a software network (nodes represent classes and edges represent dependencies),[4] techniques in complex networks are available. Chong *et al*. proposed a method to represent software systems via weighted complex networks and to define the maintainability and reliability of the system regarding the characteristics of the network structure.[5] Yan and Qi showed that large-scale software with complete functions and excellent performance is a scale-free network.[6] The small average shortest path and the connection preference between classes were reflected in this feature.[7] Gao *et al*. introduced a measure of software structure and pointed out that there is some degree of exponential decay in both the degree distribution and node weight distribution in software networks.[8] Wang *et al*. analyzed the relation between the statistical characteristics and influence of nodes in software networks, and proposed the concept of key nodes in software networks.[9] Although software is increasingly adopted by human society, its architecture is not yet completely understood.

Object-oriented technique is advantageous in upgrading software system architecture for its encapsulation, inheritance and polymorphism. Software development based on this technique differs from structure-oriented technique in matter of system architecture, module life cycle and project maintenance. Software requirements are fulfilled by modules with different functions, which makes differences in the dependency between object classes. The complexity of the dependency affects the quality of the software system. Practice in object-oriented software development establishes that the expansion of software scale leads to an increase in the number of object classes. Recently, it has become a hit to build software networks through basic object class dependencies. On this basis, potential features and measurements of software architecture can be explored.[10]

The fractal nature of complex networks is an important structural feature for predicting and interpreting the composition and evolution of networks[11] and was discovered by Song *et al*.[12] In particular, the dimension in a software network characterizes the degree of cohesion for object classes. Concas *et al*. found a linear relationship between the dimension and the parameters CBO and RFC in software networks (coupling between object, CBO; response for a class, RFC).[13,14]

†Corresponding author. E-mail: xgqi@xidian.edu.cn

© 2022 Chinese Physical Society and IOP Publishing Ltd

Turnu *et al*. investigated the relationship between the dimension and software defects, and adopted it as a global quality metric for software.[15]

The box-covering algorithm was proposed by Mandelbrot and was originally employed to find the fractal dimension of graphs in Euclidean space.[16] It is the most broadly adopted for its simplicity and ease of operation. The method was extended to complex networks by Song *et al*.,[12] in which the network is covered with boxes of a given "size". At present, the box fractal dimension algorithms include greedy coloring, compact box burning (CBB) and maximum excluded mass burning (MEMB).[17] The key of the algorithm is to get the minimum number of boxes covering the network, which is NP-hard. Scholars have conducted intensive research to advance the algorithm. For instance, a novel edge-covering method was put forward by Zhou *et al*., which outperformed node-covering in strictly self-similar networks.[18] On the other hand, the algorithm was modified considering the differences in the node degree distribution, where the number of nodes in the box was calculated by the node degree.[19]

Multifractal is a generalization of fractals in which many fractals are spatially intertwined. Multifractal theory is valuable in many science fields: analysis of natural phenomena such as earthquakes,[20] pattern recognition,[21] air traffic flow,[22,23] vehicle traffic flow,[24] and traffic modeling for computer network.[25] Mishra *et al*. investigated the localization properties of the adjacency matrix eigenvectors for small-world networks by this feature.[26] Jiang *et al*. reviewed the methods, models and properties of multifractal analysis in finance, which has implications for other fields as well.[27] In Ref. [28], Zhang *et al*. performed multifractal analysis in the financial market with small-world topology, which may provide new insights into the problem of the origin of multifractality in financial time series. How multifractal works in varied domains is crucial to understand its practical meaning.

A software system with complex object class dependencies always makes it difficult for engineers to maintain. The dependency distribution in a software network can be described by multifractal. The more irregular the network is, the more obvious the nature of the multifractal becomes. That is, the more complex and chaotic a system is, the stronger its multifractal character behaves. Thus, the multifractal characteristic of software network is strongly linked to software quality. This signifies that a well-designed software system should exhibit weak multifractal characteristic. Investigating whether the software is multifractal can be utilized to improve the performance of existing networks or to design new ones. There are various methods to achieve multifractal analysis in scientific fields, and the box-covering algorithm is one of the classical methods.

In this paper, an improved box-covering algorithm applicable to software network is presented to study the complexity and inhomogeneity of software. Experiments are conducted on different softwares to explore their multifractal nature. In addition, simulation results on different versions of one of the software reveal that the differences in dependencies between object classes decrease as the version evolves.

The contribution of this paper is threefold: (1) We focus on the multifractal property of software network, which is unaddressed by current literature. To the best of our knowledge, previous studies have concentrated on the fractal dimension of software networks. (2) The investigation is oriented to collecting data from software code rather than academic data libraries. (3) The improved box-covering algorithm based on complex networks is applied to software networks, which examines the dependencies between classes in a holistic way. Moreover, the algorithm behaves well in software networks.

The remainder of the paper is organized as follows: Section 2 presents the construction of the software network. In Section 3, we introduce the multifractal analysis method in our work and give a case to illustrate its advantages. Experiments performed in Section 4 verify the multifractal nature of software with different functions. Especially, different versions are analyzed to probe the evolution of the software architecture. Finally, the paper is concluded and the future direction is given in Section 5.

2. Software network

2.1. Software network model

Analyzing the software source code is the first step to obtain structural information. The software at the class level is represented by a weighted directed software network, which takes into account both the coupling strength and direction between classes. For instance, in Fig. 1(a), class Letter extends class Paper, and classes People and Pen are members of class Letter. This relation can be portrayed by the class diagram in Fig. 1(b).

2.2. Metrics for class dependency

2.2.1. Five dependencies in the unified modeling language

The dependency strengths between object classes are different and need to be weighted. Dependencies between object classes are reflected in software engineering where classes with similar function are placed in the same package or module. In the unified modeling language (UML), the main dependencies include dependence, association, aggregation, composition and generalization.

Dependency means that the accomplishment of the function of one object class requires reference to another object class and that changes in one partially affect the other. Association is a connection, which is stronger than dependency.

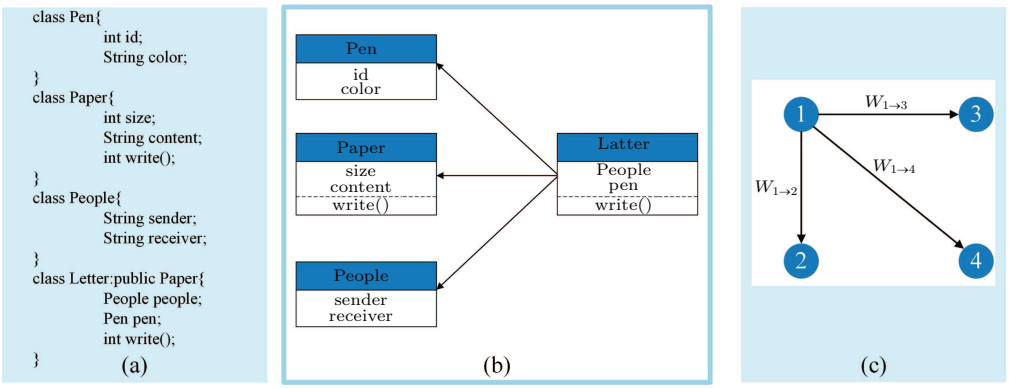


Fig. 1. The relations among code, class diagram and software network: (a) code, (b) class diagram, (c) software network.

Two dependent object classes do not add attributes to the object. However, if they are associated, the attributes of one object class are added by the other. Association makes the two object classes more coupled. Aggregate is an association that tends to have one object class containing the other. Composition is a multi-aggregate relationship where an object class contains multiple object classes and depends on them. Generalization indicates the categorical relationship between general and special elements. In practical development, generalization is the inheritance, interface and implementation of object classes. The strength of these five relations is summarized in Table 1, with D1–D5 indicating a progressive strength.

Table 1. The relation between object classes.

Relation	Strength
Dependence	D1
Association	D2
Aggregation	D3
Composition	D4
Generalization	D5

2.2.2. CK metrics

Traditional software metrics focus on structured software development paradigm, including functional decomposition and data flow analysis. Since the object-oriented software development paradigm differs significantly from the traditional one, the existing metrics are not suitable. The theoretical research and practice of object-oriented software metrics have become one of the hot topics in the field of software engineering. In this paper, we measure the strength of relations between object classes with improved CK (Chidamber and Kemerer) group metrics, namely, WMC (weighted methods per class) and LCOM (lack of cohesion in methods).[29] The details are given in the following equation:

$$CK_{i\to j} = \alpha_1 \times W_{i\to j} + \beta_1 \times L_{i\to j}, \tag{1}$$

where $W_{i\to j}$ and $L_{i\to j}$ are WMC and LCOM between classes i and j, respectively; α_1 and β_1 are weighting coefficients, satisfying $\alpha_1 + \beta_1 = 1$.

Considering the above two types of dependency metrics together, a weighted software network is established (see Fig. 1(c)), where the weights of the edges can be determined as

$$w_{i\to j} = \alpha_2 \times R_{i\to j} + \beta_2 \times CK_{i\to j}. \tag{2}$$

Here, α_2 and β_2 are weighting coefficients, satisfying $\alpha_2 + \beta_2 = 1$. $R_{i\to j}$ denotes the strength of dependency between classes i and j as described in subsection 2.2.1.

3. Method

3.1. Distance

The software network can be mapped to an adjacent matrix $\boldsymbol{W}$,

$$\boldsymbol{W} = \begin{bmatrix} w_{1\to1} & w_{1\to2} & \cdots & w_{1\to N} \\ w_{2\to1} & w_{2\to2} & \cdots & w_{2\to N} \\ & & \vdots & \\ w_{N\to1} & w_{N\to2} & \ldots & w_{N\to N} \end{bmatrix}, \tag{3}$$

where $w_{i\to j}$ is the edge weight on which object classes i and j directly depend. On this basis, the shortest path length between classes (i.e., the distance between them) is

$$d_{ij} = \min(w^P_{i\to j_1} + w^P_{j_1\to j_2} + \cdots + w^P_{j_{m-1}\to j_m} + w^P_{j_m\to j}), \tag{4}$$

where j_m is the class on the path, and P is a real number. When $P < 0$, the greater the weight is, the smaller the distance is, and vice verse. How closely an object class is dependent on other classes is measured by the closeness centrality, defined as

$$C_i = \frac{N_0 - 1}{\sum_{j\in N}(d_{ij})}, \tag{5}$$

where N denotes the network and N_0 is its size. C_i denotes the closeness centrality of class i.

3.2. Multifractal analysis

MEMB is the classical box-covering algorithm that tries to find excellent nodes as the box center by the concept of "exclusion mass". The "excluded mass" of a node is the number of uncovered nodes whose distance from it is less than a given

radius. Since the algorithm always selects the center among non-center nodes, including the covered nodes, a node may be covered several times. The choice of the center node has an impact on the effectiveness of the algorithm.

The algorithm in this paper is an improvement of MEMB, which selects the center node by the radio of excluded mass to closeness centrality (REMCC). Assume that the excluded mass of class i is M_i. Define the variable f_i to be

$$f_i = \frac{M_i}{C_i}. \tag{6}$$

The node with the maximum f_i is selected. Next, we elaborate the REMCC algorithm. The normalized probability in the REMCC algorithm is

$$\mu_i(r) = \frac{M_i(r)}{N_0}, \tag{7}$$

where r is the box size, satisfying $0 < r <= N_0$; μ is the probability measure, and $M_i(r)$ denotes the amount of classes in the i-th box.

The main methods for portraying multifractal are generalized multifractal dimension and multifractal spectrum. Consider the partition sum of Eq. (7):

$$Z_q(r) = \sum_{i}^{N_i(r)} \mu_i^q, \tag{8}$$

where $N_i(r)$ is the number of boxes when the radius is r, and q is the moment of order. When the dependencies between classes have a multifractal distribution, it has the following form:

$$Z_q(r) \sim r^{\tau_q}, \tag{9}$$

where τ_q is the mass exponent of moment q. If the partition function is defined by Eq. (8), the mass exponent is

$$\tau(q) = \lim_{r\to 0} \frac{\log(Z_q(r))}{\log(r)}. \tag{10}$$

The mass exponent $\tau(q)$ is a strictly monotonic increasing function. It is a concave function that describes whether the research object has distinct multifractal characteristics. If it is linear, the system will be monofractal, otherwise it will be multifractal.[30] The generalized multifractal dimension is extended on this basis and defined as

$$D(q) = \lim_{r\to 0} \frac{\log\left(Z_q\left(r\right)\right)}{\log\left(r\right)} \cdot \frac{1}{q-1}, \quad q \neq 1. \tag{11}$$

It is a strictly monotonic decreasing function with respect to q. The spectrum function $f(\alpha)$ is usually obtained via the Legendre transform:

$$\begin{cases} \alpha_q = \frac{d\tau_q}{d_q}, \\ f\left(\alpha_q\right) = q\alpha_q - \tau_q, \end{cases} \tag{12}$$

where α indicates the velocity at which the mass changes with r, and it determines the singularity strength. The spectrum function is concave, monotonically increasing at $q > 0$ and monotonically decreasing at $q < 0$. It provides a more natural, intuitive and accurate mathematical description for multifractal.

Algorithm 1 gives the concrete practice steps of the REMCC algorithm, which works for the multifractal analysis of software networks. In particular, the excluded mass of the node is counted as the number of classes in the box. R in line 8 is a value in the range of r.

Algorithm 1 REMCC algorithm for software networks

Require:
The set X of all classes and their dependencies.
Ensure:
Generalized multifractal dimension $D(q)$, multifractal spectrum function $f(\alpha)$.
1. Build the directed weighted software network based on class dependencies and Eq. (2);
2. **for** i in X **do**
3. $f_i = \frac{M_i}{C_i}$; 4. **end for**
5. $f_p = \max_i \{\frac{M_i}{C_i}\}$; /*Select the class p with maximum f_i as the center; */
6. **for** i in X **do**
7. $d_{ip} = \min(w^P_{i\to p_1} + w^P_{p_1\to p_2} + \cdots + w^P_{p_{m-1}\to p_m} + w^P_{p_m\to p})$; /* Calculate the distance between classes i and p*/
8. **if** $d_{ip} < R$ **then**
9. $X = X - i$ /* Update set X */
10. **end if**
11. **end for**
12. **if** $X! = \varnothing$ **then**
13. Return to step 2;
14. **end if**
15. Calculate the distribution of box masses (Eq. (7)).
16. Calculate the partition sum (Eq. (8)).
17. The mass exponent τ is fitted with partition sum and radius r by least-squares method (Eq. (10)).
18. Calculate the generalized dimension $D(q)$ and spectrum function $f(\alpha)$ (Eq.(11) and Eq. (12)).
19. **Output** $D(q)$, $f(\alpha)$.

3.3. Case analysis

An example is given to illustrate that REMCC can achieve fewer boxes compared to MEMB. Given a network, as shown in Fig. 2(a), the characteristics of the nodes in the initial state are determined as listed in Table 2.

The detailed process of the two algorithms on this network is shown in Fig. 2. The box size is set to 1. If the center is selected according to the excluded mass (MEMB), node 1 is selected. The first box covers 7 nodes (Fig. 2(b)), and the remaining nodes belong to each of the eight boxes (Fig. 2(c)). On the other hand, node 2 is selected if based on f_i (REMCC). There are 6 nodes covered by the first box, as shown in Fig. 2(e). In the end, the two algorithms produce distinct box-covering results on the same network. MEMB requires 9 boxes, while REMCC requires only 6 boxes. REMCC has superior results. Specifically, REMCC takes into account the topological characteristics of the network, i.e., the closeness centrality of nodes. Therefore, there will not be a situation where some leaf nodes are isolated by boxes.

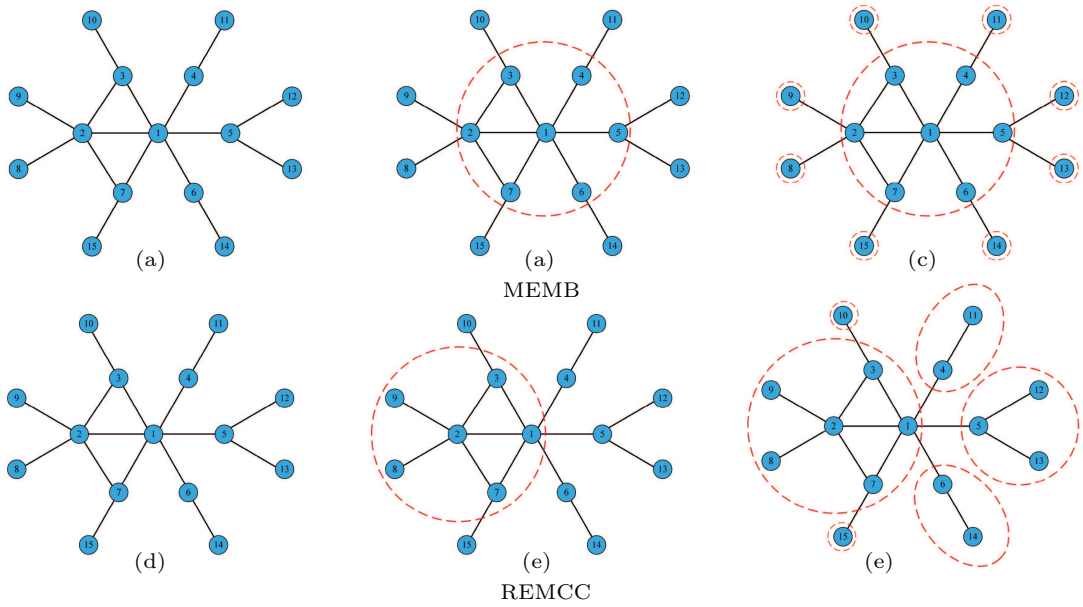


Fig. 2. Comparison of algorithms.

Table 2. Node characteristics in the initial state.

No.	$id=1$	$id=2$	$id=3,7$	$id=4,6$	$id=5$	$id=8,9$	$id=10,15$	$id=11,14$	$id=12,13$
M_i	7	6	4	3	4	2	2	2	2
C_i	0.6364	0.5185	0.4667	0.4242	0.4516	0.35	0.3256	0.3043	0.3182
f_i	10.9994	11.5718	8.5708	7.0721	8.8574	5.7143	6.1425	6.5725	6.2854

4. Experiments

4.1. Data extraction

Multifractal analysis is carried out in different Java frameworks: myabtis, dubbo, cxf and fastjson. In detail, mybatis is a persistence framework that supports custom SQL, stored procedures and advanced mappings; dubbo enables applications to realize service input and output through high-performance RPC (Remote Procedure Call); cxf helps you build and develop services using front-end programming APIs; fastjson makes it easy to complete object-to-object and object-to-string conversions. These four frameworks are object-oriented systems, but they have distinctly different functions. Table 3 lists the network topology characteristics, such as the number of nodes and edges.

Additionally, different versions of the myabtis software network are considered and their topology characteristics are given in Table 4. The scale of the network increases with the release, and its topology metrics change accordingly, especially from 3.1.1 to 3.3.0. The increase of clustering coefficient and the decrease of average path length mean that the network progressively exhibits small-world characteristics.

Table 3. Datasets for different software networks.

Network	Nodes	Edges	Average degree	Clustering	Average path length
mybatis 3.3.0	1066	9012	16.9081	0.2300	1.4469
dubbo 2.6.1	1755	11891	22.3096	0.1572	0.1579
cxf 2.7.18	1237	8005	15.0188	0.1672	0.2779
fastjson 1.2.47	522	3328	6.2439	0.1948	0.6902

Table 4. Datasets for different versions of mybatis.

Version	Nodes	Edges	Average degree	Clustering	Average path length
2.3.5	404	2880	5.4034	0.1540	0.7958
3.0.1	623	4431	8.3133	0.1900	0.8747
3.1.1	639	4744	8.9006	0.1916	1.1207
3.3.0	1066	9012	16.9081	0.2300	1.4469

In general, hubs in small-world networks tend to have larger degrees. It is known from the case in Section 3.2 that REMCC has better results for this type of network (larger degree of the center node). In other words, REMCC is effective in software networks.

4.2. Results

Figure 3(a) represents the mass exponent functions $\tau(q)$ of different software networks and they are nonlinear. This proves that these software networks are multifractal as to different dependencies. The prerequisite for evaluating the variation of the generalized multifractal dimension $D(q)$ with q is to calculate the partition sum (Eq. (8)). In this paper, it is computed from $q = -10.0$ to $q = 10.0$ in steps of 0.04, and the results is shown Fig. 3(b). The singularity of the object class distribution is correlated with the amplitude of $D(q)$. It is often used to quantitatively measure the degree of multifractality, so that the wider the spectrum is, the more multifractal it is. Specially, $D(q)$ is a horizontal function in the homogeneous case. It can be seen that fastjson has the strongest singularity in class distribution.

Likewise, we can conclude that different versions of the myabtis are multifractal (Fig. 4(a)) and that the singularity of the dependency distribution is reduced (Fig. 4(b)). The singularity is manifested heterogeneity of spatial distribution in the software network. In Figs. 3(b) and 4(b), the left tail of the function $D(q)$ depicts the low dependency, while the high dependency is associated with the right tail. Classes that are frequently invoked by other modules tend to have a larger degree, while classes with specific functionality that are only used in one module have a small degree. There are significant differences in the spatial distribution of nodes with large degrees.

The maximum, width and asymmetry of the spectrum for different software networks are summarized in Table 5. Generally speaking, the larger the width, the larger the maximum and the better the symmetry, the more pronounced the multifractal characteristic of the network. As before, fastjson has stronger multifractal performance compared to other software. We polyfit the spectrum curve by the least squares method, see Fig. 5, where the red line indicates the binomial curve. The fact that the spectrum function is parabolic implies that the network is multifractal.

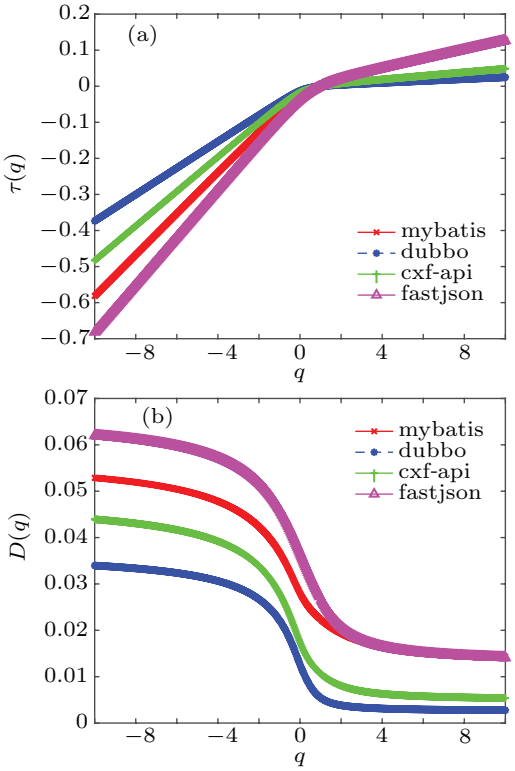


Fig. 3. Results of four software networks: (a) the mass exponent $\tau(q)$, (b) the generalized multifractal dimension $D(q)$.

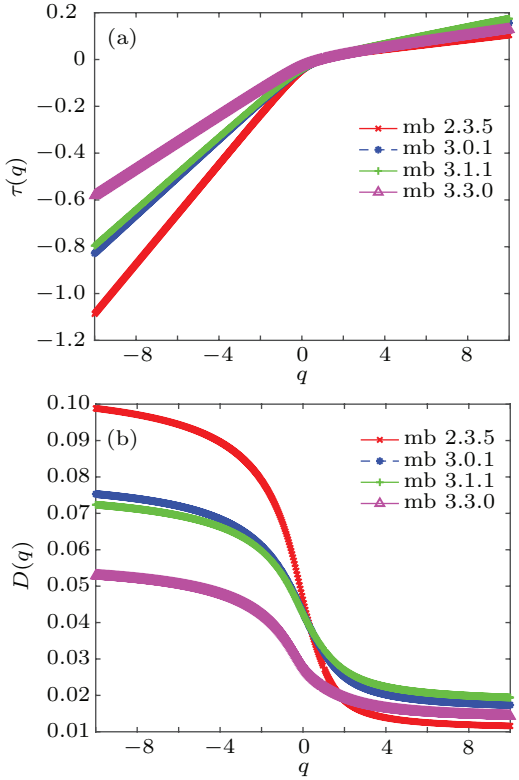


Fig. 4. Results of mybatis software networks: (a) the mass exponent $\tau(q)$, (b) the generalized multifractal dimension $D(q)$.

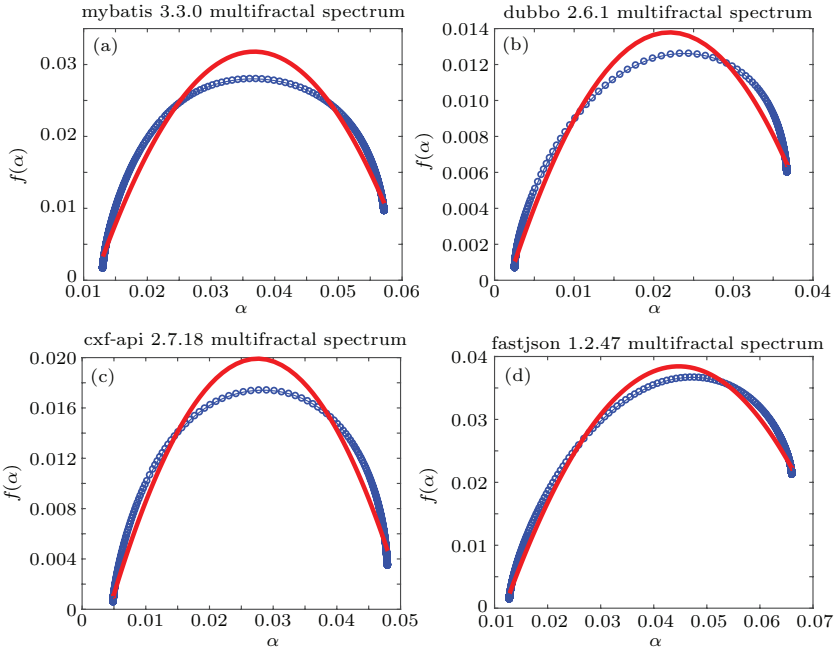


Fig. 5. Multifractal spectrum curves for different software networks: (a) mybatis, (b) dubbo, (c) cxf, (d) fastjson.

Table 5. Multifractal spectrum characteristics for different software networks.

Software	Width	Maximum	Asymmetry
mybatis	0.0442	0.0280	3.7178
dubbo	0.0342	0.0126	1.4848
cxf	0.0431	0.0174	2.0252
fastjson	0.0532	0.0367	3.1792

The multifractal spectrum characteristics for different versions of mybatis are shown in Table 6. Narrower spectrum, smaller maximum and increasing asymmetry suggest that the multifractality of mybatis gradually weakens. As a result, the singularity of dependencies between classes is decreased.

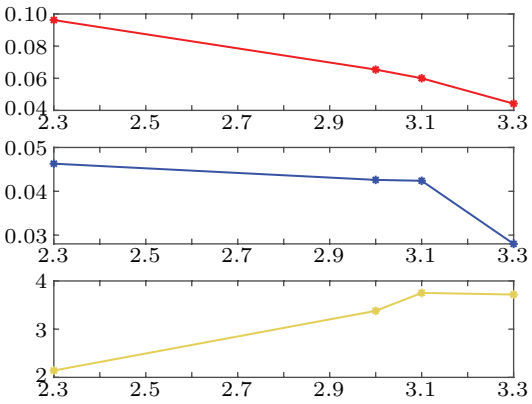


Fig. 6. The width (red line), maximum (blue line) and asymmetry (yellow line) of spectrum for different mybatis versions.

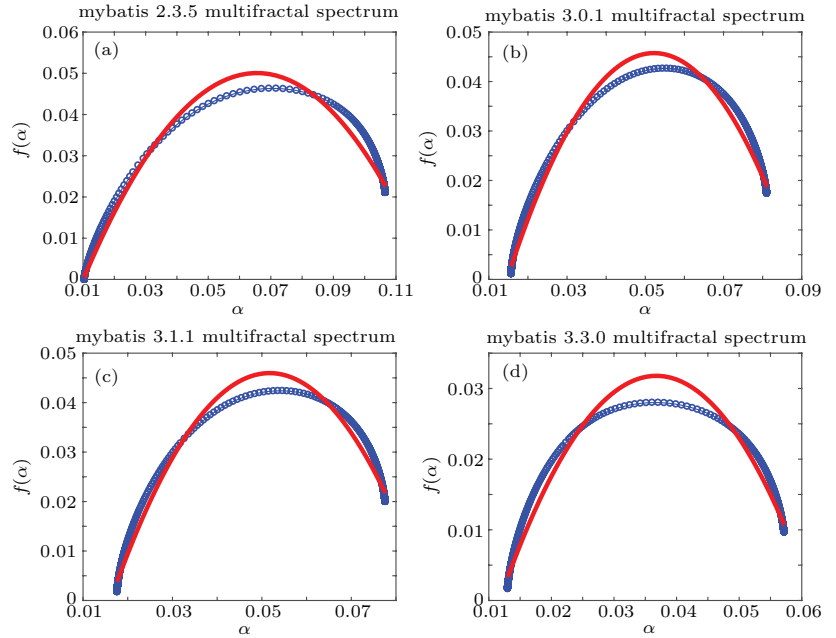


Fig. 7. Multifractal spectrum curves for different versions of mybatis: (a) mybatis 2.3.5, (b) mybatis 3.0.1, (c) mybatis 3.1.1, (d) mybatis 3.3.0.

Table 6. Multifractal spectrum characteristics for different versions of mybatis.

Version	Width	Maximum	Asymmetry
2.3.5	0.0962	0.0463	2.1416
3.0.1	0.0654	0.0426	3.3798
3.1.1	0.0600	0.0424	3.7520
3.3.0	0.0442	0.0280	3.7178

The results in the table are visualized in Fig. 6. Specifically, the spectrum width is reduced by 44.4% from version 2.3.5 to version 3.3.0. In fact, during the software improvement process, some inapposite dependencies between classes are removed by the new version, thus the singularity in dependencies between classes is gradually reduced. This trend causes a more specific division of functions between software modules.

Figure 7 displays the spectrum function curves for different versions of mybatis, which emphasizes the multifractal nature as well. As we can observe from Figs. 5 and 7, the spectrum curves have a right truncation and a long left tail. Apparently, $f(\alpha)$ increases considerably on the left side of its maximum, while it varies slightly on the right side.

5. Conclusions

In summary, a multifractal analysis approach based on modified box-covering algorithm is introduced to discuss the software structure. We analyze the dependencies of class objects in software at the code level to construct software network. The experiments are performed on four representative software and they are found to be multifractal. Moreover, they have different functions and are applied to different modules of the system. In other words, even functionally different software has similarities in architecture design. Furthermore, simulation on different versions of mybatis demonstrates that the singularity of dependencies between object classes decreases with the optimization of the software. The methodology and results of this paper provide new insights into the evaluation and design of large-scale software systems. Future work will focus on the relationship between software defects or reliability and the multifractal property of software networks.

Acknowledgement

Project supported by the National Natural Science Foundation of China (Grant Nos. 61877067 and 61572435).

References

[1] Martins L and Tony G 2005 *IEEE Trans. Softw. Eng.* **46** 346
[2] Wilfredo T 2000 *Software Fault Tolerance: A Tutorial* (NASA Langley Technical Report Server)
[3] Myers and Christopher R 2003 *Phys. Rev. E* **68** 046116
[4] Ai J, Su W, Zhang S and Yang Y 2019 *IEEE Trans. Rel.* **68** 844
[5] Chong C Y and Lee S P 2015 *J. Syst. Softw.* **110** 28
[6] Yan D and Qi G N 2006 *Acta Phys. Sin.* **55** 3799 (in Chinese)
[7] Ma Y T, He K Q, Li B and Liu J 2011 *J. Softw.* **22** 381
[8] Gao Y, Peng Y, Feng X, Chen D, Zhao W and Xu G 2012 *J. Tsinghua Univ.* **52** 1474
[9] Wang B Y and Lü J H 2013 *J. Softw.* **24** 2814
[10] Briand L C, Daly J W and Wust J 1999 *Comput. Standards Interfaces* **21** 165
[11] Kuang L, Zheng B, Li D, Li Y and Sun Y 2015 *Sci. China Inf. Sci.* **58** 1
[12] Song, Chao M, Havlin, Shlomo, Makse and A Hernán 2005 *Nature* **433** 392
[13] Concas G, Locci M F, Marchesi M, Pinna S and Turnu I 2006 *Europhys. Lett.* **76** 1221
[14] Concas G, Marchesi M, Murgia A, Pinna S and Tonelli R 2010 *Proceedings of the* 2010 *ICSE Workshop on Emerging Trends in Software Metrics*, May 4, 2010, Cape Town, South Africa, p. 24
[15] Turnu T, Concas G, Marchesi M and Tonelli R 2013 *Inf. Sci.* **245** 290
[16] Peitgen H O, Jürgens H and Saupe D 2004 *Math. Gazette* **79** 241
[17] Song C, Gallos L K, Havlin S and Makse H A 2007 *J. Stat. Mech.* **2007** P03006
[18] Zhou W X, Jiang Z Q and Sornette D 2007 *Physica A* **375** 741
[19] Zheng W, You Q, Zhang Z and Pan H 2019 *IEEE Access* **7** 68522
[20] Chiranjib B, Hirok C, Argha D, Debasis G and Bikash S 2015 *Nat. Hazards* **78** 855
[21] Zhang Q, Xi X and Luo Z 2013 *Chin. J. Sens. Actuators* **26** 282 (in Chinese)
[22] Liu H Z, Zhang X C and Zhang X 2020 *Physica A* **545** 123585
[23] Zhang X, Liu H Z, Zhao Y F and Zhang X C 2019 *Physica A* **531** 121790
[24] Wang J, Shang P and Cui X 2014 *Phys. Rev. E* **89** 032916
[25] David M T, Vargas F V, Chávez H L and Hernández C 2016 *Expert Syst. Appl.* **62** 373
[26] Mishra A, Bandyopadhyay J N and Jalan S 2021 *Chaos, Solitons and Fractals* **144** 110745
[27] Jiang Z Q, Xie W J, Zhou W X and Sornette D 2019 *Rep. Prog. Phys.* **82** 125901
[28] Zhang A H, Li X W, Su F G and Zhang Y 2015 *Chin. Phys. Lett.* **32** 90501
[29] Chidamber S R and Kemerer C F 1994 *IEEE Trans. Softw. Eng.* **20** 476
[30] Zheng W, Pan Q, Deng Y F, Zhao X K and Kang Z 2016 *Chin. Phys. Lett.* **33** 038901