STUDY ON SOFTWARE NETWORK CONTROLLABILITY USING THE SOURCE-DRIVEN MODEL

Dr. Anand Singh Rajawat[1], Prof. Namrata D. Ghuse[2], Mr. Mahesh Vijayrao Korde[3]

Article History: Received: 12.12.2022 **Revised:** 29.01.2023 **Accepted:** 15.03.2023

Abstract

With software evolution, software system complexity has risen, which has an impact on the stability of software structure. The majority of the measurement techniques currently in use concentrate on the analysis of network topology's macro-characteristics, but they lack sufficient depth and breadth to explore the nature of the complexity of the software structure. To address this issue, the complex network control theory was applied to the investigation of software network controllability. Initially, based on system control theory, the Source-Driver (SD) model was created, and the driver node sets were constructed using the minimal input theorem during the management of the software network architecture; Due to the non-uniqueness of the driver node sets, the relationship between the degree and centre degree as well as the relationship between the in-degree and the out-degree of the software network architecture were then further examined; The four indicators' values in the software system were then compared. The action on control nodes and driver nodes is not random, the controllability of the driver nodes is closely related to the in-degree, and when selecting the driver node sets, the network topology characteristics should be taken into full consideration. Experimental results show that the driver node sets in software networks are primarily composed of nodes with low degree values, but this does not mean that the nodes whose in-degree values and out-degree values are also low. The findings have significant guiding implications for the management, upkeep, and redesign of software architecture.

Keywords: Network control, software networks, driver node sets, and complex networks.

[1]Associate Professor, Department of Computer Science & Engineering, School of Engineering and Technology, Sandip University, Nashik, Maharashtra 422213
[2]Assistant Professor, Department of Computer Science & Engineering, Sandip Institute of Technology & Research Centre, Nashik, Maharashtra 422213
[3]Assistant Professor, Department of Computer Science & Engineering, Sandip Institute of Engineering and Management, Nashik, Maharashtra 422213

Email: [1]anand.rajawat@sandipuniversity.edu.in, [2]namrata.ghuse@sitrc.org, [3]mahesh.korade@siem.org.in

DOI: 10.31838/ecb/2023.12.s3.142

1. Introduction

With the fast advancement of big data technology in the setting of the Internet era, software systems based on networks have quickly grown in size, number of users, and interaction of component pieces, becoming a type of complex system (Wang et al., 2019). Understanding the fundamental properties of software requires the interdisciplinary integration of software engineering and complex systems, abstract software systems into the artificial complex networks (i.e., software networks) from the perspective of complex systems and complex networks, which explore and discover the structural characteristics, evolution rules, and behaviour characteristics of complex software systems from the whole and global perspectives.

During the last ten years, software networks have been developed, and several research techniques have produced findings in the assessment of software static structural features and software dynamic behaviour(Ma et al., 2010). Yet as the application environment becomes more intricate, software development becomes more dangerous. Software product quality is difficult to ensure. People are investigating ways to make complex networks better as complex networks theory research advances (P. Li et al., 2010). People have been interested in learning how to successfully regulate the structure and behaviour of complex dynamic systems since since (H. Pan et al., 2019) published their groundbreaking study on structural control of complex networks in 2011. By presenting the matching theory of directed graphs and the structural controllability theorem, he developed the controllability model of complex networks. To fully control the network, he input the control signals to the least driver node sets. Based on this, (W.-F. Pan et al., 2013) used the PHB matrix criteria and the eigenvalues of the network dynamics coupling matrix to calculate the number of control input nodes. In order to identify the smallest driver node sets in networks and examine the node structure inside them, (Wu et al., 2018) presented a sort of random matching approach. In-degree and out-degree, in-degree and in-degree, out-degree and out-degree, and out-degree and out-degree correlation coefficients were established by (Jiang et al., 2018) and their link to network controllability was examined. The controllability of real network edges was studied by (W.-F. Pan et al., 2013) using controllable limit theory to investigate the impact of the correlation between out-degree and in-degree (i.e. degree correlation). Through analysis and calculation, it was revealed that the upper and lower controllable limits were applicable to all varieties of real networks, which advanced our understanding of the controllability of real network edges.

The study of complex networks gives compelling evidence in favour of investigating the structural properties of massively parallel software systems. To improve the quality of software systems, (Guan-Rong, 2013), and others developed an enhanced box covering technique to examine the multi-fractal of software networks. By examining only a few number of essential functional nodes, (H. Li, 2020) examined the stability, dependability, and resilience of software networks and developed a new approach for mining the top-k critical nodes in directional weighted complex software networks based on semi global information. The interdependence of functions in various pieces of software was used by (Yuan et al., 2013) to identify susceptible nodes and offer an efficient approach for evaluating software leaks. There has been minimal study on the controllability of software network topology up to this point, despite the relative maturity of the applications of software system complexity assessment, topological characteristics, and information transfer. An audacious attempt to broaden the scope of complicated network control research is the application of the network controllability theory to software networks. Software networks share comparable network topology features with traditional complex networks (Tian et al., 2022), and thus invariably experience certain structural control issues. More research is required to see whether it exhibits comparable distinctive rules to the control features of complex networks.

The "Source-Driver" (SD) model, which expresses the control process of networks in an understandable and straightforward manner, is established in this paper based on research on the controllability of complex networks, prior experience, and analysis methods of software measurement. Next, software networks are applied to the SD model to study the controllability of software networks combined with the characteristics and rules of the software. Lastly, in order to investigate how to choose driver node sets in networks efficiently, the distribution relation on in-degree, out-degree, degree, and center-degree is thoroughly examined. The results will serve as a reference point for software system maintenance.
The remaining portions of the essay are structured as follows: The foundation for network controllability analysis and some fundamental theories concerning network control aspects are introduced in Section 2. The software network controllable model (SD model) is developed in Section 3. The experiments in Section 4 cover the SD model software network topology control process and assess the distinctive relationships between various software versions to investigate the choice of driver node sets. Ultimately, the

article comes to an end, and Section 5 provides guidance for the future.

2. Process and Technique of Control

2.1. Fundamental Concepts and Meaning

Definition a single control node It speaks about the nodes that receive the control signals. Control signals are utilised among them for receiving and promptly forwarding to any network node.

Definition Control nodes that do not share the same input nodes are the two driver nodes(Pósfai et al., 2013). The minimal driver node sets are the bare minimum of driver nodes necessary to properly control the states of every node in the networks.

Definition 3 Center-degree (Liu & Pang, 2020): This term refers to the reciprocal of the total of the shortest paths between each node, multiplied by N1. It is written as follows:

$$C_i = (N-1) \Big/ \sum_{i=1}^{N} d_{ij}$$

where dij is the length of the shortest path between nodes I and j.

1st Theorem Theorem on minimum input: The minimal output set (MOS) is equal to the minimal input set (NI) (ND). The minimal input set is any node in the networks if the networks are perfectly matched; otherwise, it is the node set that remains unmatched after the maximum matching, as shown in the following example:

$$\mathrm{NI} = \mathrm{ND} = \max 1, \{ \mathrm{N} \times \mathrm{M} * \} \qquad (2)$$

where |M*| is the directed network's maximum number of matched nodes.

Kalman criterion (Wu et al., 2018) Theorem 2 If the matrix is NNM controlled,

$$\mathrm{W} = [\mathrm{B}, \mathrm{AB}, \ldots, \mathrm{A}^{\mathrm{N}\times 1}\mathrm{B}] \ (3)$$

The matrix must have full rank, or ank(W) must equal N, in order for the system to be controlled.

Theorem 3: Theorem of structural controllability (24). The system structure is manageable if the networks don't have expansive structure or unreachable nodes.

2.2. Network Controllability

A set of N dimensional vectors can be used in complex networks to describe the state of a node at a given time t $(t) = (x_1(t) \ldots x_N(t))^T$, expecting that the node reach the system state $x(t') = (x_1(t') \ldots x_N(t'))^T$ at moment t′. If nodes can inter-couple to achieve the desired state, they can be said to be controlled. A situation is uncontrolled if some states are hard to achieve and the only way to compel them is by inputting control signals from the outside. according to Figure 1.

About the system equation

$$x'(t) = Ax(t) + Bu(t)$$

$u(t) = (u_1(t) \ldots u_M(t))^T$ provides information about the input control signal's condition at time t. The adjacency matrix, or A, of the network's nodes reflects their connectedness and degree of interaction with one another. B is the input matrix (MN), which explains how the nodes and control signals are connected to one another.

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ a_{21} & 0 & 0 & 0 \\ a_{31} & 0 & 0 & a_{34} \\ a_{41} & 0 & 0 & 0 \end{bmatrix}; \quad \boldsymbol{B} = \begin{bmatrix} b_1 & 0 \\ 0 & b_2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix};$$

$$\boldsymbol{W} = \begin{bmatrix} b_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & b_2 & a_{21}b_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{31}b_1 & 0 & a_{34}a_{41}b_1 & 0 & 0 & 0 \\ 0 & 0 & a_{41}b_1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

One node's dynamics equation is

$$x_i' = \sum_{k=1}^{N} a_{ik} x_k(t) + \sum_{j=1}^{M} a_{ij} u_j(t)$$

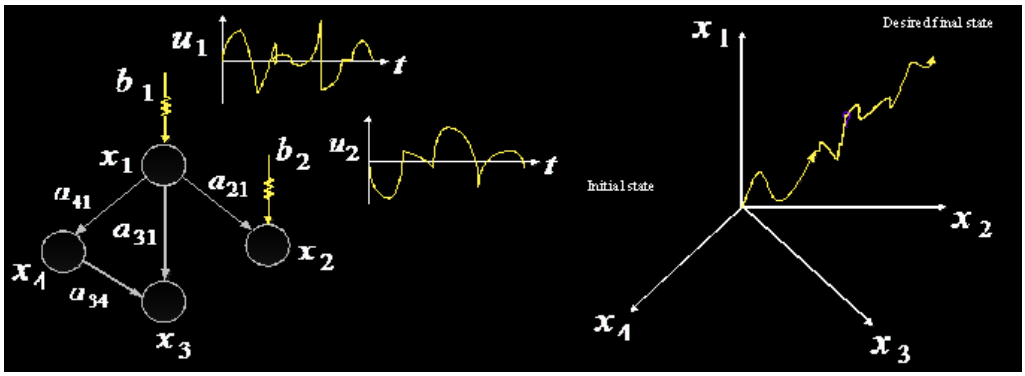

Figure 1: State space of nodes schematic diagram

The network is controllable if rank(W) = N, satisfying the Theorem 2 in this case.

In general, a driver node for an input signal u(ti) may be several. Making networks globally controllable with the fewest possible input control signals is the primary objective of network controllability technology. This subject was converted by (Jiang et al., 2018) into the lowest number of driver nodes that the control network needs, which fixes the main issue with network control. The maximum matching nodes and unmatched nodes in the directed graph should be solved in accordance with Theorem 1 and Theorem 3. The mismatched nodes are driver nodes, as indicated in Figure 2, and if all of the unmatched nodes in the network are driven by a single direct input signal, the entire network is fully controllable.

Figure 2 shows a tiny network as an illustration (a). The network is first turned into a bipartite graph structure, as shown in Figure 2(b). By looking at the greatest matching edge of this bipartite graph, it can be observed that v2, v3, v4, and v5 are matching nodes whereas v1 is an unmatched node, or a driving node. Figure 2 depicts the controlled state in detail (d). In order to manage the status of every node in the whole network, version 1 makes every node accessible.

3. Software networks' ability to be controlled

3.1 Class-Level Software Network Extraction

To get structural data, one must first analyse the programme source codes. Classes are the most fundamental components of object-oriented software, as seen in Figure 3(a), for instance. Software source code classes are represented as nodes in networks and class interaction relationships as edges. As shown in Figure 3(b), the class diagram must first be obtained in the software system before being abstracted into a network model (c).

3.2. Software Networks in a Controllable Mode

The majority of actual systems can be driven by nonlinear processes in practise, yet in many ways, such as structure, the controllability of nonlinear systems is remarkably comparable to that of linear systems. Every complex network may be dealt with using the controllable condition of the Kalman criteria, which can also be used to deal with the majority of nonlinear systems.

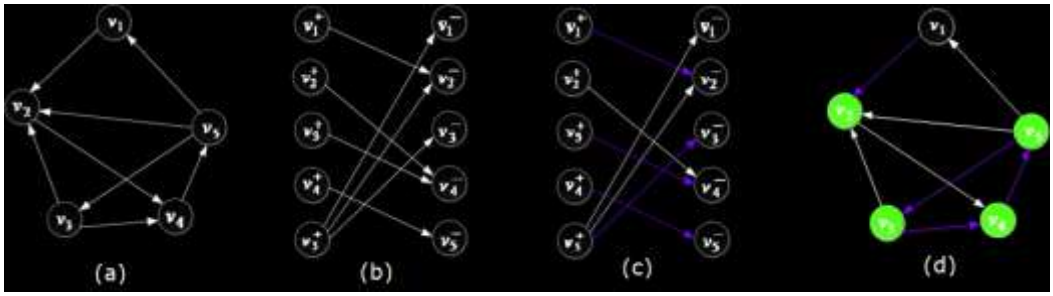

Figure 2: The process of creating driven node sets: Network topology, corresponding bipartite diagram, maximum matching edge set, and controllable state of the networks are all included.

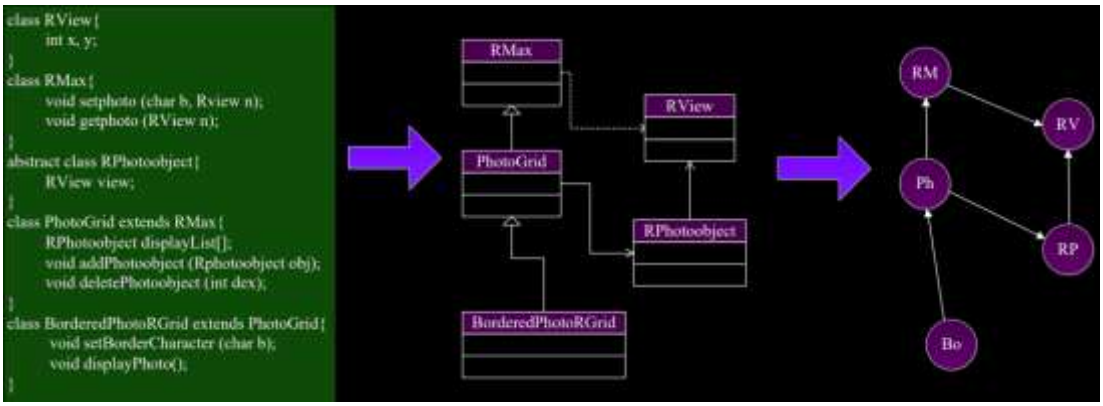

Figure 3: Network topology, a portion of the source code, and a class diagram are all examples of software networks.

Software engineering holds that class modules in object-oriented software structures should be thought of as nodes in a network, and calling behaviours between classes should be thought of as edges, creating software networks. Yet, communication and interaction among nodes are regulated and controlled processes. In order for the software system to function reliably across a range of application domains, measure and manage the software network topology from a macro viewpoint. Consequently, an essential issue in software network structure management is how to regulate network topology with cheap cost and great efficiency.

It has been proven that the Source-Driver (SD) model, which is based on an examination of the control process of complex networks, may be used to extract the SD model for the information transmission process in the software network architecture, as illustrated in Figure 4. It is separated into the source layer and the driver layer. Nodes are intended in two levels as controlled targets. Nodes in layer are source nodes (control nodes), sending control signals to other nodes on a selective basis. Matching nodes and driving nodes are the nodes in layer. The control signals supplied by the source nodes are immediately received by the driver nodes, who subsequently broadcast them to other nodes in the subnets at layer.

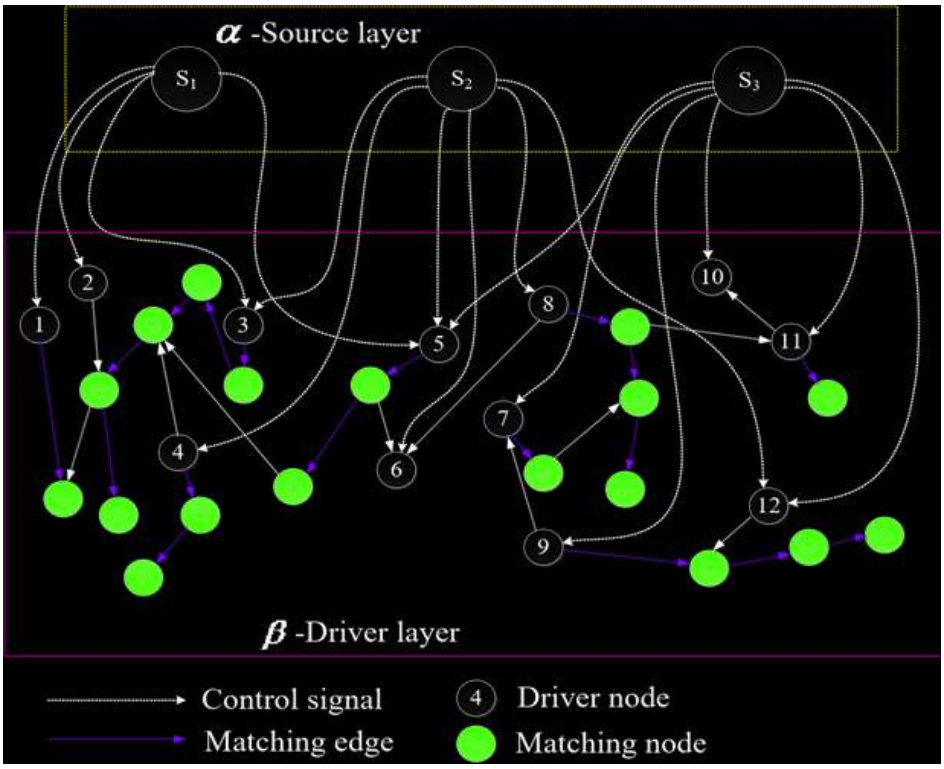

Figure 4: model SD

A directed controllable software network control condition on the SD model is shown in Figure 4. If there are three independent source nodes in layer, S1, S2, and S3, they transmit control signals to the

network's driver nodes at layer in order to manage the whole network. In addition, the signals conveyed by the source nodes to other nodes in layer are not entirely random, hence the production of driving node sets in complex networks is not singular. Contrary to other studies on the controllability of large networks, the pairing of driver nodes with other nodes in layer is undoubtedly not entirely random. The relationships, linkages, and information transfer inside software networks are built on the calls and dependencies between different software system components. As a result, source nodes in software networks selectively send control signals to the nodes in layer on SD model to service them. Clearly, more than one source node may control a driver node, and the more driver nodes a source node has control over, the more powerful their influence over the entire network is.

Under the SD model, source nodes communicate directly with driver nodes; in contrast to large networks, this reduces the number of driver nodes, which may be managed to control the whole network. As a result, software will have lower maintenance costs and more accurate measurements.

4. Observational Studies

Center-degree is a distinctive measure for identifying core nodes in object-oriented software networks, which are situated in the centre and have the biggest impact on the overall network. Degree in the structure of a software system indicates the reciprocal call patterns between local modules and reflects connections with nearby nodes. The direct control and being controlled states among distinct modules are represented by out-degree and in-degree. In light of the SD model framework, the degree distribution characteristic and the size of the driver node set in software networks are strongly associated.

To conduct the trials, choose 10 distinct software scales made with object-oriented programming languages like C++ and Java. Table 1 displays their fundamental network topology statistics after being extracted into software networks for various sizes.

Compile statistics on the attributes of software networks, including the driver node ratio, average out-degree, average degree, and average degree. Figure 5 displays their findings.

Figure 5 makes it intuitively clear that the average degree values of various software are all low, indicating that the majority of nodes have low degrees and that all software networks have average in-degrees that are higher than average out-degrees. With the exception of MySQL, all other software networks' average out-degrees are essentially the same as their average degrees, but for all of them, the average center-degree is the lowest. It illustrates how the software's module designs and interactions lessen overall complexity. All software networks have a driver node ratio of around 0.5, which is somewhat higher than that of other complicated networks. As illustrated by the computed results shown in Figure 4, one of the primary objectives of software design at the outset is to minimise the complexity of the software structure. Since the degree values of the majority of nodes in software networks are typically low, driving nodes and matching nodes will be generated here. On the other hand, the calling relationships (connection edges) between modules inside the software structure will increase if degree values are high for the majority of nodes. We may get the conclusion that nodes with low degree values typically make up the driving node sets. The better driver node is therefore not highly valued from the standpoint of control costs, which is essentially consistent with the literature's finding.

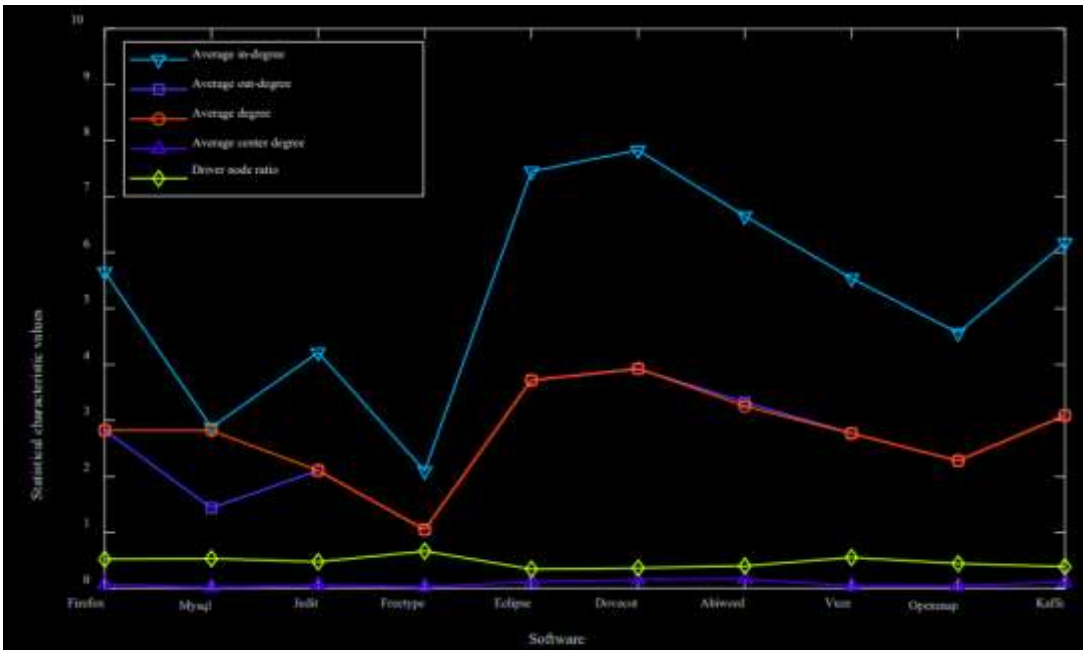

Figure 5: Topological software network statistical characteristics

Table 1: The dimensions of computer networks

	Firefox	Mysql	Jedit	Freetype	Eclipse	Dovecot	Abiword	Vuze	Openmap	Kaffe	Firefox	Mysql
N	10,116	3457	930	358	17,604	371	1300	3678	1932	7849	10,116	3457
L	29,672	5480	3359	387	32,718	729	2117	10,525	4518	12,120	29,672	5480

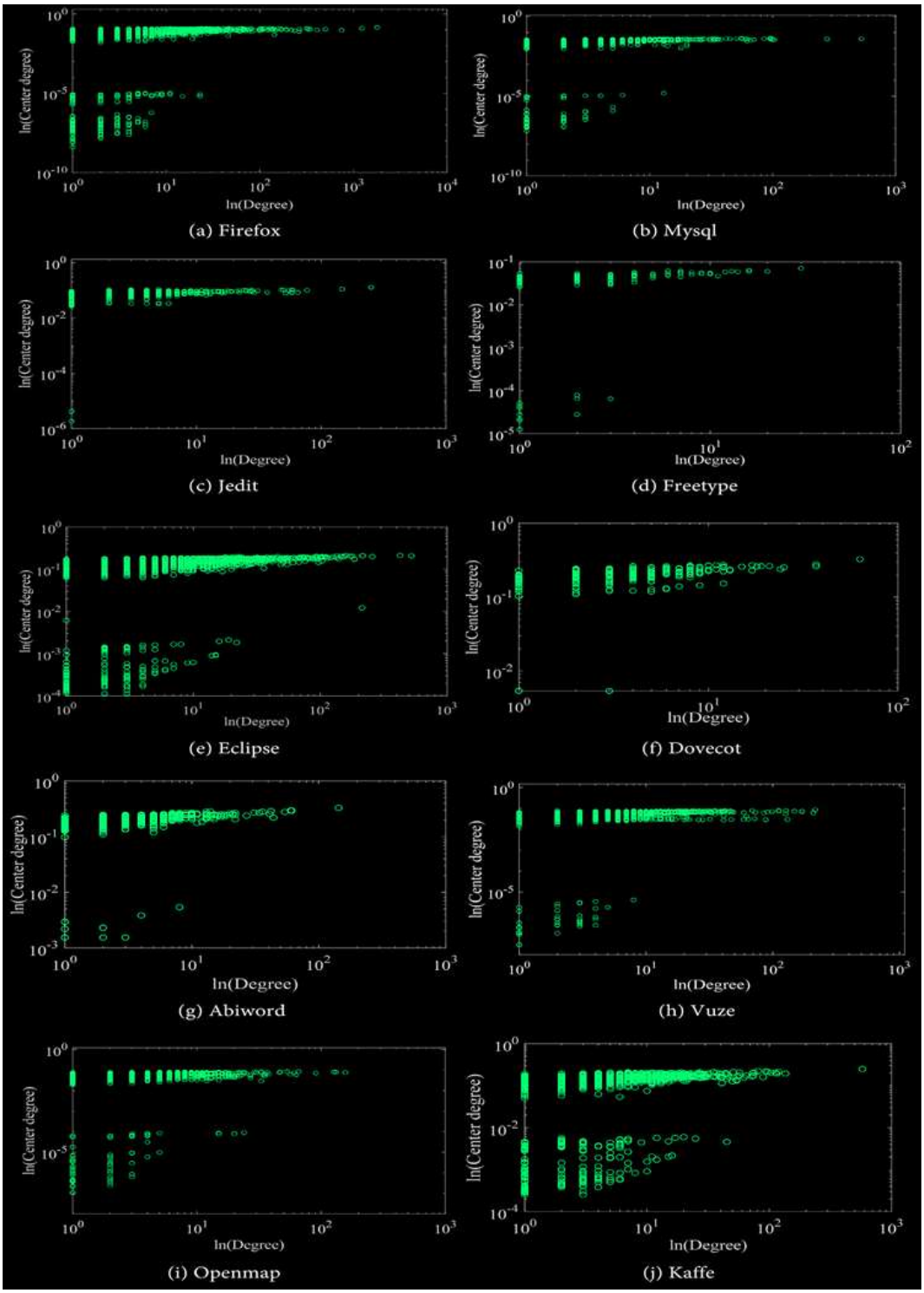

Figure 6: Relationship between centrality and distribution

A node's importance and controlling force, however, cannot be entirely explained by a degree indication. Nodes of intermediate degree or high degree can still be detected in the driver node sets, despite the possibility that there may be a large number of low degree nodes present. The degree distribution of nodes in software networks, like scale-free networks, complies with the power-law distribution. Lowering the degree value prevents the software system structure from being unnecessarily complicated. The attributes of a node itself should be taken into account while choosing driver node sets. Center-degree (Definition 3) is an indicator used in software networks to assess the significance and effect of nodes. The closer it is to the core location, which is the centre of the networks, the higher its value. According to Figure 6.

To examine the link between degree and center-degree of nodes in network architecture, use ten

distinct software scales. Take the double logarithmic coordinates of Figure 6 to observe clearly. The six software networks may be observed to have nodes of three different types: 1) High degree and center-degree nodes, with a maximum of 5 of these nodes in software networks. They often serve as core nodes and carry out the major operations of the software systems. 2) Nodes having a high center-degree but a low degree. Even though these nodes are a part of the software system's primary functional modules, they have simple constructions. The login screen in a typical software system is an example of this type of module. Nodes with low and middle degrees are number three. The majority of these nodes are equivalent to the fundamental components that software systems use to carry out some fundamental tasks, and the logical compositions of these modules are straightforward. Although it may be deduced that nodes with high center-degrees must also have high degrees, the converse may not be true, meaning that high center-degree nodes do not necessarily have high degrees. As a result, the first kind of nodes is the main factor to be taken into account while choosing the driver node sets.

The control extent of nodes is directly connected to their in-degree and out-degree when combined with the SD model. The driver nodes classify the incoming control signals using the in-degree indication and the output signals using the out-degree indicator. Hence, in-degree and out-degree have an impact on driver node actions together. The link between the in-degree and out-degree of nodes must thus be examined. These 10 software networks' orphan nodes and leaf nodes are dislodged, and the distribution diagram is then drawn using double logarithmic coordinates, as illustrated in Figure 7.

It is clear that most nodes in directed software networks exhibit a linear relationship between their in-degree and out-degree, whereas a few nodes exhibit deviation. This is also the reason why nodes in software networks often have low degrees. Also, the matching process between nodes is slowed down by the nodes in a tiny region below the straight line. A software system's design might reveal some intriguing patterns if you look closely. Figure 8 illustrates the network topology of Firefox as an example.

The degree value of each node is represented in the depicted diagram by its size, with different colours denoting different degrees. The greater the size of a node, the higher the degree value. The graph illustrates the concept of local aggregation. A few nodes are linked to them. Also, it demonstrates that most nodes' degree values are tiny, demonstrating that nodes with little degree values prefer to engage with nodes with big degree values.

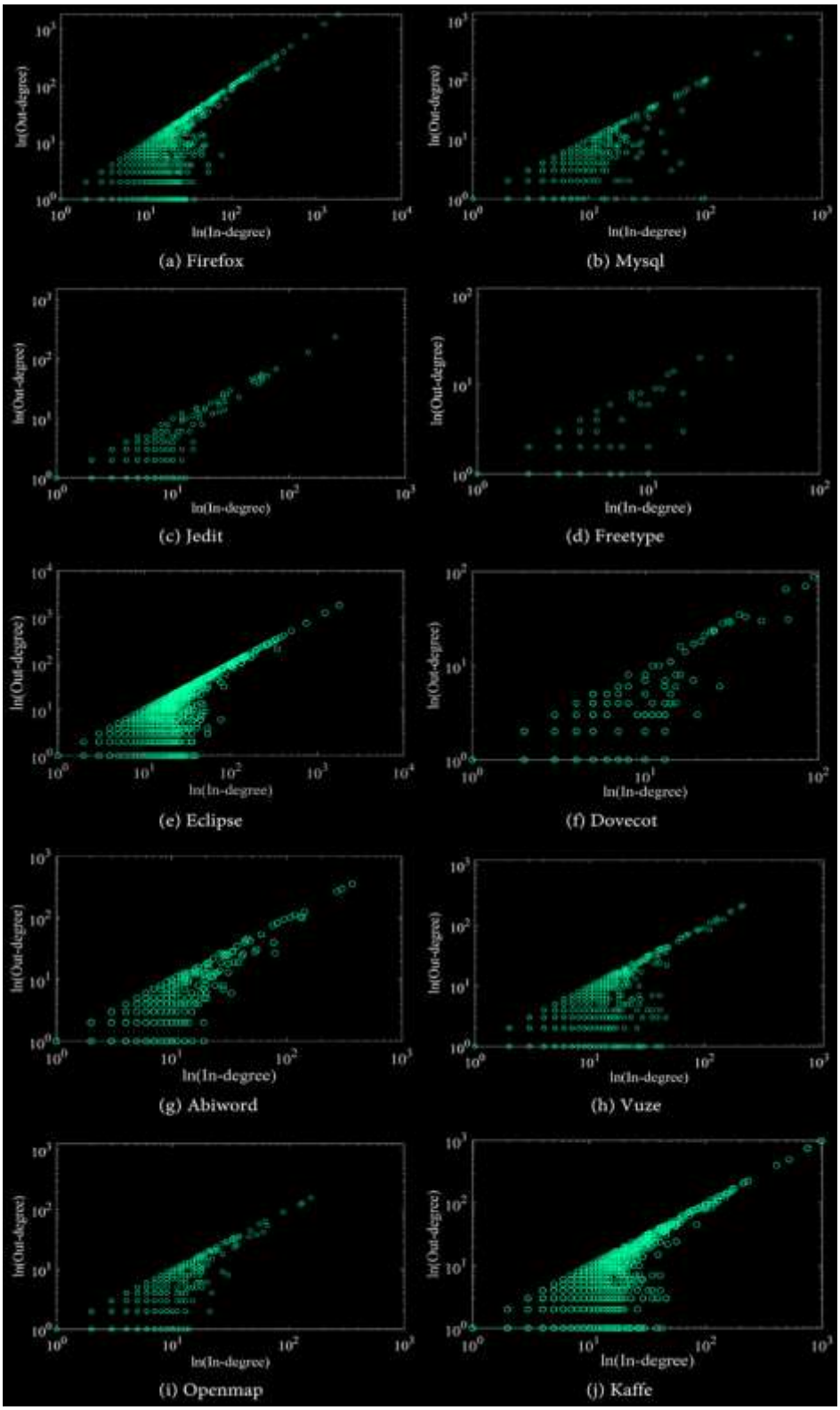

Figure 7: Relationship between in- and out-degree distribution

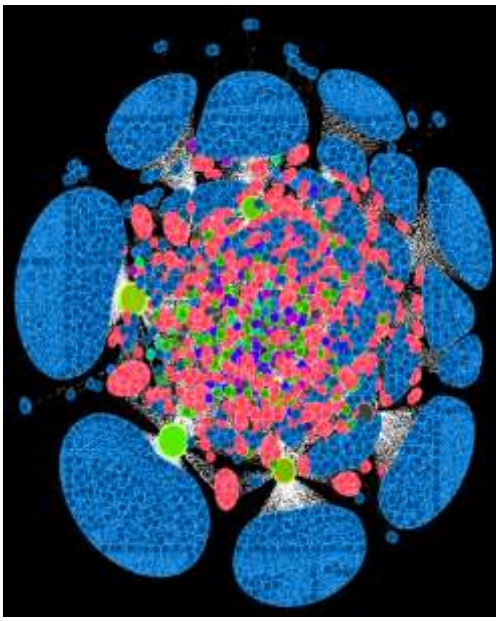

Figure 8: Firefox's software network topology

Table 2: Comparison of Firefox software network's defining characteristics

Node number	Node name	din	dout	d	C
5784	"nsISecurityPref" interface	1797	1797	0	0.1439
3507	"nsCOMPtr" class	1233	1223	10	0.1232
4611	"nsICSSPseudoElement" class	3	2	1	0.1231
4364	"nsHypotheticalBox" struct	410	398	12	0.0978
568	"already_AddRefed" struct	143	143	0	0.0487
6951	"nsRefPtr" class	347	203	144	0.1095
9999	"XPCWrappedNativeScope" class	15	3	12	6E-06

Compare the in-degree din, out-degree dout, degree d, and center-degree C of the six specific nodes you have chosen, which are presented in Table 2.
This leads to the discovery that the Node 5784 has the biggest center-degree value, but that its in-degree value and out-degree values are also the largest. In addition, its degree value is zero, which is an extreme case. The Node 4611, on the other hand, has the largest center-degree value despite having the smallest in- and out-degree values. These studies lead to the conclusion that control behaviours and states should be measured in terms of the in-degree and out-degree of nodes since degree values cannot accurately capture the scenario of sending and receiving signals among nodes in directed networks. In other words, in-degree and out-degree are crucial considerations while choosing the driver node sets.

5. Conclusions

People are constantly interested in researching the control challenge of complicated networks. The practical value of scientific research lies in expanding the application field, notwithstanding the continual abundance of theoretical research findings. The structural complexity of software networks may be measured from a novel angle by integrating control theory with software networks. This study has studied the controllability of software networks in this manner.
The choice of the driver node sets has a significant impact on the effectiveness of network control. In order to study the control procedure of software network topology, the SD model with two layers is constructed in this method. Based on it, the relationship between degree and center-degree as well as the topological features of software networks are analysed, and the findings show that while the majority of driver node sets tend to be made up of nodes with a low degree, nodes with

high center-degree should be given priority to take into account. After thoroughly exploring the software system structure, the correlation between in-degree and out-degree is then examined, and the four metrics are condensed. It is discovered that, despite some nodes' very small degree values, their in-degree and out-degree values are significantly different, and as a result, their center-degree values can change. As a result, such nodes should be carefully monitored when managing the software network structure. The crucial variables in node management are particularly in-degree and center-degree.

The controllability of software networks is still a very active area of study. To maintain the reliable and secure functioning of software, future research will focus on the topological properties of the driver node sets in the software network and their information transfer.

6. References

Guan-Rong, C. (2013). Problems and challenges in control theory under complex dynamical network environments. Acta Automatica Sinica, 39(4), 312–321.

Jiang, D., Huo, L., & Song, H. (2018). Rethinking behaviors and activities of base stations in mobile cellular networks based on big data analysis. IEEE Transactions on Network Science and Engineering, 7(1), 80–90.

Li, H. (2020). H∞ group consensus for partial-state coupled linear systems with fixed and switching topologies in the cooperation-competition networks. Journal of the Franklin Institute, 357(1), 314–342.

Li, P., Zhao, H., Qiao, Y., Liu, Z., Li, H., & Li, B. (2010). A platform of software network measurement design and implement. 2010 2nd International Conference on Advanced Computer Control, 2, 336–339.

Liu, S.-L., & Pang, S.-P. (2020). Effect of degree correlation on edge controllability of real networks. Chinese Physics B, 29(10), 100202.

Ma, Y.-T., He, K.-Q., Li, B., Liu, J., & Zhou, X.-Y. (2010). A hybrid set of complexity metrics for large-scale object-oriented software systems. Journal of Computer Science and Technology, 25(6), 1184–1201.

Pan, H., Zheng, W., Zhang, Z., & Lu, C. (2019). 软件网络分形结构特征研究 (Study on Fractal Features of Software Networks). 计算机科学, 46(2), 166–170.

Pan, W.-F., Jiang, B., & Li, B. (2013). Refactoring software packages via community detection in complex software networks. International Journal of Automation and Computing, 10(2), 157–166.

Pósfai, M., Liu, Y.-Y., Slotine, J.-J., & Barabási, A.-L. (2013). Effect of correlations on network controllability. Scientific Reports, 3(1), 1–7.

Tian, H., Meng, X., Hu, N., Wang, Y., & Yang, T. (2022). Research on the Controllability of Software Networks Based on the Source-Driven Model. Journal of Computer and Communications, 10(8), 26–40.

Wang, J., Zhang, K., Sun, X., Tan, Y., Wu, Q., & Wu, Q. (2019). Package network model: a way to capture holistic structural features of open-source operating systems. Symmetry, 11(2), 172.

Wu, J., Dong, M., Ota, K., Li, J., & Guan, Z. (2018). Big data analysis-based secure cluster management for optimized control plane in software-defined networks. IEEE Transactions on Network and Service Management, 15(1), 27–38.

Yuan, Z., Zhao, C., Di, Z., Wang, W.-X., & Lai, Y.-C. (2013). Exact controllability of complex networks. Nature Communications, 4(1), 1–9