

SFFSlib: A Python library for optimizing attribute layouts from micro to macro scales in network visualization

Ke-Chao Zhang(张可超), Sheng-Yue Jiang(蒋升跃), and Jing Xiao(肖婧)

Citation: Chin. Phys. B, 2025, 34 (**5**): 058903. DOI: 10.1088/1674-1056/adcb98
http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

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

SNSAlib: A python library for analyzing signed network

Ai-Wen Li(李艾纹), Jun-Lin Lu(陆俊霖), Ying Fan(樊瑛), and Xiao-Ke Xu(许小可)
Chin. Phys. B, 2025, 34 (**3**): 038902. DOI: 10.1088/1674-1056/ada439

Node ranking based on graph curvature and PageRank

Hongbo Qu(曲鸿博), Yu-Rong Song(宋玉蓉), Ruqi Li(李汝琦), Min Li(李敏), and Guo-Ping Jiang(蒋国平)
Chin. Phys. B, 2025, 34 (**2**): 028901. DOI: 10.1088/1674-1056/ad9a9b

Source localization in signed networks with effective distance

Zhi-Wei Ma(马志伟), Lei Sun(孙蕾), Zhi-Guo Ding(丁智国), Yi-Zhen Huang(黄宜真), and Zhao-Long Hu(胡兆龙)
Chin. Phys. B, 2024, 33 (**2**): 028902. DOI: 10.1088/1674-1056/ad1482

Constructing refined null models for statistical analysis of signed networks

Ai-Wen Li(李艾纹), Jing Xiao(肖婧, and Xiao-Ke Xu(许小可)
Chin. Phys. B, 2021, 30 (**3**): 038901. DOI: 10.1088/1674-1056/abc2c4

Community detection in signed networks based on discrete-time model

Jianrui Chen(陈建芮), Li Zhang(张莉), Weiwei Liu(刘维维), Zaizai Yan(闫在在)
Chin. Phys. B, 2017, 26 (**1**): 018901. DOI: 10.1088/1674-1056/26/1/018901

SPECIAL TOPIC — Computational programs in complex systems

SFFSlib: A Python library for optimizing attribute layouts from micro to macro scales in network visualization

Ke-Chao Zhang(张可超)[1,3], Sheng-Yue Jiang(蒋升跃)[2], and Jing Xiao(肖婧)[1,†]

[1] *College of Big Data and Internet, Shenzhen Technology University, Shenzhen* 518118, *China*
[2] *School of Cyber Science and Technology, University of Science and Technology of China, Hefei* 230026, *China*
[3] *College of Information and Communication Engineering, Dalian Minzu University, Dalian* 116600, *China*

(Received 24 December 2024; revised manuscript received 20 March 2025; accepted manuscript online 11 April 2025)

Complex network modeling characterizes system relationships and structures, while network visualization enables intuitive analysis and interpretation of these patterns. However, existing network visualization tools exhibit significant limitations in representing attributes of complex networks at various scales, particularly failing to provide advanced visual representations of specific nodes and edges, community affiliation attribution, and global scalability. These limitations substantially impede the intuitive analysis and interpretation of complex network patterns through visual representation. To address these limitations, we propose SFFSlib, a multi-scale network visualization framework incorporating novel methods to highlight attribute representation in diverse network scenarios and optimize structural feature visualization. Notably, we have enhanced the visualization of pivotal details at different scales across diverse network scenarios. The visualization algorithms proposed within SFFSlib were applied to real-world datasets and benchmarked against conventional layout algorithms. The experimental results reveal that SFFSlib significantly enhances the clarity of visualizations across different scales, offering a practical solution for the advancement of network attribute representation and the overall enhancement of visualization quality.

Keywords: complex network visualization, layout algorithm, signed network, fuzzy community structure, social bot network

PACS: 89.75.Fb, 87.23.Ge, 05.10.–a

DOI: 10.1088/1674-1056/adcb98 **CSTR:** 32038.14.CPB.adcb98

1. Introduction

A complex network is a powerful data structure used to model intricate relationships and interaction patterns between entities.[1–3] In the modeling of complex networks, entities are typically represented as nodes, and the relationships between them are abstracted as edges connecting these nodes.[4] Complex networks provide a foundational framework for research in many disciplines, and the study of these networks has become increasingly attractive to a wider range of researchers. Complex networks can be applied in various important fields of scientific research, including sociology,[5,6] physics,[7,8] biology,[9,10] and computer science. The visualization of complex network data is vital for advancing the understanding of network structures and dynamics across diverse research domains, significantly contributing to deeper insights into network properties and underlying patterns.[11]

Network visualization methods are essential tools in the study of complex networks, these methods help researchers understand the internal structure of networks.[12] The internal structure of a network can be represented across various scales, from the microscopic to mesoscopic and macroscopic levels.[13,14] The microscale examines the fundamental components of a network, including nodes, edges, and their associated properties. Visualizing these elements allows researchers to delve into the functions and behaviors of the basic building blocks of the network. The mesoscale focuses on intermediate structures, such as communities and subgraphs, where visualization facilitates a more detailed analysis of local interactions and properties, enriching the understanding of network dynamics at this level. The macroscale highlights the global characteristics of the network, including its overall topology and properties, where visualization techniques offer a comprehensive view of the network's architecture and large-scale dynamics.

The mainstream complex network libraries currently available for network visualization, such as NetworkX (https://networkx.org/), iGraph (https://igraph.org/), and Gephi (https://gephi.org/), offer a range of capabilities for visualizing complex networks. These tools provide various visualization methods. At the microscopic scale, classical layout methods available in libraries such as NetworkX and iGraph, including the spring layout method,[15] the Kamada–Kawai (KK) method,[16] and the Fruchterman–Reingold (FR) method,[17] can reduce node and edge overlap, providing researchers with a clearer view of node-level details in the

†Corresponding author. E-mail: jingxiao0629@foxmail.com

© 2025 Chinese Physical Society and IOP Publishing Ltd. All rights, including for text and data mining, AI training, and similar technologies, are reserved.

network. At the mesoscopic scale, the tools provided by NetworkX and iGraph can effectively visualize community structures,[18–21] offering a clearer depiction of connections between network communities and enhancing understanding of the local structure of the network. At the macroscopic scale, widely-used libraries such as NetworkX and iGraph serve as fundamental tools for visualizing large-scale networks, focusing on enhancing visualization methods while preserving global properties, thereby enabling researchers to gain a global view of the network.[22]

Although there have been a large number of studies proposing methods for network visualization at different scales, the following challenges are still faced in the field of network visualization from micro to macro. At the microscopic scale, existing visualization methods are ineffective in placing components with similar edge properties in close proximity to each other. For example, in signed networks with labeled edges, the inability to effectively place nodes connected by edges with the same attributes in close proximity to each other prevents researchers from intuitively interpreting the relationships and structure of the network. At the mesoscale, existing visualization methods are insufficient for accurately representing communities, particularly in the case of nodes with fuzzy affiliations between overlapping communities. These methods often fail to capture the varying degrees of membership of nodes in overlapping communities, limiting researchers' ability to analyze node affiliations and interpret inter-community relationships. Furthermore, the lack of fixed node positions complicates visual comparisons between different community divisions, further hindering insights into the network's mesoscale structure. At the macroscale, existing approaches often overlook the distribution of different node types and the size of associated clusters when visualizing heterogeneous networks with diverse node types. For instance, current techniques struggle to effectively represent cluster sizes and node degree distributions in social robot networks comprising both human and robotic nodes. This limitation can make it difficult for researchers to accurately analyze and interpret the interactions and structures within such networks. Overall, these limitations restrict researchers from effectively capturing the dynamics and interactions among various types of nodes. Consequently, the development of user-friendly visualization libraries has become both necessary and urgent to address these challenges.

To address the aforementioned challenges, we propose the SFFSlib, an intuitive network visualization solution designed to enhance the layout visualization across scales, from the micro to the macro level. The key contributions of this library are as follows.

At the microscale, the SFFSlib incorporates visualization methods designed for networks with attributes. Specifically, it uses an elastically laid-out network visualization method that brings similarly connected nodes closer together while distancing those with dissimilar connections in a signed network. This approach enhances the representation of various edge attributes and highlights the dynamics of cooperation and competition among entities within the network.

At the mesoscale, the SFFSlib incorporates a variety of community visualization methods. Specifically, pie charts are employed for visualizing fuzzy overlapping communities, illustrating node memberships in overlapping communities and allowing researchers to explore fuzzy node affiliations in greater detail. Additionally, SFFSlib utilizes relative node fixation, which fixes specific nodes in place, enabling comparisons of changes in node affiliations across different community detection methods.

At the macroscale, the SFFSlib offers a specialized visualization method for heterogeneous networks. It distinguishes node types with unique labels and adjusts node sizes to represent their relative importance within the network. Clusters are abstracted as thumbnail nodes, emphasizing the network's broader structural patterns and interconnections. This combination of techniques enables a clearer representation of the network's overall topology, enhancing the ability to analyze its global properties and dynamics.

This paper aims to introduce the features and advantages of SFFSlib, providing researchers with a clear understanding of its functions and applications to support enhanced network visualization in related fields. SFFSlib is accessible at https://github.com/Jcowboy/network-visualization-library.git.

This paper is organized as follows. Section 2 introduces SFFSlib, providing details on its design approach, key features, function calls, and parameterization. Section 3 presents the evaluation metrics for visual layouts, along with the comparative layout algorithms and datasets used. Section 4 compares SFFSlib's performance against these metrics and algorithms, showcasing its visualization capabilities across different scales. Finally, Section 5 summarizes the findings and explores the potential for future development.

2. Overview of SFFSlib

The SFFSlib offers optimized layout algorithms for network visualization across three hierarchical scales: micro, meso, and macro. Figure 1(a) represents a signed network with positive and negative edges, Fig. 1(b) represents a network with a fuzzy overlapping community structure, and Fig. 1(c) represents a heterogeneous social robot network composed of human and social robot nodes. These different types of networks can all be visualized using SFFSlib. In Fig. 1(d), we first propose the attribute-driven edge proximity strategy (ADEPS) for microscale visualization. This algorithm addresses layout challenges in signed networks by clustering positive edges together and separating negative edges. This enhancement en-

ables researchers to more effectively explore and analyze microstructural patterns. In Fig. 1(e), for mesoscale visualization, we propose the fuzzy overlapping community visualization (FOCV) algorithm. FOCV optimizes the visualization of node membership degrees for overlapping nodes between overlapping communities. In addition, a node fixation strategy is proposed to assist researchers in gaining deeper insights into node-level observations under different community partitions. In Fig. 1(f), for macroscale visualization, we propose the degree cluster mapping strategy (DCMS). DCMS can dynamically adjust the size of nodes with large degrees and scale the visualization for nodes with different degrees of clustering, which is able to optimize the visual layout for the overall partitioning situation and dynamics in heterogeneous networks such as social robot networks. Table 1 below provides an overview of the key functions in our visualization algorithm.

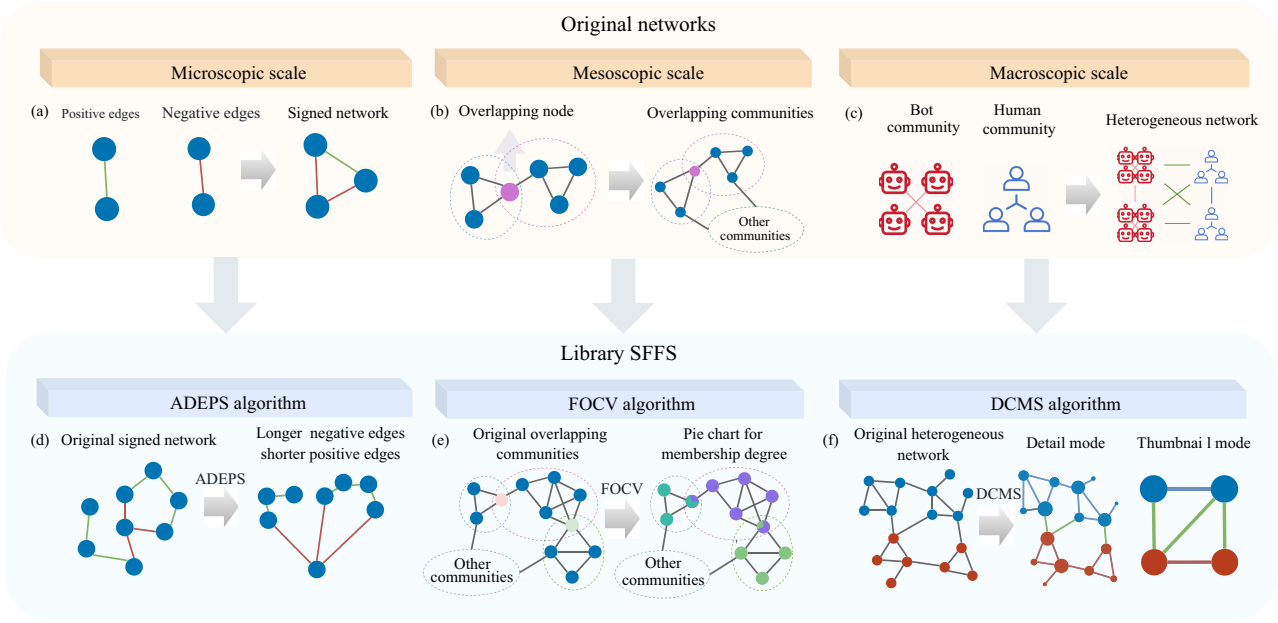


Fig. 1. Overview of the SFFSlib and its main features. (a) Signed networks with positive and negative edges. (b) Overlapping community structure with overlapping nodes. (c) Heterogeneous social robot network composed of human and social robot nodes. (d) Micro-scale: attribute-driven edge proximity strategy for signed network. (e) Meso-scale: fuzzy overlapping communities visualization algorithm. (f) Macro-scale: heterogeneous network visualization algorithm.

As shown in Table 1, main functions for visualization at different scales in the SFFSlib are described. This modular structure makes it easier for researchers to perform visualization tasks with only an understanding of the required input data format and function parameters. Many of the functions provide configurable hyperparameters with default values for common use cases. This design minimizes the complexity of function calls in typical cases, simplifies the calling process, and improves user-friendliness. In this section, we provide a detailed explanation of the key steps of important methods in SFFSlib through several code listings. These examples illustrate how visualization methods at various scales are implemented in practice, with a particular focus on the signature visualization algorithms in SFFSlib.

The parameter values presented in Table 1 are explained as follows. The *filepath* is a parameter for importing network data as a text file. The *options* is a dictionary that includes design specifications for font, font size, and font color. The B represents the membership degree of each node to different communities in the network and is a two-dimensional list with the same number of columns in each row. Each row represents a node, and each column within a row indicates the node's affiliation to each community, with values being of the float type, for example. The *community_list* denotes the community partitioning within the network and is a two-dimensional list where the number of columns in each row may vary. Each row represents a community containing certain nodes. The *colors* is a list used to set the color for each community. The *tag_file* and the *group_tag_file* are text files for robot label data and community label data, respectively.

The system architecture diagram of SFFSlib is shown in Fig. 2. In the context of microscopic visualization, the ADEPS algorithm is primarily utilized for the visualization of signed networks. The main functions implemented within ADEPS include *Init_Position* for calculating the initial positions of nodes, *Get_Coefficient* for computing the network structure parameters, *Get_NodeDegree* for computing the node degrees, and *Get_Weighted* for facilitating the force-directed layout of symbolic networks. For mesoscopic visualization, the FOCV algorithm is used for fuzzy overlapping community visualization and the NFS algorithm is used for fixed node visualization. Both FOCV and NFS algorithms contain *Get_Departition* for obtaining association divisions and

Get_Local_Centrality for calculating the semi-local centrality of nodes. But the FOCV algorithm employs *FCV_Pos_Update* and *FCV_Evolution* during the iterative process of updating node positions, which arranges overlapping nodes between the respective communities. In macroscopic visualization, the DCMS algorithm is used to visualize heterogeneous social bot networks. In the DCMS algorithm, the primary functions implemented include *Distinguish_Edges* for distinguishing edges with different attributes, *Generate_Center_Pos* for generating the coordinates of the center for each group, *Position_Update* for iteratively updating the node coordinates, and *Generate_Layout* for generating the overall layout.

Table 1. Description of the key functions in SFFSlib.

Name	Description	Related technology	Pros. and Cons.
Signed_Network_Visualization (filepath, options, node_size)	Visualize the signed network. The distance between positive edges is closer, thereby facilitating easier visual distinction.	ForceAtlas2 (2014)[24]	The layout process is rapid, but, it fails to visually differentiate between positive and negative edges in the signed network visualization.
Fuzzy_Community_Visualization (filepath, B)	Visualize the fuzzy overlapping communities. Overlapping node affiliation is represented by a pie chart.	Vehlow (2013)[25]	The visualization captures overlapping nodes' membership in fuzzy communities, but the layout does not easily reveal their community associations.
Overlapping_Community_ Visualization (filepath, community_list)	Visualize the overlapping communities. Node affiliation is equalized for each belonging community.	Crampes (2014)[28]	The visualization can encompass directed networks and overlapping communities; however, the arrangement of shared nodes between overlapping communities is not sufficiently accurate.
Visualization_of_Fixed_Nodes (filepath, community_list, colors, options)	Visualize different community division results with fixed node positions.	Wang (2023)[27]	The algorithm maintains visualization stability within online dynamic networks, but, it does not exert the same efficacy when community partitioning undergoes changes.
Social_Bot_Network_ Visualization (file_path, tag_file, group_tag_file, options)	Visualize the social bot network, returning both a detailed view and a thumbnail view.	Zhao (2017)[26]	The community structure is more clearly represented at the macroscopic scale, but, it fails to characterize the heterogeneous nodes and edges of a heterogeneous network.

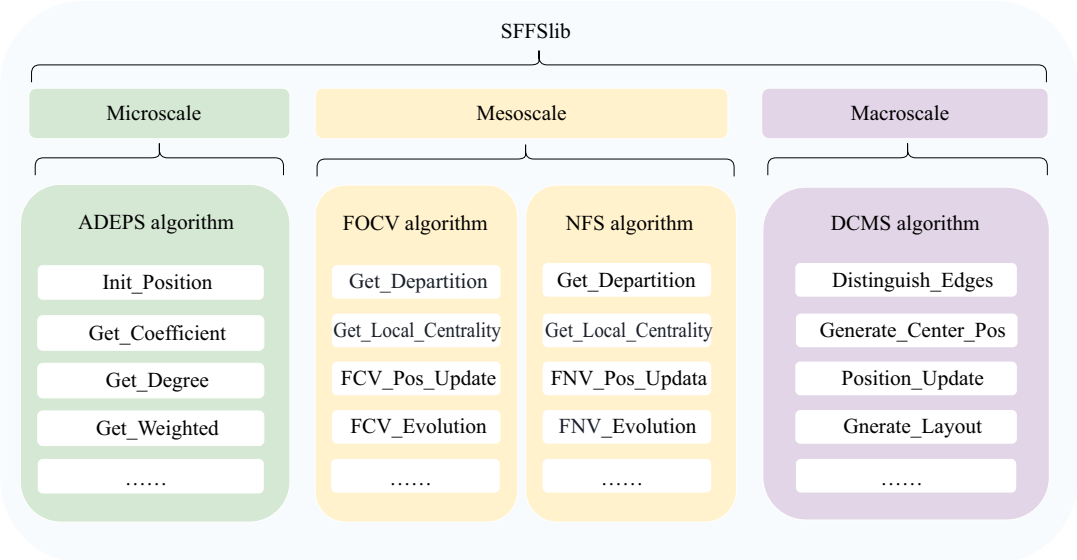


Fig. 2. System architecture diagram of SFFSlib. In the realm of microscopic visualization, the primary implementation is of the ADEPS algorithm. For mesoscopic visualization, the main implementations include the FOCV algorithm and the NFS algorithm, among others. In the domain of macroscopic visualization, the key implementation is of the DCMS algorithm.

2.1. Attribute driven edge proximity strategy

The real-world networks are characterized by multiple types of connections, with signed networks representing a prominent category.[23] In signed networks, edges are marked with symbols that indicate the positive or negative nature of connections between nodes. Traditional force-directed layout algorithms,[17,24,30] however, compute and visualize network structures by considering the absolute values of negative edges, thereby overlooking the distinctive properties of these edges. To address this limitation, we propose an attribute-driven edge proximity strategy, integrated within the SFFSlib alongside the attribute driven edge proximity strategy al-

gorithm. This strategy enhances edge localization by accounting for the attribute similarities between edges, ensuring that edges with comparable attributes are placed in closer proximity. Such a configuration improves the clarity of the network structure and the relationships between nodes. Listing 1 provides a concrete implementation of signed network visualization that leverages this proximity strategy.

In Listing 1, lines 6–10 are responsible for loading the network data and constructing the edge dictionary for later use. The ADEPS_coefficient function in line 16 calculates the parameters *g_cr* and *g_ca*, which are essential for the layout algorithm. These parameters are derived based on the distance between the furthest nodes in the network and the average distance between all nodes. Line 18 utilizes the ADEPS_weighted function, which enhances the traditional FR algorithm by applying an attribute-driven edge proximity strategy. In this strategy, negative edges exert repulsive forces, while positive edges apply attractive forces, effectively distinguishing the interactions of different types of edges. Lines 22–32 use Matplotlib and NetworkX libraries to visualize the network, with node positions determined by the layout algorithm. Specifically, in lines 27 and 30, positive edges are visualized in green, and negative edges are visualized in red, making it easier to distinguish between the two edge types in the network.

Listing 1 Example of attribute driven edge proximity strategy

```
1  import numpy as np
2  from matplotlib import pyplot as plt
3  import networkx as nx
4  import SFFSlib
5
6  G = ADEPS_read_file("RD_0.txt")
7  G_matrix = G.toarray()
8
9  Gn_RD = nx.read_weighted_edgelist("RD_0.txt", nodetype= int )
10 G1 = Gn_RD.to_undirected()
11
12 edgedict = SFFSlib.ADEPS_getdict(filepath, G1)
13 weight = SFFSlib.ADEPS_getweight(filepath)
14
15 seed = np.random.RandomState(0)
16 g_cr, g_ca = SFFSlib.ADEPS_coefficient(Gn_RD)
17
18 pos = SFFSlib.ADEPS_weighted(
19     G_matrix, dim=2, weight='weight', g_ca=g_ca, g_cr=g_cr, seed=seed
20 )
21
22 plt.figure(figsize=(10, 10))
23 plt.grid(True, which='both', axis='both')
24
25 nx.draw(G1, pos=pos, node_size=500, with_labels=True, **options)
26
27 for i in range (len (edgedict)):
28     edge_color ='forestgreen' if weight[i] > 0 else 'red'
29     nx.draw_networkx_edges(
30         G1, pos=pos, edgelist=edgedict[i + 1], edge_color=edge_color
31     )
32 plt.show()
```

By incorporating ADEPS into the layout algorithm, the method ensures that edges with similar attributes are located closer together, thereby improving the clarity of the network structure. Specifically, the distinction between positive and negative edges is preserved, with positive edges generating attraction and negative edges generating repulsion. This solution not only enhances the visualization of the signature network, but also presents the intrinsic relationships between nodes more accurately.

2.2. Fuzzy overlapping community visualization and node fixation strategy

Between the basic organizational structure and the global structure lies the mesoscale community structure, which reflects an associative feature of clusters of nodes sharing

similar properties.[31] The real-world networks have fuzzy overlapping communities belonging to multiple communities, and nodes have different degrees of affiliation with different communities.[29] Existing visualization methods tend to focus only on representing precise communities and ignore the visualization of nodes with overlapping attributes. In addition, existing community visualization algorithms typically generate random node positions with each computation, making it difficult to intuitively compare node-level community structures when switching between different community detection algorithms.[30,32–34]

The SFFSlib includes a visualization solution FOCV that effectively captures the fine-grained nature of nodes with overlapping attributes. By using color partitions to represent the degree of affiliation of overlapping nodes, this approach offers a clearer and more detailed view of fuzzy overlapping communities. Additionally, we propose a fixed-node community visualization function that ensures consistent node positions across different community detection results on the same network. This enables more intuitive comparisons of various community partitioning outcomes. The implementation code for FOCV using this strategy is presented in Listing 2, while Listing 3 provides the detailed implementation of fixed-node visualization.

Listing 2 Example ofthe fuzzy overlapping community visualization

```
1  import  numpy as np
2  import  graph_tool as gt
3  import  SFFSlib
4
5  filepath =  “karate.” txt
6  G= SFFSlib.read_file(filepath)
7  G_matrix = G.toarray()
8  B= Node_Membership_Matrix
9
10 partitionWithCenters = SFFSlib.get_center_points(G_matrix)
11 pos1= SFFSlib.evolution(B,G_matrix,partitionWithCenters,G, t=None, C=1,
12        g_cr=1.1, g_ca=0.03,iterations=100, threshold=1e-5)
13
14 draw_G = SFFSlib.getG(pos1,filepath)
15 pos = SFFSlib.getpos(pos1,draw_G)
16 vp = draw_G.new_vertex_property("vector<double>")
17 B = np.array(B)
18 size=[]
19
20 for  i  in  B:
21      size.append(np.count_nonzero(i!=0))
22 for  i  in  range (pos1.shape[0]):
23      vp[draw_G.vertex(i)]=B[i]
24      gt.graph_draw(draw_G,pos=pos,
25                    vertex_size=15*(B.shape[1]+size[i])/B.shape[1],
26                    vertex_text=draw_G.vertex_index,
27                    vertex_font_size=10,vertex_text_color='black',
28                    vertex_shape='pie',vertex_pie_fractions=vp,
29                    output='overlapping1.png',inline=True)
```

In Listing 2, we demonstrate the visualization of fuzzy overlapping communities. Lines 1–3 import all necessary modules. Lines 5–7 load the network data, while line 8 uses *Node_Membership_Matrix* to store the fuzzy community assignment results and membership degree information in a two-dimensional array of floating-point values. Line 10 calls the get_center_points function to locate the central nodes of each community. In line 11, the evolution function iteratively updates the repulsive and attractive forces between nodes, the repulsive forces between community centers, and the attractive forces between nodes and their community centers to refine the layout. The iterations parameter specifies the maximum number of iterations to prevent infinite looping, while the threshold parameter sets the convergence threshold — indicating that the layout has stabilized when the average movement distance of nodes falls below this value. Lines 22–29 primarily utilize Python's graph_tool library to visualize the network with the computed node positions.

In Listing 3, community detection results from different algorithms can be loaded for the same network in a two-

dimensional list format in line 8. After the node position iterations are computed in line 10, the community partition results loaded in line 8 are then applied in line 20. This approach ensures that changing the community detection results in line 8 does not alter node positions. Lines 16–17 set the node colors and font sizes, aligning node colors with the number of communities to distinguish them visually through different colors. Lines 19–22 utilize the NetworkX library for visualization. Finally, line 23 saves the resulting visualization image.

By incorporating the precise affiliations of overlapping nodes between communities into the calculation process and representing these affiliations degree with pie charts, the extent of node overlap between communities can be effectively visualized. This approach enhances the clarity of the overlap and provides a detailed view of the specific degree of affiliation of overlapping nodes. In the visualization of fixed-node communities, the positions of fixed nodes are maintained consistently across different community detection results. This technique enables visual comparisons of community structures, allowing researchers to easily observe the differences between results from various community detection methods.

Listing 3 Example ofthe node fixation strategy

```
1  import  SFFSlib
2  from matplotlib  import  pyplot as plt
3  import networkx as nx
4
5  filepath = “karate.” txt
6  G = SFFSlib.read_file(filepath)
7  G_matrix = G.toarray()
8  community_list = result_by_Community_Detection_Algorithm
9
10 pos = SFFSlib.evolution(G, t=None, C=1, g_cr=0.7, g_ca=0.8,
11                        iterations=110, threshold=1e-5)
12
13 Gn = nx.read_edgelist(filepath,nodetype= int )
14 G1 = Gn.to_undirected()
15
16 colors = [ '# eb8f90' ,  '# ffb471' , 'adbed2#' , '#12406a' ]
17 options = {' font_family' : ' serif ' , ' font_size': ' 8',  ' font_color' : ' #fffff' }
18
19 nx.draw(G1,pos,edge_color= "gray",with_labels=True, node_size = 260,**options)
20 for i in range (len (community_list)):
21     nx.draw_networkx_nodes(G1,pos,nodelist=community_list[i], node_size=300,
22                            node_color=colors[i],label=True)
23 plt.savefig( " Community_visualization_with_fixed_node_positions . svg" )
```

2.3. Degree cluster mapping strategy

In networks comprising heterogeneous nodes, such as those containing both social bots and real human users, social bots frequently display behaviors distinct from human interactions.[35] These include automated, repetitive, or coordinated actions that can significantly alter the network's true structure and interactions. Effectively visualizing the distribution of different node types and the size of their associated clusters is essential for gaining a clearer understanding of the network dynamics. However, traditional visualization methods often fail to accurately depict the relationships between bots and real users, limiting the detection of anomalous behaviors and inhibiting deeper exploration of network dynamics.[30,32–34]

The SFFSlib provides two distinct visualization modes: detailed and thumbnail graphs. In detailed graphs, nodes with higher degrees are appropriately enlarged, enhancing their visibility within the network. In contrast, thumbnail graphs aggregate nodes that share common community attributes, facilitating a clearer view of network structures at a broader level. By clearly distinguishing between node types, these visualizations aid in the identification of clusters or patterns related to bot activity, thereby offering a more comprehensive understanding of the underlying network behavior.

Listing 4 illustrates the process of degree cluster mapping strategy. First, three relevant data files must be imported: a network data file containing edge information (line 3), a label file indicating whether nodes are human or bot nodes (line 4), and a label file identifying the community to which each node belongs (line 5). The code in lines 8–9 sets the colors for the nodes and edges, where the first two parameters of node_colors define the colors for human and bot nodes, re-

spectively. The three parameters for edge colors specify the color settings for connections between human nodes, between human and bot nodes, and between bot nodes. The function show_robot_group in lines 18–19 generates the corresponding detailed and thumbnail views based on the input network data. The abstract concepts of degree distribution and degree of aggregation of nodes in networks can be effectively visualized by using node sizes in detailed graphs and representing communities as nodes in thumbnail views. Additionally, differentiating the colors of the connecting edges between human and robot nodes enhances the visualization of the various types of interactions within the social robot network.

Listing 4 Example ofthe degree cluster mapping strategy

```
1  import SFFSlib
2
3  file_path = 'football.txt'
4  tag_file = 'football_tag.txt'
5  group_tag_file = 'football_grouptag.txt'
6  G = SFFSlib.read_file(file_path)
7
8  options = {'node_colors': ['#60afff', '#ff6073'],
9             'edge_colors' : ['#146bc2', '#bf1010', '#6bbe71']}
10
11 group = SFFSlib.group_tag_process(tag_file,group_tag_file)
12 center_pos, group_radius = SFFSlib.generate_center_pos(group, len(group) // 2)
13 group_dict = SFFSlib.generate_group_dict(center_pos, group)
14
15 pos = SFFSlib.generate_layout(G_matrix, group_dict, t=None, C=1,
16       g_cr=1.2, g_ca=1.8,iterations=100, threshold=1e-5)
17
18 SFFSlib.show_robot_group(file_path, G, tag=tag,
19       group_tag=group_tag, direction=False,save=True, show=True, **options)
```

3. Evaluation metrics, comparative algorithms, and data

3.1. Evaluation metrics

We utilize four evaluation metrics[36,37] to assess the distribution effect of nodes across various layout methods, including node distribution variance (NDV), edge length deviation (ELD), node distribution deviation (NDD), and signed network layout balance index (SNLBI). A detailed explanation of these evaluation metrics is provided below.

Node distribution variance (NDV): This metric evaluates whether the layout of the network makes sufficient use of the available space. First, the entire space is divided into n equal rectangular areas, with the number of nodes in the i-th area denoted as x_i. The average number of nodes across all areas is denoted as $\bar{x}$, and the node distribution variance is denoted as s. The formula is given as

$$s = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2. \tag{1}$$

Edge length deviation (ELD): This metric reflects the uniformity of edge lengths by comparing the difference between the shortest and longest edge lengths against the average edge length. The length of an edge is denoted as l, and the edge length deviation is denoted as ELD. The formula is given as

$$\mathrm{ELD} = \frac{(\max(l) - \mathrm{mean}(l)) + (\mathrm{mean}(l) - \min(l))}{\mathrm{mean}(l)\cdot \mathrm{num}(l)}, \tag{2}$$

where $\max(l)$ and $\min(l)$ represent the maximum and minimum edge lengths respectively, $\mathrm{mean}(l)$ represents the average of the edge lengths, and $\mathrm{num}(l)$ represents the number of edges. In the signed network visualization experiments conducted in Subsection 4.1, the dataset utilized in this study distinguishes between positive and negative edges. Accordingly, the ELD is analyzed in two categories: positive edge length deviation (PELD), calculated exclusively for positive edges, and negative edge length deviation (NELD), calculated exclusively for negative edges.

Node distribution deviation (NDD): This metric reflects the effectiveness of the repulsive forces applied in the algorithm. Let the node distribution deviation be denoted as NDD. The formula is given as

$$\mathrm{NDD} = \frac{|d - \min(d)|}{A}, \tag{3}$$

where d denote the optimal distribution distance, and $\min(d)$ denotes the minimum node distance. The optimal distribution distance is defined as the display area of the graph divided by the number of nodes, shown as

$$d = \frac{A}{N}. \tag{4}$$

Symbolic network layout balance index (SNLBI): This metric aims to quantify the balance and clarity of positive and negative edges within the layout of a symbolic network. The formula is given as

$$\mathrm{SNLBI} = \alpha \cdot \mathrm{EC} + \beta \cdot \mathrm{VDN} \cdot n, \tag{5}$$

where α and β are weighting coefficients derived from the correlation analysis between EC and VDN. EC represents the number of edge crossings between positive and negative edges in the layout, and VDN denotes the variance in the number of nodes per unit area. n is the total number of nodes. The formula of EC is given by

$$\mathrm{EC} = \frac{c}{c_{\max}}, \quad (c_{\max} > 0), \tag{6}$$

where $c_{\max}$ represents the maximum possible number of edge crossings in the network, and its calculation formula is

$$c_{\max} = \frac{m(m-1)}{2} - \frac{1}{2}\sum_{j=1}^{n}(d(v_j)\cdot(d(v_j)-1)). \tag{7}$$

Here, m represents the total number of edges in the network, n denotes the total number of nodes, and $d()$ is a function that calculates the degree of a node.

3.2. Datasets

The datasets used in this study span various domains, including social interactions, community structures, and other dynamic systems. The following datasets are employed in the analysis.

Wiring:[38,39] A network of social relationships among 14 shop floor workers at the Hawthorne plant of the Western Electric Company. Positive values represent friendship relationships among workers, while negative values denote hostile relationships. The network consists of 14 nodes and 32 edges, including 13 positive and 19 negative edges.

Dolphins:[40] A social network representing 62 dolphins living in the New Zealand region, where nodes represent different dolphins and edges represent their contact relationships. The network contains 62 nodes and 152 edges and can be divided into four communities according to the closeness of the connections.

Karate:[41,42] Collected by Wayne Zachary in 1977 from the members of a university karate club. Each node represents a club member, while each edge signifies a tie between two members. The network comprises 34 nodes and 78 edges, and can be approximately partitioned into four communities based on the connectivity between nodes.

Football:[43] A network of American football games between Division IA colleges during the regular season Fall 2000, where nodes represent football teams and edges indicate that a match has been played between football teams. The network consists of 115 nodes and 616 edges, and can be partitioned into six communities.

Tribes:[44] A dataset representing the network of relationships among highland mountain tribes, initially collected by Reed in 1954 from the Gama tribe group in eastern Papua New Guinea . The network comprises 16 nodes and 58 edges, where the nodes represent tribes and the edges denote intertribal connections. The network comprises 29 positive and 29 negative edges. Positive edges represent friendly relations, while negative edges denote hostile relations.

Cloister:[45] A relationship network dataset titled 'Crisis in the Cloister', published on the World Wide Web. This dataset captures social interactions and sociometric rankings among a group of monks, collected by Sampson during his residency as an experimenter. The network consists of 18 nodes and 125 edges, where the nodes represent monks and the edges denote relationships among them. Positive edges indicate friendships, while negative edges signify hostility.

Stranke94:[46] It illustrates the relationships between political parties in the Slovenian parliament, comprising 10 nodes and 45 edges. The nodes represent different political parties, while the edges denote the differences and similarities between them. Positive edges indicate similarity, negative edges signify dissimilarity, and the absence of edges reflects indifference in the relationship between the parties.

3.3. Baseline algorithm

This study compares several layout algorithms of the same type to demonstrate the superiority of the layout algorithm implemented in our SFFSlib. The following provides a detailed description of the baseline algorithms discussed in this paper.

FR:[17] FR is a layout algorithm that computes the attractive and repulsive forces between nodes in the network and combines them with the optimal distance to achieve a superior layout. The optimal distance between nodes is defined as

$$k = C\sqrt{S/N}, \tag{8}$$

where S is the canvas area, N is the number of nodes, and C is found experimentally. Using f_{a} for the attractive force and f_{r} for the repulsive force, the formulas for the attractive and repulsive forces are expressed as

$$f_{\mathrm{a}} = \frac{d^2}{k}, \quad f_{\mathrm{r}} = \frac{-k^2}{d}, \tag{9}$$

where d represents the Euclidean distance between the two nodes.

GRA:[30] This algorithm has a repulsive force between every two nodes and an attractive force between two nodes that are connected or in the same community. The repulsive force formulas for the GRA method are defined as

$$F_{\mathrm{r}}(p_i, p_j) = CK^2/d(p_i, p_j), \tag{10}$$

where C adjusts the relative strength of the repulsive and attractive forces, and d represents the visualization distance between nodes i and j. K is the optimal distance, defined as $\sqrt{S/N}$, where S is the area of the canvas and N is the number of vertices. The attractive force formulas for the GRA method are defined as

$$F_{\mathrm{a}}(p_i, p_j) = \lambda \cdot K/d(p_i, p_j), \tag{11}$$

where λ denotes the weight of the edge between nodes i and j.

CGDA:[33] This algorithm builds upon the force-directed layout algorithm by adding community attraction to each node and incorporating the k-means algorithm, enabling nodes within the same community to converge toward the center of the community. This algorithm does not require prior classification of nodes and can perform node clustering simultaneously during the layout process. Community attractive force is defined as

$$f_{\mathrm{g}} = gM[v]d_v, \quad d_v = \min(d_1, d_2, \ldots, d_v), \tag{12}$$

where $M[v]$ denotes the mass of node v, which is determined by the importance of the node. g takes values between 0 and 2, with larger values corresponding to more nodes. d_k represents the Euclidean distance from node v to the center of association C_k, and the minimum distance is taken as the value of d_v. Community attractive force is defined as

$$f_{\mathrm{r}}(u_i, u_j) = g_c M(c_i) M(c_j)/d_u, \tag{13}$$

where d_u denotes the Euclidean distance between u_i and u_j, $M(C_i)$ denotes the quality of association C_i, and the parameter g_c needs to be adjusted according to the scale of the actual network data and the size of the canvas.

The Zhou algorithm:[34] The Zhou algorithm incorporates degree centrality-based community repulsion and gravity, improving upon the traditional FR layout method. The formula for degree centrality is given as

$$\mathrm{DC}_i = \frac{k_{\mathrm{i}}}{N-1}, \tag{14}$$

where k_i is the degree of node i, and N is the total number of nodes in the network.

ForceAtlas2:[24] This is a force-directed layout method. It simulates physical systems to spatialize networks. Nodes repel each other like charged particles, while edges attract their nodes like springs . ForceAtlas2 incorporates the node degree into the calculation of the repulsive force, thereby mitigating the visual confusion caused by a small number of highly connected node clusters.

The Zhu algorithm:[32] The algorithm begins by treating each community as a node and constructing a network where communities serve as the nodes. Next, the community network is laid out using physical analogies to determine its structure. The coordinates of the center point for each community area are then calculated based on the network layout. Subsequently, the community areas are populated with nodes corresponding to the members of each community. Finally, the positions of these nodes are adjusted to prevent overlap, ensuring a clear and visually coherent layout.

4. Performance evaluation of SFFSlib

In this section, we demonstrate the advantages of our proposed algorithm in the SFFSlib, from two perspectives. First, we illustrate the intuitive visual benefits of the SFFSlib across various visualization scenarios by presenting a range of example visualizations. Second, we quantitatively compare the aesthetic quality of the methods implemented in SFFSlib with those of other existing visualization techniques, utilizing standard aesthetic metrics to evaluate the performance of ADEPS and FOCV. To illustrate the visualization effects of SFFSlib from the micro to macro visualization, we employ signed network visualization for micro-scale representation, fuzzy overlapping community visualization and fixed node community visualization for meso-scale representation, and social bot network visualization for macro-scale representation.

4.1. Performance of attribute driven edge proximity strategy

In this subsection, we conducted experiments to demonstrate the effectiveness and superiority of the symbolic network visualization methods in SFFSlib. First, we performed visualization experiments on symbolic networks containing both positive and negative edges and introduced other baseline algorithms for comparative analysis. Subsequently, we employed commonly used visualization metrics along with additional baseline algorithms for quantitative comparisons. The visualization experiments utilized the Wring and Stranke94 datasets.

The experimental procedure is as follows. First, we applied the GRA algorithm to visualize the Wring and Stranke94 network datasets. After calculating the positions of nodes within the network, positive edges were rendered in green and negative edges in red during visualization. Then, we repeated the above steps using the ADEPS algorithms from SFFSlib to facilitate direct visual comparisons.

In Fig. 3, the red edges represent negative relationships in the symbolic network, while the green edges represent positive relationships. Figures 3(a) and 3(c) illustrate the widely used FR algorithm, which struggles to clearly distinguish between positive and negative edges in the symbolic network. Without the use of color differentiation, it is challenging to identify the distinction between positive and negative edges.

In contrast, the algorithm presented in the SFFSlib, shown in Figs. 3(b) and 3(d), effectively positions nodes connected by positive edges closer together, while nodes connected by negative edges are placed further apart, thereby enhancing the visibility of the distinction between positive and negative edges in most cases.

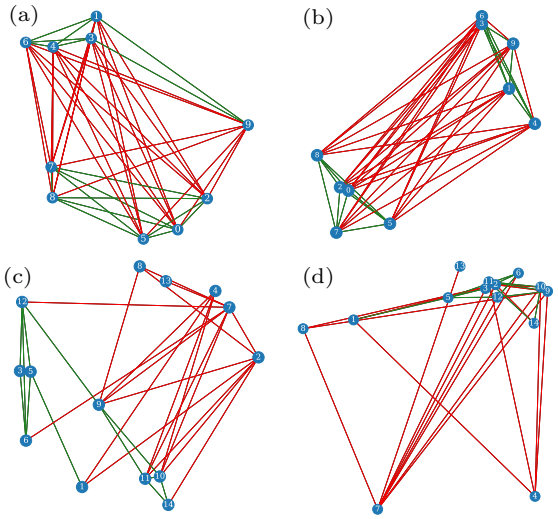


Fig. 3. The visualization results of the signed network using the GRA algorithm and the ADEPS algorithms implemented in this library X. (a) Visualization with the GRA algorithm on the Stranke94 network dataset, where positive and negative edges are not distinguished. (b) Visualization on the Stranke94 network dataset using the ADEPS algorithm in library X, where the positive edges are universally shortened and the negative edges are universally lengthened. (c) Visualization with the GRA algorithm on the Wring network dataset, where positive and negative edges are not distinguished. (d) Visualization on the Wring network dataset using the ADEPS algorithm in library X, where the positive edges are universally shortened and the negative edges are universally lengthened.

We evaluated the performance and advantages of the symbol network visualization algorithm, ADEPS, within our SFFSlib. We included the FR, ForceAtlas2, and GRA algorithms as comparison algorithms. Edge length deviation and the SNLBI index were selected as evaluation metrics, with the edge length deviation further subdivided into positive and negative edge length deviations to align with the symbol network's application context. Besides the Wiring network and the Stranke94 network, additional datasets were used, including the Tribes network and the Cloister network.

As shown in Table 2, the signed network layout algorithm ADEPS in our SFFSlib demonstrates a smaller balance index compared to other algorithms. This indicates that the ADEPS algorithm effectively reduces edge crossings and alleviates local topological disorder. Additionally, the ADEPS algorithm exhibits lower positive and negative edge length deviations, suggesting that it enables a more uniform distribution of edges within the context of symbol networks. This indicates that the ADEPS algorithm in SFFSlib not only distinguishes positive and negative edges in a symbolic network in an intuitive and efficient manner, but also produces a more aesthetically pleasing layout overall.

Table 1. The experimental results for signed network visualization methods.

Networks	Matrics	FR	ForceAtlas2	GRA	ADEPS
Wiring	NELD	0.36	0.4	0.52	0.32
	PELD				0.31
	SNLBI	23.2	18.9	16.4	11.6
Tribes	NELD	0.32	0.58	0.5	0.33
	PELD				0.2
	SNLBI	37.4	41.3	30.1	30
Cloister	NELD	0.28	0.46	0.53	0.42
	PELD				0.35
	SNLBI	35.7	25.2	22.8	19.1
Stranke94	NELD	0.36	0.5	0.7	0.21
	PELD				0.45
	SNLBI	35.4	34.7	17.2	16.3

4.2. Performance of networks with community structures

In fuzzy overlapping communities, overlapping nodes belong to multiple communities with varying degrees of membership. To demonstrate the effectiveness of the overlapping community visualization method in SFFSlib, we performed comparative visualizations using the Zhu algorithm and the CGDA algorithm on the Dolphin network. The process is as follows. Community detection was first performed to obtain the community division results. Then, by using the Zhu algorithm, the CGDA algorithm, and the FOCV algorithm from SFFSlib, the positional information of each node in the network was computed based on the community structure. Finally, the network was visualized to illustrate the results.

The advantages of the fuzzy overlapping community visualization algorithm FOCV in SFFSlib are readily apparent in Fig. 4. The Zhu layout algorithm can only handle crisp communities, resulting in clusters that are densely packed, making it challenging to differentiate communities based on their spatial separation. While the CGDA algorithm distinctly separates different communities, it is unable to depict overlapping nodes between them. In contrast, the fuzzy community visualization algorithm in SFFSlib leverages pie charts to visually represent the degree of community membership for overlapping nodes. The proportion of a specific community color in a pie chart indicates the level of association that the overlapping node has with that community.

As shown in Fig. 4, nodes 8, 19, 20, 37, 39, 40, and 50 are identified as overlapping nodes between communities, situated at community boundaries and belonging to two or more communities. Nodes 8, 20, 37, and 50 are shared among the three communities represented by red, yellow, and green colors. Notably, node 20 shows a larger green segment than the other three nodes, indicating a higher degree of association with the green community compared to the other overlapping nodes.

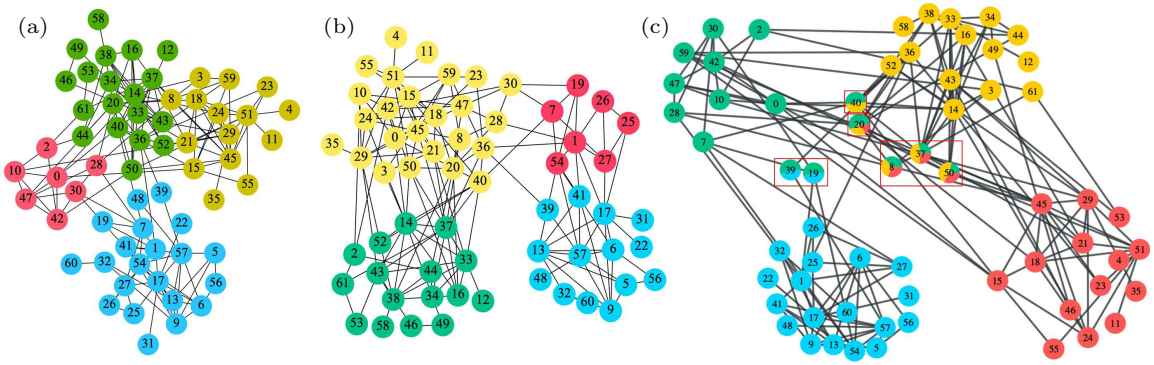


Fig. 4. Visualization using different community visualization algorithms for the Dolphin network. (a) The Zhu algorithm. (b) The CGDA algorithm. (c) The FOCV algorithm within SFFSlib.

We evaluated the performance of the FOCV included in SFFSlib. The experiments introduced four comparison layout algorithms: the FR algorithm, the Zhu layout algorithm, the CGDA algorithm, and the Zhou algorithm. Using point distribution variance, node distribution deviation, and edge length deviation as evaluation metrics, we conducted tests on the Dolphins and Football networks. The final experimental results are shown in Tables 3 and 4.

From the experimental results presented in Tables 3 and 4, it is evident that the overlapping community visualization algorithm within SFFSlib exhibits lower point distribution variance and node distribution deviation compared to other algorithms. This indicates that our algorithm effectively minimizes node disarray and reduces the clutter within localized topological structures, resulting in a more evenly distributed layout with reduced crowding.

Table 3. The experimental results for the Dolphins network.

Metrics	FR	The Zhu algorithm	CGDA	The Zhou algorithm	SFFSlib
Node distribution variance	0.3882	0.4885	0.4882	0.3326	0.3050
Node distribution deviation	0.1619	0.1617	0.1608	0.1602	0.1403
Edge length deviation	0.1752	0.1490	0.1551	0.1460	0.2003

Table 4. The experimental results for the Football network.

Metrics	FR	The Zhu algorithm	CGDA	The Zhou algorithm	SFFSlib
Node distribution variance	0.7074	0.6586	0.6771	0.6001	0.4384
Node distribution deviation	0.8689	0.8754	0.8681	0.8675	0.1565
Edge length deviation	0.3690	0.3751	0.3461	0.3216	0.5507

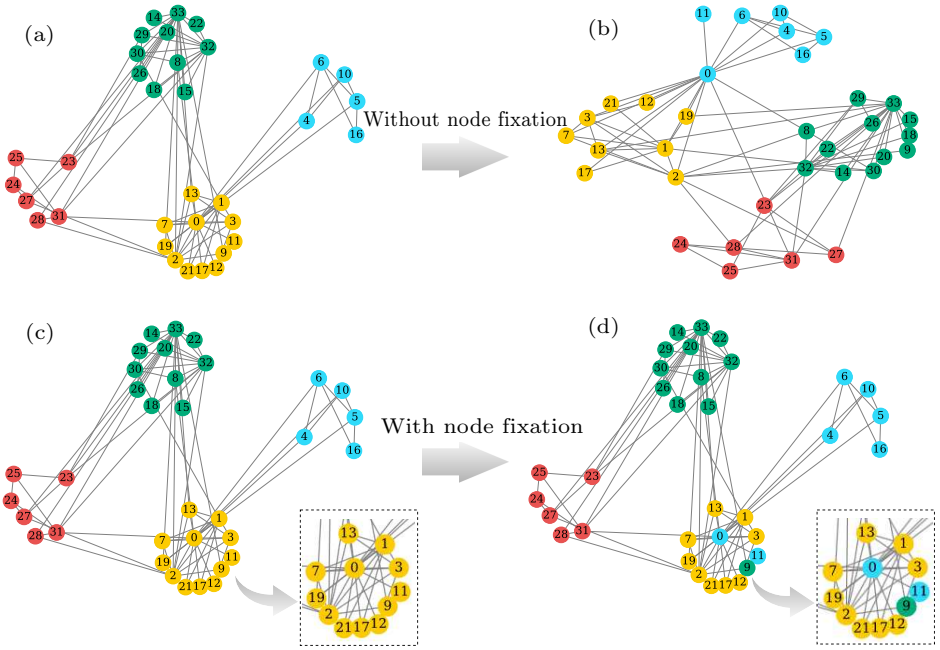


Fig. 5. Fixed versus unfixed nodes in the context of different community detection algorithms. (a) Community detection and visualization using hierarchical multilevel clustering algorithm with unfixed nodes. (b) Community detection and visualization using the GNM algorithm with unfixed nodes. (c) Community detection and visualization using hierarchical multilevel clustering algorithm with fixed nodes. (d) Community detection and visualization using the GNM algorithm with fixed nodes.

Due to the way the overlapping community visualization algorithm in SFFSlib positions overlapping nodes between their associated communities, it demonstrates a relatively higher edge length deviation. However, since overlapping nodes constitute a small portion of the network's total nodes, this impact on overall visual clarity remains minimal, and the increase in edge length deviation is within an acceptable range.

Using the karate network, we visualized community structures produced by the hierarchical multilevel clustering algorithm and the GNM algorithm, both with fixed-node positions and without fixed-node positions. The results are shown in the Fig. 5. In Fig. 5, the top row illustrates the visualizations for two community detection algorithms without fixed node positions, while the bottom row shows the visualizations with fixed node positions. For instance, in the hierarchical multilevel clustering algorithm result on the left side of the second row, nodes with IDs 0, 9, and 11 belong to the yellow community. When the community detection algorithm is changed to the GNM algorithm (right side of the second row), it is immediately evident that nodes 0, 9, and 11 no longer belong to the original yellow community. However, this distinction is more challenging to discern in the top row. Therefore, after fixing the node positions, it becomes easier to observe the differences in community division resulting from the two different community detection algorithms. Conversely, without fixed node positions, it is difficult to discern the similarities and differences between the two community detection outcomes.

4.3. Performance of degree cluster mapping strategy

Heterogeneous networks consist of nodes with different attributes. A social bot network, as a type of heterogeneous network, includes human nodes and bot nodes. To demonstrate the effectiveness of the visualization algorithm DCMS in SFFSlib for social bot networks, we conducted experiments using simulated social networks and compared the results with the Zhou algorithm. The experiment was conducted on the football network, where a random function was used to label some nodes as human nodes and others as bot nodes, thereby constructing the social bot network. The experimental procedure is as follows. First, we construct the social bot network and perform community detection. Then, compute the positions of the nodes to obtain the positional information within the network. Finally, Visualize the nodes and edges to illustrate the network structure.

The visualization results of the social bot network simulated with the Football network are shown in Figs. 6 and 7. In Figs. 6(a) and 6(c), the results of the visualization of the social robot network using the Zhou algorithm for layout are shown. Fig. 6(b) presents the detailed graph visualization produced by SFFSlib, while Fig. 5(d) displays the thumbnail view of SFFSlib.

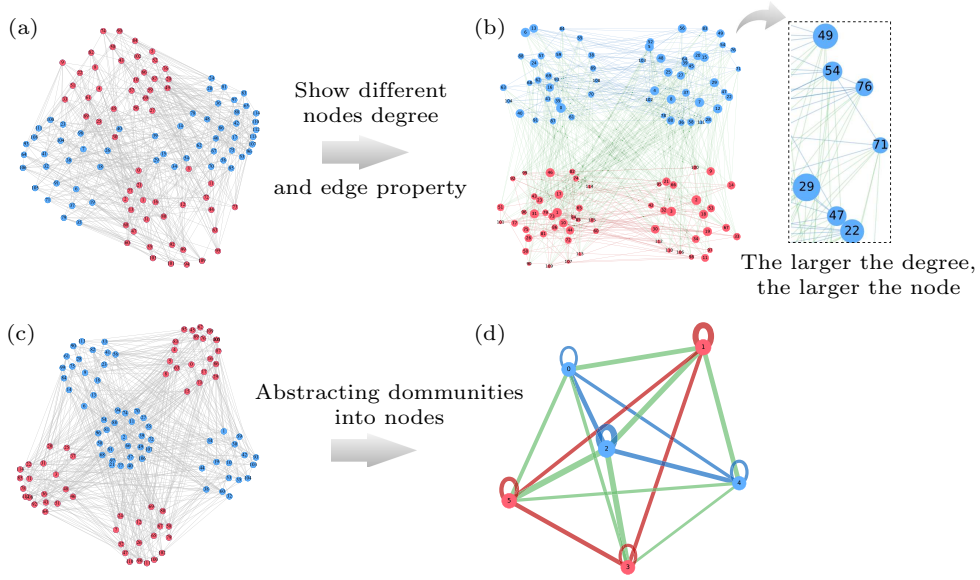


Fig. 6. The visualization results of the social bot network using the Zhou algorithm and the algorithm implemented in this SFFSlib. (a) Visualization with the Zhou algorithm, when the number of communities simulated is 4. (b) Visualization with the algorithm in SFFSlib, when the number of communities simulated is 4. The larger the degree, the larger the area shown when nodes are visualized. (c) Visualization with the the Zhou algorithm, when the number of communities simulated is 6. (d) Visualization of abstracting nodes into communities using the algorithm in SFFSlib.

In the detailed graph shown in Fig. 6(b), blue nodes represent human users, red nodes indicate robot users, blue edges signify communications among human users, red edges represent interactions among robots, and green edges denote communications between human and robot users. Node sizes are proportional to their degrees; larger nodes correspond to higher degrees. In the thumbnail depicted in Fig. 6(d), each node represents a community, where blue nodes correspond to human communities and red nodes to robot communities. The edge colors in the thumbnail are consistent with those in the detailed graph, and

edge thickness reflects the number of connections between the corresponding communities. Figure 7 presents simulations with varying numbers of communities on the football network. The results demonstrate that, regardless of the number of communities, the method implemented in SFFSlib consistently optimizes the canvas layout, effectively highlighting the connections between communities.

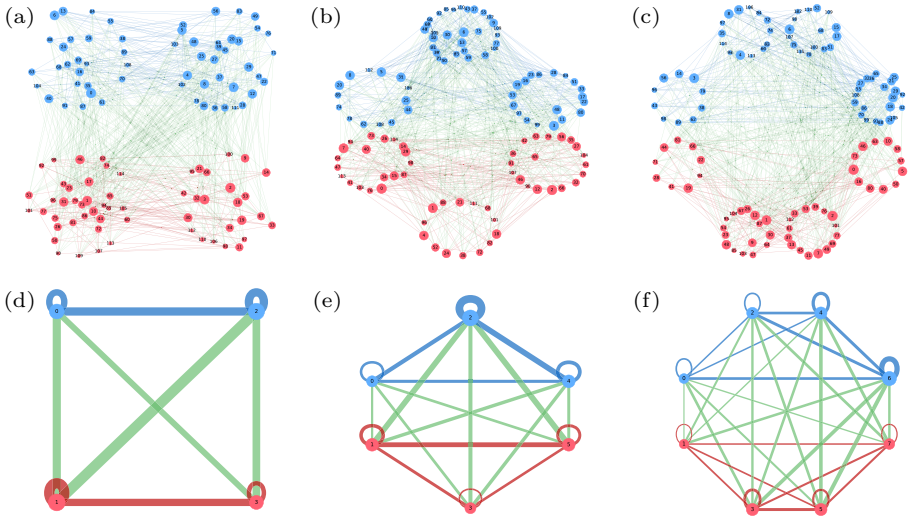


Fig. 7. Detailed visualization of the social bot network and the thumbnail view. (a) Detailed graph of the community partitioned into four groups. (b) Detailed graph of the community partitioned into six groups. (c) Detailed graph of the community partitioned into eight groups. (d) Thumbnail graph of the community partitioned into four groups. (e) Thumbnail graph of the community partitioned into six groups. (f) Thumbnail graph of the community partitioned into eight groups.

5. Conclusion

In this work, we developed SFFSlib, a Python framework built upon the open-source packages NetworkX, Graph Tool, NumPy, and Matplotlib, with the primary goal of providing a comprehensive toolset for visualizing complex networks across multiple scales, from micro to macro. At the micro scale, the SFFSib employs a visualization strategy that improves the representation of signed networks by elongating negative edges and shortening positive edges, thereby facilitating the distinction between edges with different properties. At the meso scale, SFFSlib integrates a visualization approach that improves the display of node membership degree in overlapping regions of fuzzy overlapping communities and facilitates the comparison of different community partitions through fixed-node community visualizations. At the macro scale, SFFSlib employs a visualization algorithm for heterogeneous networks, which better highlights the relative importance of nodes within the overall network structure, and provides thumbnail representations to assist researchers in understanding the significance of individual communities in the broader context of the network. Experimental evaluations on empirical networks demonstrate that the algorithm implemented in SFFSlib significantly improves the effectiveness of various network layout forms. Furthermore, SFFSlib outperforms alternative visualization methods based on several key performance metrics. Thus, SFFSlib provides enhanced capabilities for researchers to examine network structures across different scales.

Although the current version of SFFSlib demonstrates effective performance in specific domains, it should be emphasized that certain technical limitations persist, requiring further systematic investigation. First, at the microscale , the SFFSlib is solely focused on optimizing the visualization of signed networks with positive and negative edges, excluding strategies for networks with additional symbolic representations such as neutral edges. Second, at the mesoscale, SFFSlib is primarily designed to optimize the visualization of fuzzy overlapping communities, but it does not adequately address the visualization of networks featuring hierarchical or nested community structures, thereby constraining its capability to depict intricate multi-level community relationships. Finally, at the macroscale, the SFFSlib prioritizes the mapping of degree clusters within heterogeneous networks, but it does not comprehensively optimize the visualization of multi-layered networks, in which various layers denote different relationship or interaction types, thereby restricting its efficacy in encapsulating and illustrating inter-layer dependencies and interactions.

Program availability

The source codes are openly available in Science Data Bank at https://doi.org/10.57760/sciencedb.j00113.00212.

Acknowledgements

Project supported by the National Natural Science Foundation of China (Grant Nos. 61773091 and 62476045),

the LiaoNing Revitalization Talents Program (Grant No. XLYC1807106), and the Program for the Outstanding Innovative Teams of Higher Learning Institutions of Liaoning (Grant No. LR2016070).

References

[1] Mancoridis S, Mitchell B S and Rorres C 1998 *Proceedings of the 6th International Workshop on Program Comprehension*, June 26, 1998, Ischia, Italy, p. 45

[2] Gao M, Li Z and Li R 2023 *Patterns* **4** 100839

[3] Ran Y J, Xu X K and Jia T 2024 *PNAS Nexus* **3** 113

[4] Mursa B E M and Andreica A 2024 *Swarm and Evolutionary Computation* **86** 101526

[5] Yu Z H, Lu S, Wang D and Li Z W 2021 *Information Sciences* **580** 857

[6] Hanson K R and Theis N 2024 *Sociological Methodology* **54** 142

[7] Bahadorian M, Alimohammadi H and Mozaffari T 2019 *Scientific Reports* **9** 19831

[8] Yu D G, Zhou Y J and Zhang S Y 2024 *New J. Phys.* **26** 013031

[9] Joshi A, Kumar A and Kaushik V 2024 *Functional Genomics and Network Biology* 2nd Edn. (Singapore: Springer), p. 71

[10] Yu C, Zhang T, Chen F and Yu Z 2024 *PeerJ* **12** e18476

[11] Becker R A, Eick S G and Wilks A R 1995 *IEEE Transactions on Visualization and Computer Graphics* **1** 16

[12] Velitchko F, Alessio A, Markus B, and Silvia M 2024 *IEEE Transactions on Visualization and Computer Graphics* **30** 5847

[13] Lyu H, Kureh Y H, Vendrow J and Porter M A 2024 *Nat. Commun.* **15** 224

[14] Wang H, Yan H and Rong C 2024 *ACM Comput. Surv.* **56** 1

[15] Geipel M M 2007 *International Journal of Modern Physics C* **18** 1537

[16] Zaida C R, Anuska F, Sandra M, Luka K, Felix M A 2012 *Scientometrics* **93** 699

[17] Fruchterman T M J and Reingold E M 1991 *Software: Practice and experience* **21** 1129

[18] Fan X Y 2020 *Master's Projects* **23** 050502

[19] Scalfani V F, Patel V D and Fernandez A M 2022 *Journal of Cheminformatics* **14** 87

[20] Maivizhi R, Sendhilkumar S and Mahalakshmi G S 2016 *Proceedings of the International Conference on Informatics and Analytics* (Pondicherry, India: Association for Computing Machinery) p. 8

[21] Abdelsadek Y, Chelghoum K, Herrmann F, Kacem I, and Otjacques B 2018 *Information Sciences* **424** 204

[22] Rodrigues J J F, Traina A J M, Faloutsos C and Traina J R C 2006 *Proceedings of the Eighth IEEE International Symposium on Multimedia (ISM'06)*, 2006, p. 227

[23] Leskovec J, Huttenlocher D and Kleinberg J 2010 *Proceedings of the SIGCHI conference on human factors in computing systems* April 10, 2010, Atlanta, Georgia, USA, p. 1361

[24] Jacomy M, Venturini T, Heymann S and Bastian M 2014 *PloS one* **9** e98679

[25] Vehlow C, Reinhardt T, Weiskopf D 2013 *IEEE Transactions on Visualization and Computer Graphics* **19** 2486

[26] Zhao R Q, Wu Y, Chen X 2017 *Journal of Computer-Aided Design & Computer Graphics* **29** 9 (in Chinese)

[27] Wang G J, Chen H R and Zhou R 2023 *Applied Sciences* **13** 12873

[28] Crampes M, Plantie M 2014 *Advances in Complex Systems* **17** 1450001

[29] Palla G, Derényi I, Farkas I and Vicsek T 2005 *Nature* **435** 814

[30] Huang Z H, Wu J X, Zhu W T and Wang Z Y 2021 *Physica A* **565** 125506

[31] Rossetti G and Cazabet R 2019 *ACM Comput. Surv.* **51** 37

[32] Zhang M N, Xiao J, Xu X K 2023 *Complex Systems and Complexity Science* **20** 10 (in Chinese)

[33] Zhou R, Wang G J and Deng H T 2022 *Journal Of Computer Science & Technology* **39** 479 (in Chinese)

[34] Zhou X, Huang T L and Liang X 2017 *Computer and Modernization* **0** 1 (in Chinese)

[35] Cai M, Luo H, Meng X, Cui Y and Wang W 2023 *Information Processing & Management* **60** 103197

[36] Coleman M K and Parker D S 1996 *Software: Practice and Experience* **26** 1415

[37] Purchase H 1997 *International Symposium on Graph Drawing* Kyoto, Japan, p. 248

[38] Breiger R L, Boorman S A, and Arabie P 1975 *Journal of Mathematical Psychology* **12** 328

[39] Alhajj R and Rokne J 2018 *Encyclopedia of Social Network Analysis and Mining* (New York: Springer New York) p. 1034

[40] Lusseau D, Schneider K, Boisseau O J, Haase P, Slooten E, and Dawson S M 2003 *Behavioral Ecology and Sociobiology* **54** 396

[41] Zachary W W 1977 *Journal of Anthropological Research* **33** 452

[42] Kunegis J 2013 *Proceedings of the 22nd International Conference on World Wide Web*, May 13, 2013, Rio de Janeiro, Brazil, p. 1343

[43] Girvan M and Newman M E J 2002 *Proc. Natl. Acad. Sci. USA* **99** 7821

[44] https://snap.stanford.edu/data/index.html

[45] De N W, Mrvar A and Batagelj V 2018 *Exploratory social network analysis with Pajek: Revised and expanded edition for updated software* (Cambridge: Cambridge University Press), p. 46

[46] Alès Z, Engelbeen C, Figueiredo R 2023 *Informs J. Comput.* **36** 672