Hierarchical Layout Methods in a Hybrid Framework of Topological Ordering and Graph Grammar

Ning Li*

College of Computer and Artificial Intelligence, Nanjing University of Finance and Economics, Nanjing, 210023, China
**Corresponding author:1546136968@qq.com*

Abstract: *Flowchart layout is of significant value in fields such as software engineering and workflow modeling. However, traditional automatic layout algorithms, including hierarchical layout and force-directed methods, exhibit notable limitations. To address these shortcomings, this paper presents a hybrid optimization framework that integrates graph grammar with topological ordering, aiming to achieve a balance between semantic integrity and computational efficiency. While traditional hierarchical layout algorithms, represented by the Sugiyama framework, eliminate loops by enforcing hierarchical structures, they often compromise the semantic relationships inherent in the original processes. Conversely, force-directed algorithms maintain topological features but face limitations in practical applications due to their computational complexity and stringent constraints on rules. The methodology proposed in this paper is enhanced by the following innovations: Semantics-Driven De-looping and Layering Mechanism: This approach introduces generative rules from graph grammar to remove loops in the original graph structure. Unlike traditional de-looping and layering strategies, which achieve structural optimization through rigid layering, our method relies on rule matching to effectively preserve the semantic integrity of the original processes. Parallelized Implementation of Hierarchical Sorting: This innovation encompasses the hierarchical partitioning of graph structures through topological sorting and graph grammar in both directions, thoroughly considering the topological framework and semantic information embedded within the graphs.*

Keywords: *automatic layout optimization; graph grammar; topological ordering; semantic integrity*

1. Introduction

In the era of big data, visualization technology is playing an increasingly important role as a bridge connecting data and human cognition. This technology integrates computer graphics, image processing and human-computer interaction technologies to realize efficient presentation and interactive analysis of data by transforming abstract data into intuitive graphic or image forms. Its core goal is to break through the limitations of human cognition, reveal the complex laws behind the data through visual mapping, and provide powerful support for decision-making and knowledge discovery. As a core tool for describing process logic in visualization technology, flowchart shows strong explanatory power and adaptability in many disciplines by virtue of its standardized symbol system and hierarchical structure design.

Flowcharts are based on standardized graphical symbols to build a semantically clear logic network, whose basic elements include ellipses (start/stop symbols), rectangles (processing steps), diamonds (decision points), parallelograms (inputs/outputs) and arrows (process lines). These basic semantic units form a directional topological network through the combination of four core logical structures: sequential, selective, cyclic and branching. For example, in the field of software engineering, the sequential structure clearly describes the linear execution process of an algorithm, while the cyclic structure provides an intuitive logical framework for program design by realizing conditional iteration through diamond-shaped decision nodes. This structured expression not only reduces the cognitive difficulty of complex systems, but also makes cross-domain collaboration possible.

With the continuous expansion of application scenarios, flowchart technology faces new challenges. Traditional layout methods gradually show their limitations when dealing with complex systems:

Hierarchical layout (e.g., Sugiyama framework) eliminates loops by enforcing hierarchies, and its core process consists of four steps: node hierarchies, intersection reduction, coordinate computation, and plotting. The method ensures that edges connect only neighboring layers by assigning nodes to different layers and introducing pseudo-nodes, but this hard layering mechanism can cut off reaction paths that are characterized by natural loops in scenarios such as biometabolic networks. For example, in tricarboxylic acid cycle visualization, the traditional hierarchical layout assigns the citrate synthase reaction and the malate dehydrogenase reaction to different layers, which disrupts the continuity of the metabolic pathways and leads to difficulty in identifying the correlations of key metabolic intermediates.

Although force-directed algorithms can naturally present the loop structure, their physical simulation mechanism leads to O (n^2) time complexity. The layout computation time increases exponentially when the number of nodes exceeds 200. In a fault recovery flowchart experiment with 500 nodes, the layout computation time of the traditional force-directed algorithm is more than 2 hours, and the dynamic balance of gravitational repulsion between nodes is easily affected by the initial position, which leads to the lack of stability of the layout results. In addition, the method is unable to effectively reflect the temporal characteristics of business processes when dealing with data flows with clear hierarchical relationships.

These technical bottlenecks are particularly prominent in emerging fields. Taking bioinformatics as an example, metabolic networks contain a large number of loop feedback mechanisms, and the traditional hierarchical layout will destroy the natural correlation of enzymatic reactions, while the computational efficiency of force-directed algorithms is difficult to cope with large-scale networks with thousands of nodes at any time. In quantum computing logic design, the timing dependencies of quantum gate operations often form complex loops, and the existing layout methods are not able to efficiently generate the layout, and it is difficult to maintain the semantic integrity of quantum state transitions. For example, in a flowchart layout experiment of a quantum error-correcting code, the traditional method results in 37% of the key control edges crossing, which seriously affects the efficiency of logic verification.

Facing the above challenges, this paper focuses on the key technological breakthroughs in flowchart layout. By constructing a hybrid optimization framework based on graphical method and topological ordering, it aims to achieve the double improvement of semantic integrity and computational efficiency.

2. Related Work

In the field of graph data visualization, graph layout plays a crucial role. It aims to present graph - structured data in an intuitive and effective way, helping people understand complex data relationships. Currently, the main research directions of graph layout lie in force - directed layout methods and layout methods based on aesthetic criteria. The force - directed layout method simulates the forces in a physical system to make the nodes distribute naturally in the graph, reducing edge crossings and optimizing the compactness of the layout, thus enhancing the visualization effect. The layout method based on aesthetic criteria, on the other hand, focuses on constructing an aesthetically pleasing and easy - to - interpret graph layout according to a series of aesthetic principles, such as symmetry and balance. These two main research directions are constantly evolving, promoting the development of graph layout technology. They provide powerful data visualization tools for many fields, such as social network analysis and software engineering, enabling researchers and practitioners to better explore the hidden information behind graph data and reveal the internal structure and laws of complex systems.

The origin of force-directed layout algorithms can be traced back to 1963, and its predecessor is Tutte's algorithm [1], which was improved based on the center of gravity method for solving the layout problem of a three-connected graph. Subsequently, Eades [2] proposed the spring embedder, which treats the nodes of a graph as coils and the edges as springs, with all nodes in an initial state and the spring forces acting on the coils until the graph reaches a state of minimum energy.Kamada and Kawai [3] proposed the KK (Kamada-Kawai) algorithm, which is based on Eades' spring embedder. Kamada and Kawai [4] proposed the KK (Kamada) algorithm based on the Eades spring model, which takes the shortest path distance between the nodes as the ideal distance in the layout, and relates the uniform layout of the nodes to the dynamically equilibrated spring system, and generates an equilibrium graph layout when the energy function of the spring system reaches a minimum value. In the electrical model, gravitational and repulsive forces are introduced. The two nodes that are connected are subject to the gravitational force fa and the nodes that are not directly connected are subject to the repulsive force fr .

In each iteration of the FR algorithm, only the gravitational and repulsive forces on one node are calculated and moved.The DH (Davidson Harel) algorithm [5] uses the core idea of simulated annealing to both prevent nodes from approaching non-neighboring edges and to reduce the crossing of edges. The energy value of the graph is the sum of gravitational and repulsive forces on all nodes.

All of the above methods have the problem of not being able to show the clustering of nodes in the graph. Noack [6] proposed the LinLog algorithm, which is based on the BH algorithm and optimizes the energy function with the help of a quadtree model to cluster nodes in close proximity and visually classify different clusters. Inspired by LinLog and FR algorithms, Jacomy et al. [7] proposed the ForceAtlas2 algorithm. In this algorithm, the gravitational force is proportional to the distance between nodes, and the repulsive force is adjusted according to the degree (degree) of the nodes to bring the leaf nodes closer to the parent nodes. In addition, ForceAtlas2 incorporates simulated annealing ideas to achieve localized temperature and adaptive cooling rate to generate coherent layouts.

In order to improve the efficiency of force-directed algorithms for visualizing the layout of large-scale graph data, some researchers propose a multi-layer iterative force-directed algorithm [8 - 9]. The algorithm mainly contains three steps, namely, coarsening, initialization and single-layer layout: firstly, the graph is gradually decomposed into small graphs, and the sequence of coarsened graphs is retained (coarsening); next, the force-directed algorithm of the coarsened graph is used to perform the initialization of the layout (initialization); and finally the nodes are added according to the sequence of the coarsened graphs and the layout is performed (single-layer layout), so as to complete the layout of the original graphs. The roughening step is the most critical because the structure of the roughened graph affects the subsequent computation of the gravitational and repulsive forces, which in turn affects the visual effect of the final graph layout.Hachul and Jünger [10] proposed the FM (3 fast multipole multilevel method) algorithm in conjunction with a multilevel iterative method.FM3 algorithm retains the collapsed nodes after the folded nodes have been collapsed in the roughened graph by controlling the radius of the subgraphs. The FM3 algorithm is more effective than other multilevel iterative methods in generating satisfactory layouts by controlling the radius of subgraphs in the roughing stage to preserve the sequence of roughened graphs after folding nodes.

The aesthetics of a graph layout greatly affects the readability of the graph and the difficulty of graph-related tasks. Different researchers have conducted user experiments based on graph-related tasks to investigate the task performance and related behaviors of users during the exploration of graph visualization.Holton et al. [11] used different edge representations to visualize graphs, and the results of the user experiments showed that the use of curved edges was not conducive to the efficient completion of the task.Huang [12] used eye tracking techniques to find out that the increased number of edge crossings led to slower eye movements and lower task completion efficiency when subjects searched on the graph, and the more the angle of edge crossings, the more the task was completed. lead to slower eye movements and less efficient task completion when subjects search on the graph; and the smaller the angle of the edge crossings, the more repeated eye movements near the crossings. This shows that the use of inappropriate representation elements and difficult-to-recognize layouts in graph visualization reduces the efficiency of users' perception of graphs, which in turn affects the performance of graph-related tasks.

Therefore, summarizing the aesthetic evaluation criteria of graph layout and constraining the graph visualization layout based on these criteria can effectively improve the readability of graph visualization.Purchase [13] summarized seven aesthetic evaluation criteria for node-link graph visualization. In graph visualization, the number of edge intersections and small intersection angles can reduce the readability of graph visualization and seriously affect the efficiency of users' perception of graph data. Therefore, minimizing the number of edge crossings and maximizing the minimum angle of edge crossing angles are the most commonly used aesthetic constraints when visualizing graph data. While symmetry of graph layout and orthogonality of edges and nodes have relatively less impact on users' cognition of graph data, they are usually used as optional aesthetic constraints to improve the aesthetics of graph visualization. Consistent flow direction is usually applied to tree visualization.

3. Method

In the field of data visualization, when dealing with data that exhibits a specific hierarchical structure or sequential order, hierarchical layout becomes an ideal choice due to its unique advantages. The commonly adapted data structure for this layout is the Directed Acyclic Graph (DAG). In real - world scenarios, hierarchical layout is widely applied. For example, in the flowcharts used in daily corporate operations, it clearly shows the sequence of business processes. In organizational structure diagrams, it intuitively presents the hierarchical relationships within an enterprise. And in state transition diagrams, it accurately describes the transformation logic between system states.The hierarchical layout approach was first elaborated by Kozo Sugiyama in 1981, hence the name Sugiyama layout. The layout aims to automatically generate easy-to-understand hierarchical diagrams based on the data characteristics of the diagrams.Sugiyama decomposes the problem of diagram layout into multiple steps, each of which focuses on solving a different subproblem.The Sugiyama algorithm consists of the following four main steps:

Node Layering: Nodes in the graph are divided into layers based on the orientation of the edges. If there are edges that span multiple layers, a pseudo-node is added to each layer that the edge crosses, thus ensuring that each edge connects only two neighboring layers.

Reduced crossover: by changing the order of nodes in each layer in order to reduce edge crossover.

Calculate node coordinates: adjust the position of the nodes while maintaining the order of the nodes determined in the previous step.

Drawing: draws a graph based on the node position information generated by the preceding steps and removes pseudo nodes and associated edges added during the process.

During the layout process, the reduction process of graph grammar plays a crucial role. Introducing graph grammars, graph grammars are formal tools used to set a graph language and analyze the grammar of a graph. A graph grammargg is a triple$(\mathrm{A}, \mathrm{P}, \mathrm{E})$ where$\mathrm{A}$ is the initial graph of the grammar,P is a set of generating equations, and$\mathrm{E}$ is the embedding rules of the grammar. The initial graph is the starting point of the transformation of graphs by the graph grammar, and all graphs in the grammar are obtained by transforming the initial graph. The graph grammar can either find subgraphs in the main graph that are isomorphic to the generative left graph and satisfy the replacement condition, and replace them with the generative right graph; or find subgraphs that are isomorphic to the generative right graph and satisfy the replacement condition, and replace them with the generative left graph. In both processes, we refer to the replaceable subgraph as a handle (Redex). The former process is called derivation, which produces a set of graphs from the initial graph, called the language produced by the graph grammar. The latter process is called reductio ad absurdum, and reductio ad absurdum determines whether a graph belongs to the language produced by a given graph grammar.

Figure 1 illustrates a set of generative formulas for a graph grammar, and Figure 2 shows the process of statute for a graph that contains the basic structure of a flowchart.

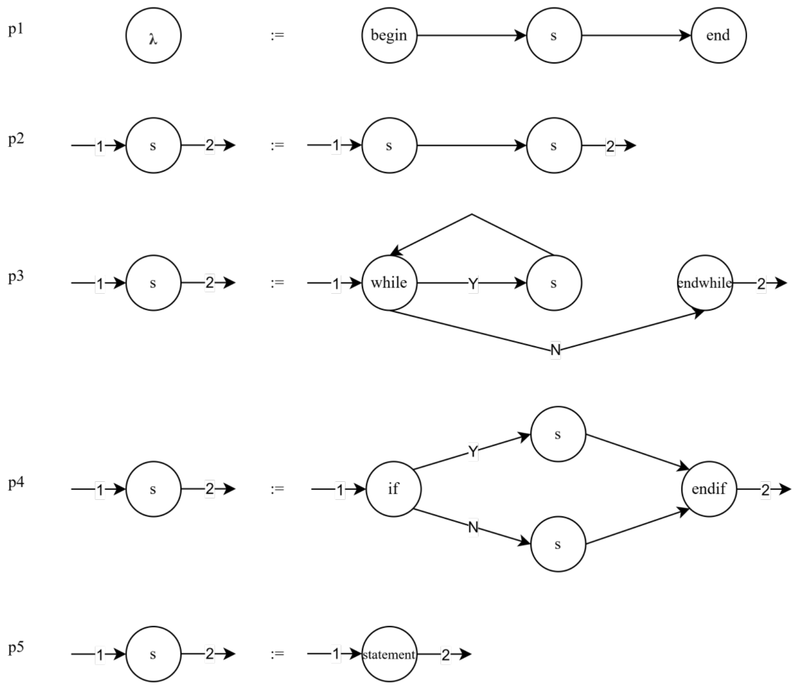

Figure 1 A set of generative formulas for the graphic method

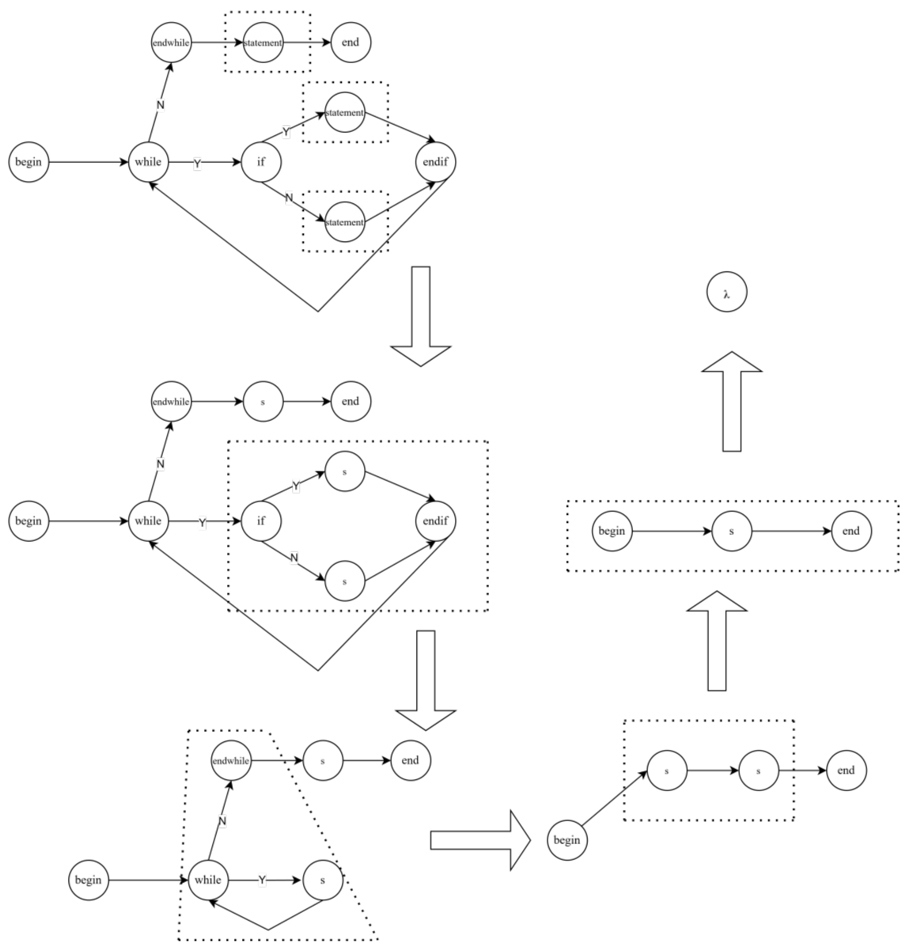

Figure 2 Statutory process of a diagram

The following detailed description of the graph layout method in the hybrid framework of graph grammar and topological ordering is divided into the following four main steps:

Step 1:Since traditional hierarchical layout methods target directed acyclic graphs, there are often many loops in flowcharts, and traditional de-looping methods do not consider the node semantics of the graph, the graph grammar allows graph rewriting, which can be based on generative formulas to de-loop substructures containing loops, and therefore preserves the graph substructure more completely.

Step 2: Use the longest path algorithm to topologically sort the graph after de-ringing, the number of layers for each node is RANK, after topologically sorted layering at this point in the flow graph is a straight chain, the same layer will be more than one node overlap.

Step 3: Need to sort and stratify the nodes overlapping in the same layer, the nodes in the same layer sort number level, at this time the graph grammar is introduced to sort the nodes, the graph is statute, record the number of times of the statute, each node is statute, its corresponding number of statute for the number of sorting level.

Step 4: Draw the graph, according to each node layer rank and the same level of sorting number level graph, x coordinate for k * rank, the same rank with the level of the number of nodes for the i, where the jth node of the y coordinate for v * [(level-1) + j / (i + 1)], k and v by human settings to adjust the scale of the graph layout, in the graph of the time to restore in the graph code about the time of Remove loops and set the removed edges as curve edges.Figure 3 shows the layout effect of this method for a flowchart.

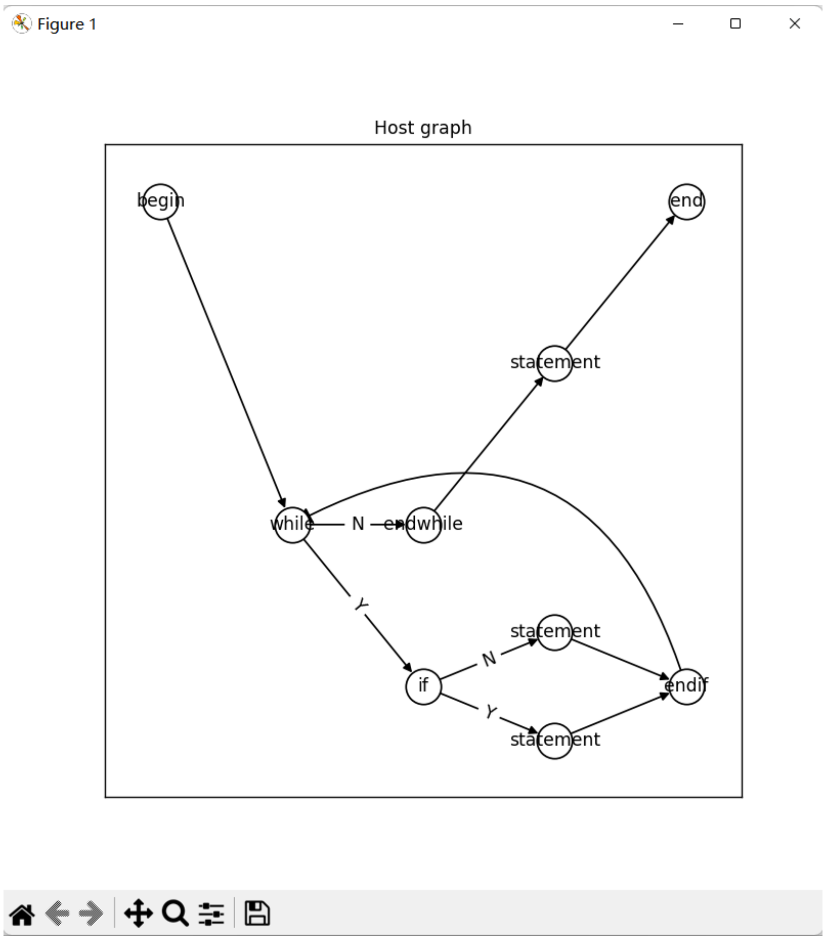

Figure 3 Layout of a flowchart

4. Comparison

In the research field of graph layout algorithms, the mainstream force-directed layout methods and hierarchical layout methods each face serious challenges when dealing with complex systems with ring structures.

The force-directed layout method has the advantage of naturally presenting loops, which can intuitively reflect the topological characteristics of loop structures in complex systems. In scenarios such as biometabolic networks, failure recovery processes, etc., this intuitive rendering helps researchers to understand the cyclic relationships among the elements in the system. However, the method has a significant limitation with a time complexity of O (n^2). This means that the computational process incurs significant delays when the number of nodes exceeds 200.

Feedback arc set (FAS)-based de-loop methods, such as the Greedy FAS algorithm, aim at constructing directed acyclic graph (DAG) approximations by removing some edges to simplify the structure of complex systems for subsequent analysis and processing. However, this edge removal strategy inevitably brings the problem of critical path information loss while realizing the de-loop. In biological metabolic networks, critical paths often represent the core metabolic pathways; in the fault recovery process, critical paths are related to the most effective recovery steps. Once the critical path information is lost, it may lead to a wrong understanding of the system operation mechanism, which in turn affects the decisions made based on these analyses.

Hierarchical algorithms, typified by the Sugiyama algorithm, are able to generate a hierarchical DAG layout through the three phases of de-ringing, hierarchizing, and sorting. This layout helps to show the system structure clearly to a certain extent, which is convenient for users to understand. However, when dealing with complex systems with ring structure, the forced hierarchical operation of the algorithm will destroy the semantic association of the original processes. For example, in a biological metabolic network, some metabolic reactions are closely semantically related to each other despite the existence of loops, and forced layering may separate these closely related reactions at different levels, making it difficult for the user to understand the intrinsic logical relationship between them, and thus failing to accurately grasp the operation mechanism of the whole system.

In contrast, graph grammar shows unique advantages in removing loops. Graph grammar can flexibly transform the graph according to its production rules and embedding rules. When dealing with a graph with a loop structure, graph grammar can accurately identify the key sub-graph structures within the loop. Through reasonable replacement or adjustment, it can skillfully remove the loop structure while retaining the semantic relationships among the elements in the graph to the greatest extent possible. For example, in an electric power transmission network, there may be complex loop network structures. During the process of loop removal, graph grammar can maintain the connection logic between each transmission line and the substation. This ensures that when users conduct subsequent analysis, they can still clearly understand the potential power transmission paths and their interrelationships, avoiding the loss of key information due to the loop removal operation. It provides a more effective means for understanding complex systems.

In summary, all existing layout methods have shortcomings when dealing with ring structures in the system, and a new layout method is needed to balance topological feature preservation and computational feasibility. The present method achieves a better layout effect by parallel topological ordering and graphical code approximation, which fully considers the topological structure and semantic information of the graph.

5. Conclusion

In the field of graph data visualization, traditional layout algorithms such as the Sugiyama framework and the force-directed model have built a mature paradigm, but their limitations are significantly exposed when dealing with non-idealized graph structures. These algorithms are difficult to achieve effective partitioning and structural optimization of acyclic subgraphs when dealing with complex graph structures, especially graphs containing special topological features, and they are also unable to improve the quality of automatic layout of complex flowcharts.

To break through the above dilemma, this paper proposes to construct an adaptive layout framework based on Graph Grammar. Graph Grammar demonstrates unique theoretical advantages by driving layout generation through formalized rules. Specifically, Graph Grammar Productions can encode layout specifications into executable grammar rules, thus realizing the dynamic integration of global constraints and local optimization. At the same time, the graph grammar can non-destructively process the ring structure to make up for the shortcomings of traditional algorithms.

Future work will mainly focus on the research of the theoretical aspects of graph grammar rules to better meet the layout requirements of different graph models. As a key tool for connecting abstract logic and concrete practice, the structural innovation and application expansion of flowcharts are crucial for the cognition and optimization of complex systems. With the rapid development of science and technology, future research needs to further explore the adaptability of flowcharts in emerging fields such as quantum computing and bioinformatics, and promote the iterative upgrading of visual analysis paradigms, so as to satisfy the ever-increasing demand for complex data visualization, and continue to empower the in-depth understanding and efficient processing of complex systems.

References

[1] TUTTE W T.How to draw a graph[J].Proceedings of the London Mathematical Society, 1963, 3(1): 743-767.
[2] EADES P.A heuristic for graph drawing[J].Congressus Numerantium, 1984, 42: 149-160.
[3] KAMADA T, KAWAI S.An algorithm for drawing general undirected graphs[J].Information Processing Letters, 1989, 31(1): 7-15.
[4] FRUCHTERMAN T M J, REINGOLD E M.Graph drawing by force-directed placement[J]. Software: Practice and Experience, 1991, 21(11): 1129-1164.
[5] DAVIDSON R, HAREL D.Drawing graphs nicely using simulated annealing[J].ACM Transactions on Graphics, 1996, 15(4): 301-331.
[6] NOACK A.An energy model for visual graph clustering[C]//Proceedings of the 11th International Symposium on Graph Drawing, Perugia, 2003: 425-436.
[7] JACOMY M, VENTURINI T, HEYMANN S, et al.ForceAtlas2, a continuous graph layout algorithm

for handy network visualization designed for the Gephi software[J].PLoS ONE, 2014, 9(6): e98679.
[8] HADANY R, HAREL D.A multi-scale algorithm for drawing graphs nicely[J].Discrete Applied Mathematics, 2001, 113(1): 3-21.
[9] WALSHAW C.A multilevel algorithm for force-directed graph drawing[C]//Proceedings of the 8th International Symposium on Graph Drawing, Heidelberg, 2000: 171-182.
[10] HACHUL S, JÜNGER M. Large-graph layout algorithms at work: an experimental study[J]. Journal of Graph Algorithms and Applications, 2007, 11(21): 345-369.
[11] HOLTEN D, VAN WIJK J J.A user study on visualizing directed edges in graphs[C]//Proceedings of the 2019 SIGCHI Conference on Human Factors in Computing Systems, Boston, 2009: 2299-2308.
[12] HUANG W. Establishing aesthetics based on human graph reading behavior: two eye tracking studies[J].Personal & Ubiquitous Computing, 2013, 17(1): 93-105.
[13] PURCHASE H C. Metrics for graph drawing aesthetics[J].Journal of Visual Languages & Computing, 2002, 13(5): 501-516.