Advances in Intelligent Systems Research, volume 147

International Conference on Network, Communication, Computer Engineering (NCCE 2018)

Research on Auto-Drawing Technology of Orthogonal Network Topological Graph with High Degree Nodes

Zhongyan Liang [a)], Jie Hu [b)] and Qiaojin Guo [c)]

The 28th Research Institute of China Electronic Technology Group Corporation, Nanjing 210000, China.

[a)] Corresponding author: aptivaleung@foxmail.com
[b)] hujieyc@139.com
[c)]guoqiaojin@163.com

Abstract. Automatic network topology drawing technology has been studied by many scholars, including straight-line layout and orthogonal layout. There are lots of toolkits to do this task, such as Graphic and OGDF (The Open Graph Drawing Framework). However, the automatic layout results for some drawings are still not satisfactory. In this paper, we research on the automatic drawing algorithm supporting orthogonal polylines, which can better support the network topology with high degree nodes. We have compared with classic algorithms, implemented by Graphic and OGDF, and Microsoft Visio, and the test results show that our algorithm is more suitable for aesthetic requirements and needs less manual intervention.

Key words: Orthogonal layout, Topological graph, Automatic drawing.

INTRODUCTION

When developing some monitoring software, it is necessary to graphically display the connection relationship of a group of physical objects to users, such as network topology, communication topology, and power topology. The physical objects in these topological diagrams often have a large number of types and complicated connection relationships. If the hard-coded methods are used to draw the graphs, it will not only take time and effort, but if there is a change in the later period, it is also not conducive to maintenance. These topological graphs are generally composed of fixed-shape icons and connecting lines. If it is possible to develop an algorithm that automatically calculates the position of the icons and links through the given topology graph information, and allows manual adjustments, many development and maintenance time can be saved.

RELATED WORKS

The graph layout algorithm has a long history of development. From the regular graph layout algorithm to the later developed force-directed algorithm, and the subsequent visualization algorithm for the complex graphs using various compression functions, it has undergone a long period of development [1]. Many scholars have conducted in-depth research on this technique. Evades [2] proposed a Spring-Embedded Model, which focused on the "spring force" in physics. Fruchterman and Rheingold [3] proposed an atomic attractive force model (Force-Directed Model) to replace the spring model. Calculating the coordinates of the node by calculating the attractive force between the node and the neighboring nodes, and the repulsive force between them and the surrounding nodes. Kamara and Kawai [4] proposed an Energy Model, improved Evades' spring model, and determined the position of the nodes in the graph by finding the minimum value of the system's total energy. Hu [5] proposed a multi-scale force-directed layout using a spring-electrical model. This algorithm combines multi-level method with the Barnes and Hut's [6] quad tree (2D) / ochre (3D) algorithm to effectively overcome local minima and to be able to approximate both

short-range and long-range forces satisfactorily and efficiently. Wills [7] proposed a radial layout. This algorithm was designed for very large graphs, which can display networks with 20,000-1,000,000 nodes in real time. The orthogonal layout algorithms, which represents edges as sequences of horizontal and vertical straight-line segments, were proposed in [8, 9]. The circular layout places the vertices on a circle to form vertices of a regular polygon which is spaced evenly. This layout is ideal for communication network topologies, such as star or ring networks [10], and for the cyclic parts of metabolic networks [11].

THE PROPOSED METHOD

In general, topological graph algorithms should meet aesthetic standards. First of all, in order to clearly show the structure of the graph, the number of intersections of the edges should be reduced. Secondly, adjacent nodes should be as close as possible to reduce the length of the edges. Thirdly, to use straight lines as much as possible on the edges of the graph, avoiding curves and slashes. Fourthly, multiple edges of the same node try to balance the layout with this node as the center. Lastly, laying out the nodes as much as possible on different levels, horizontally or vertically.

In this paper, we research on automatic drawing algorithms for network topology graph with high degree nodes.

We mainly focus on the following types of network equipment: server, switch, SAN (Storage Area Network), SME (System Management Expert), MDVR (Mobile Digital Video Recorder), matrix, project, UPS (Uninterruptible Power Supply), printer, console, router, and firewall. In addition, we divide these devices into two categories as shown in the following table: core devices and terminal devices.

TABLE 1. Device Categories.

Device Categories	Device Descriptions
Core device	Switch, SAN, SME, Router, Firewall
Terminal device	Server, MDVR, matrix, project, UPS, printer, console

Our algorithm is described as follows:

The graph is denoted as $G=(V,E)$, where $V=(x,y,w,h)$ is the set of nodes with coordinates (x,y), width w and height h, and $E=[(x_0,y_0),(x_1,y_1),...,(x_{n+1},y_{n+1})]$ with the number of vertices n. E Contains at least two coordinate points, connecting two nodes. The output is a set of layout coordinates for each graph element [12] and the coordinates of vertices of each edge.

First of all, the canvas is divided into five parts: top, bottom, left, right and center part.

For core devices, we use the circular layout [12-14] as the automatic drawing engine and draw it in the center of the canvas.

For terminal devices, we use the following algorithm:

Servers and consoles are arranged at the top or bottom of the canvas according to the distance to the device connected to it.

MDVR, matrix, project, UPS, and printer separate the left and right sides of the canvas according to the distance from the device connected to it.

If no line type is specified, a linear connection is used between the devices, and a polyline is used between a core device and a terminal device. There is generally no connection requirement between terminal devices.

EXPERIMENTS

The Experiment Results

The graph used in our experiments exist multiple connection relationships. For example, the server and console link two switches at the same time. The ideal target topology drawing in the experiment is shown in the following figure and we use steps to manually modify the automated layout to the target drawing and the number of crossings [12] to evaluate algorithms:

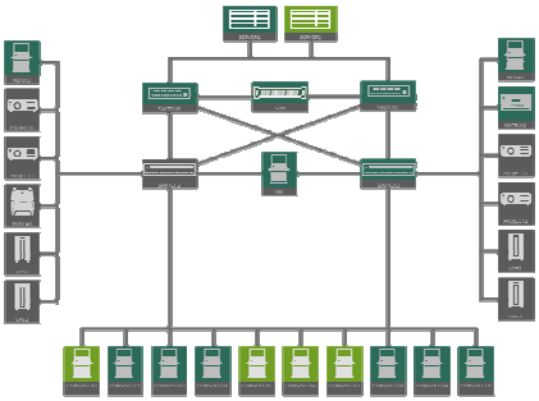
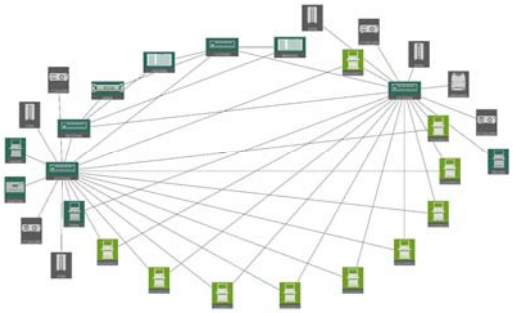


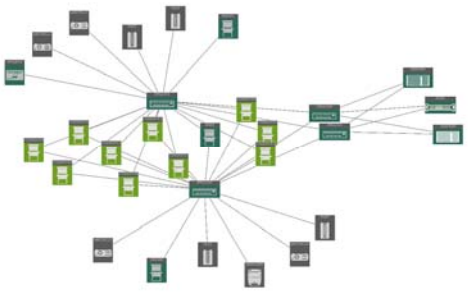
FIGURE 1. The ideal target topology drawing.

Discussions

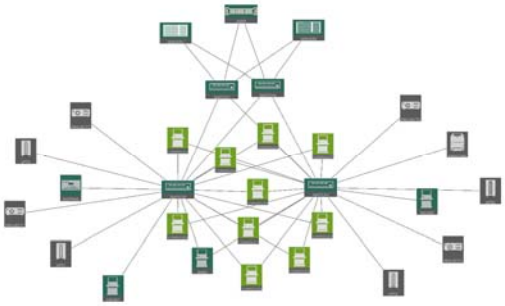
The algorithms shown in Fig. 2 (a) ~ (f) do not support polylines fail to meet the requirement that the target topology graph needs to support polylines. Because each console is connected to two switches, it cannot be seen at a glance that the connection relationships between switches and consoles. For Fig. 2 (g), we manually draw the icon position with reference to the target drawing's icon position, and automatically connect by Microsoft Visio with the given line type. Since the vertices are hardly all on the same line, the drawing is cluttered. Fig. 2 (h) uses the automatic alignment and auto-adjustment of Microsoft Visio. The arrangement result of Fig. 2 (h) is more chaotic than Fig. 2 (g). The numbers of edge crossing excluding coincident lines are shown in Table 2, where the proposed method shows a good edge crossing number. The result of the proposed algorithm (Fig. 2 (j)) is more orderly, and the connections are clear. The target layout can be drawn with minimal manual intervention.



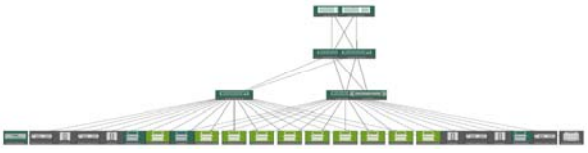
(a)Circular layout [12-14]



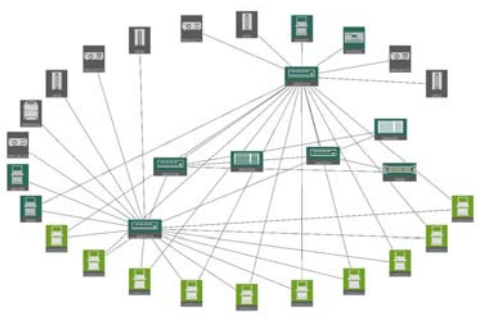
(b)Force-directed algorithm [3]



(c)Hu's algorithm [5]

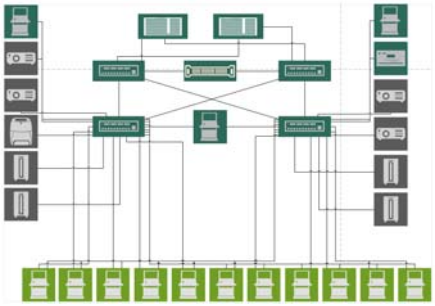


(d)Hierarchical layout [15, 16]

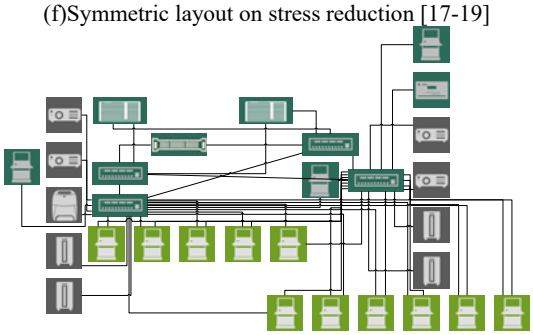


(e)Wills' algorithm [7]

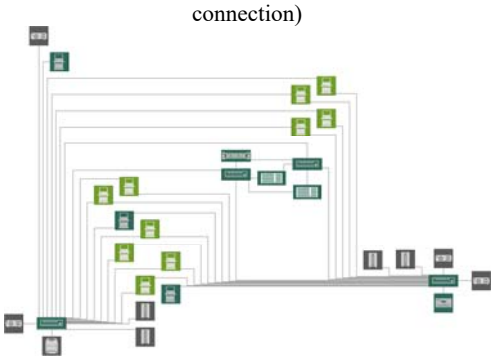
(f)Symmetric layout on stress reduction [17-19]



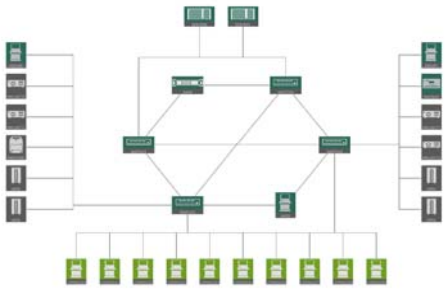
(g)Visio Layout Result (Manual layout, automatic connection)



(h)Visio Auto Layout (automatic alignment and automatic adjustment)



(i)Orthogonal Layout [8, 20]



(j)The proposed algorithm

FIGURE 2. Automatic drawing results using different algorithms. Algorithm (a) ~ (f) do not support polylines.

TABLE 2. Steps to manually modify the automated layout to the target drawing

Figure	Steps	Number of Edge Crossing
(a)	Not Available due to lack of polyline supporting (N/A)	50
(b)	N/A	13
(c)	N/A	22
(d)	N/A	58
(e)	N/A	77
(f)	N/A	16
(g)	>20	44
(h)	>20	70
(i)	>20	0
(j)	4	1

CONCLUSION

In this paper, we propose an algorithm for network topology graph with high degree nodes. The experimental results show that algorithms that only support straight lines are not suitable for topologies that have high degree nodes. In order to make the drawing conform to the aesthetic standards, we need to do special processing on orthogonal polylines, and we need to arrange the breakpoints on the same line. This paper is only a preliminary study of the network topology graph that supports orthogonal polylines. And there are still many problems that need to be solved. Especially for complex network structures.

REFERENCES

1. Liu, L. and Y. Li, Network Topology Layout Algorithm Based on Network Levels. Command Information System and Technology. 6(5): 35-39 (2015).
2. Eades, P.A. "A heuristic for graph drawing". in Congressus Numerantium. (1984), pp. 149-160.
3. Fruchterman, T.M.J. and E.M. Reingold, Graph Drawing by Force-directed Placement. Software Practice and Experience. 21(11): 1129-1164 (1991).
4. Kamada, T. and S. Kawai, an algorithm for drawing general undirected graphs. Information Processing Letters. 31(1): 7-15 (1989).
5. Hu, Y.F., Efficient and high-quality force-directed graph drawing. Mathematica Journal. 10: 37-71 (2005).
6. Barnes, J. and P. Hut, A hierarchical O(NlogN) force-calculation algorithm. Nature. 324: 446-449 (1986).
7. Wills, G. "Nicheworks - interactive visualization of very large graphs". in Symposium on Graph Drawing GD'97 (volume 1353 of Lecture Notes in Computer Science). (1997), pp. 403-414.
8. Tamassia, R., G.D. Battista, and C. Batini, Automatic graph drawing and readability of diagrams. IEEE Trans Syst Man Cybern. SMC-18(1): 61-79 (1988).
9. Klau, G.W. and P. Mutzel. "Optimal compaction of orthogonal grid drawings". in Integer Programming and Combinatorial Optimization (volume 1610 of Lecture Notes Comput. Sci.). edited by G. Cornuejols, R.E. Burkard, and G.J. Woeginger (Springer-Verlag, 1999), pp. 304-319.
10. Doğrusöz, U., B. Madden, and P. Madden. "Circular layout in the Graph Layout toolkit". in Graph Drawing: Symposium on Graph Drawing, GD '96 (volume 1190 Lecture Notes in Computer Science). (Springer, Berkeley, California, USA, 1997), pp. 92-100.
11. Becker, M.Y. and I. Rojas, A graph layout algorithm for drawing metabolic pathways. Bioinformatics. 17(5): 461-467 (2001).
12. Six, J. and I. Tollis. "A framework for circular drawings of networks". in Proc. Symp. Graph Drawing GD'99 (volume 1731 of Lecture Notes in Computer Science). (2000), pp. 107-116.
13. Six, J. and I. Tollis. "Circular drawings of biconnected graphs". in Proc. ALENEX 99. (1999), pp. 57-73.
14. Kaufmann, M. and R. Wiese. "Maintaining the mental map for circular drawings". in Proc. Symp. Graph Drawing GD'02 (volume 2528 of Lecture Notes in Computer Science). (2002), pp. 12-22.
15. Sugiyama, K., S. Tagawa, and M. Toda, Methods for Visual Understanding of Hierarchical System Structures. IEEE Trans Syst Man Cybern. SMC-11(2): 109-125 (1981).
16. Emden, R., et al., A Technique for Drawing Directed Graphs. IEEE Trans. Software Engineering. 19(3): 214-230 (1993).
17. Kruskal, J. and J. Seery. "Designing network diagrams". in Proc. First General Conf. on Social Graphics. (1980), pp. 22-50.
18. Cohen, J., Drawing graphs to convey proximity: an incremental arrangement method. ACM Transactions on Computer-Human Interaction. 4(11): 197-229 (1987).
19. Gansner, E., Y. Koren, and S. North. "Graph drawing by stress majorization". in Proc. Symp. Graph Drawing GD'04. (2004).
20. Tamassia, R., On embedding a graph in the grid with the minimum number of bends. SIAM J. Comput. 16(3): 421-444 (1987).