Based on Force-Directed Algorithms Method for Metagraph Visualization

Larysa Globa[1], Maksym Ternovoy [2], Olena Shtogrina[3], Oleksandra Kryvenko[4]

[1] Professor, National Technical University of Ukraine "Kyiv Polytechnic Institute", Institute of Telecommunication Systems, Kyiv, Ukraine
lgloba@its.kpi.ua
[2] Associate professor, National Technical University of Ukraine "Kyiv Polytechnic Institute", Institute of Telecommunication Systems, Kyiv, Ukraine
ternovoy@its.kpi.ua
[3] Assistant, National Technical University of Ukraine "Kyiv Polytechnic Institute", Institute of Telecommunication Systems, Kyiv, Ukraine
L_Shtogrina@mail.ru
[4] Student, National Technical University of Ukraine "Kyiv Polytechnic Institute", Institute of Telecommunication Systems, Kyiv, Ukraine
okryvenko@stud.its.kpi.ua

Abstract. The paper describes the method for automatic metagraph visualization based on the principles of force-directed algorithms. The criteria under which the final image is understandable for users and corresponds to a predetermined metagraph are defined. This approach defines the set of the rules for forces between metagraph nodes depending on the types of the nodes between which the forces act. The analogue of Venn diagram is used to visualize the metagraph nodes. The method was tested on random metagraphs with up to 60 vertices and up to 25 metavertices.

Keywords: metagraph, visualization, layout, metavertex, graphical representation, graph.

1 Introduction

Nowadays information visualization tools are actively developing. These tools are converting the data into a form that allows using an ability to analyze visual images. The methods of graphical analysis can increase efficiency in decision support process. To visualize the interrelated data it is better to use graphs. But there are a lot of cases when common graph theory cannot be used to visualize data correctly. For example, we cannot describe and visualize vertices, nested vertices and relation with them in graph terms. In these cases the metagraphs can be used.

Metagraph is a graphical construct specified by its generating set and the set of edges defined on the generating set [4]. It can be used to model rule bases, data bases, model bases, business processes, etc.

There are many visualization algorithms for small (up to 200 nodes) and large (thousands of nodes) graphs. Visualization algorithms for small size graphs based on the physical analogues or force-directed algorithms are the simplest. They can be used to draw graphs of any kind. Images created with these algorithms meet the requirements of aesthetics criteria: they contain few intersections of edges and symmetry [3]. This group includes force algorithms [6, 8], algorithms that use the action of gravity [7], magnetic forces [14], algorithms based on the minimization of energy [12] and others. Visualization of large graphs is more complex problem. To solve this problem other approaches are needed. Large graphs visualization requires algorithms that use graph clusterization, graph GC-filtration [10], forces approximation [3], multi-scale methods [9, 10, 15], topological feature-based method [1], etc. These approaches are effective for graphs, but are unsuitable for metagraph because of different structure. Unlike graph visualization there are no well-known algorithms for metagraph visualization. In this paper we propose the method for automatic metagraph visualization.

2 State of the art and background

As mentioned above metagraph is specified by its generating set and the set of edges defined on the generating set [4]. In [2], by analogy with the hypergraph theory two components such as metavertices set and metaedges set are added to metagraph description. To solve the problem of metagraph visualization we decide to extend the metagraph definition given in [4] through the vertices set and metavertices set consideration separately. The set of edges contains all metagraph edges, no matter what types of nodes they connect. Suggested metagraph definition is described below.

Definition 1. Metagraph is a construct $S = \langle V,M,E \rangle$, where

$V = \{v_r | r = \overline{1,N_V}\}$ - set of metagraph vertices,

N_V - number of metagraph vertices;

$M = \{m_q | q = \overline{1,N_M}\}$ - set of metavertices, N_M - number of metavertices;

$E = \{e_h | h = \overline{1,N_E}\}$ - set of edges, N_E - number of metagraph edges.

Metavertex $m_q = \{v_r | v_r \in V, r = \overline{1,N_{m_q}}\}$ is a vertex which includes some other vertices which called inner vertices for this metavertex, N_{m_q} is a number of vertices included in m_q.

Metagraph node $mv \in (V \cup M)$ is a vertex or a metavertex.

Metagraph edge is oriented pair of vertices $e_h = (mv_{out}, mv_{in})$, where mv_{out} - tail, mv_{in} - head.

Graph visualization algorithms don't have mechanism that allows including vertices in other vertices. If we apply these algorithms to metagraph there are layout problems. Suppose given metagraph S_1 (Figure 1):

$$S_1 = \left\langle \begin{array}{l} \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}, \\ \{m_1, m_2, m_3, m_4\}, \\ \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\} \end{array} \right\rangle,$$

where $m_1 = \{v_2, v_3\}$, $m_2 = \{v_4, v_5\}$, $m_3 = \{v_3, v_5, v_6, v_7, v_8, v_9\}$, $m_4 = \{v_{10}, v_{11}\}$, $e_1 = (m_1, v_1)$, $e_2 = (m_2, v_1)$, $e_3 = (v_5, v_3)$, $e_4 = (v_6, v_2)$, $e_5 = (v_{10}, v_6)$, $e_6 = (m_4, m_3)$, $e_7 = (v_9, v_7)$.

Have a look at the ways to use graph visualization algorithm for metagraph. The first way is to interpret all metagraph nodes as graph vertices. In this case the metavertices positions and their inner vertices positions are computed independently. Therefore inner vertices of metavertex may be positioned at a significant distance from each other. This leads to a loss of the metavertex form.

The second way is to take into account only vertices and then lead round inner vertices to visualize metavertex. In this case it isn't guaranteed that only inner vertices are located in metavertex. Also incident to metavertex edges cannot be determined. Figure 2 shows the wrong location metagraph nodes when applied the algorithm for graph visualization. Wrong located nodes are marked grey. Also there are no edges between metavertices.

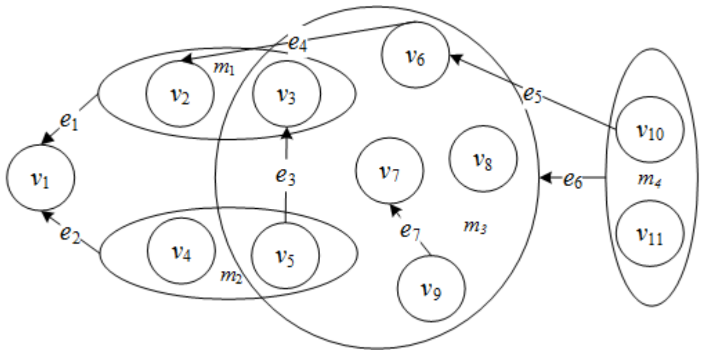


Fig. 1. Correctly positioned metagraph S_1

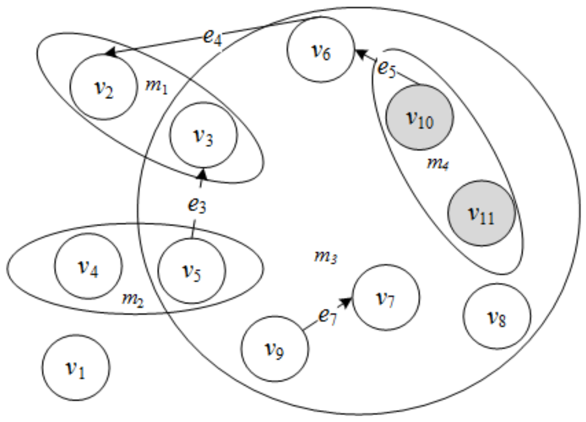


Fig. 2. Incorrectly positioned metagraph S_1

3 Metagraph layout criteria

Some basic requirements for graphical representation of the graph are described in [3, 6-8, 14]. They are the minimum number of edge crossings, approximately equal edges length, the display of symmetries existing in the graph, etc. These requirements are also valid for metagraph. But correct location of metavertices and their shape are more important for metagraph drawing. Proposed layout criteria are described below.

Each vertex v_r and metavertex m_q has a position $p_{v_r} = (x_{v_r}, y_{v_r})$, $p_{m_q} = (x_{m_q}, y_{m_q})$. $P_V = (p_{v_1}, p_{v_2}, \ldots p_{v_{N_V}})$ is the vector of vertices positions, $P_M = (p_{m_1}, p_{m_2}, \ldots p_{m_{N_M}})$ is the vector of metavertices positions. These vectors fully determine the locations of nodes for visualization. The distance between nodes is calculated as distance between points: $l_{e_h} = \left\| p_{mv_i} - p_{mv_j} \right\|$, where p_{mv_i}, $p_{mv\,j}$ are the designation of the i-th and j-th nodes coordinates correspondently.

Definition 2. Graphical representation of the metavertex is the geometric figure F_{mq}, with position p_{m_q} that corresponds to the center of the figure.

Definition 3. Graphical representation of the edge is the curve F_{e_h} defined between the coordinates of the head and tail node.

Definition 4. Graphical representation of the metagraph is the set $W_S = \langle P_V, F_M, F_E \rangle$, where $F_M = \{F_{mq}\}$ is the set of metavertex figures, $F_E = \{F_{e_h}\}$ is the set of edges curves.

We consider graphical representation is correct, if mapping $S \to W_S$ is isomorphic. We propose metagraph layout criteria, which consists of three requirements:

1. Coordinates of vertices are not equal: $\forall i, \forall j : i \neq j \Rightarrow p_{v_i} \neq p_{v_j}$. Metavertices coordinates can be equal in the presence of common inner vertices.
2. The only inner vertices coordinates are located in metavertex figure F_{m_q}

   Metavertex figure F_{m_q} contains the only inner vertices of the metavertex m_q: $\forall v_r : v_r \in m_q \Rightarrow p_{v_r} \in F_{m_q}$, $\forall v_r : v_r \notin m_q \Rightarrow p_{v_r} \notin F_{m_q}$.
3. Metavertices figures without the common vertices do not intersect: $\forall j, \forall k : m_j \cap m_k = \varnothing \Rightarrow F_{m_j} \cap F_{m_k} = \varnothing$.

4 The method for visualization

Problem statement for metagraph visualization.

Given:

1. Metagraph $S = \langle V,M,E \rangle$.
2. P_V, P_M - start location of vertices and metavertices.
3. Metagraph layout criteria.
4. Rectangular region U. The result image should be placed in U.

Find: W_S.

To solve this problem it is necessary to determine: the placement of nodes that belong to metavertices and do not belong to them, location of metavertices that contain common vertices, relative position of vertices in metavertex figure, metavertex figures and curves F_{e_h}.

The proposed method is based on the Fruchterman and Reingold visualization algorithm [8]. Metagraph is represented as a system of objects, connected by springs according to certain rules. Each spring acts on the pair of nodes with the force of attraction or repulsion. Vertices move under the influence of the sum of these forces. The algorithm stops when the system reaches a point of equilibrium. The nodes are placed inside the rectangular area U.

With each iteration nodes move at a distance $\delta(t)$ in the direction of the total force acting on the node. Function $\delta(t)$ depends on the temperature and decreases. Repulsive force acts between each pair of nodes, except pairs metavertex and its internal vertex.

The attraction force acts between:

- adjacent nodes;
- metavertices and their inner vertices (this force provides the location of inner vertices in metavertex shape and vertices movement for the metavertex);
- all vertices in metavertex (this force provides a smaller area of the metavertex figure).

Then, the repulsive force induced by mv_i and acting on mv_j is defined as:

$$f_{rep}(i,j) = Kr_{ij} \frac{l^2}{\left\| p_{mv_i} - p_{mv_j} \right\|} p_{0_{ji}} \quad , \tag{1}$$

if $i = j$, $f_{rep}(i,j) = 0$.

Vector $p_{0_{ji}}$ is the direction vector from p_{mv_j} to p_{mv_i}. The optimal length l for an edge of metagraph is calculated as a function of allocation area and number of nodes:

$$l = \sqrt{\frac{area(U)}{N_V + N_M}} \, . \tag{2}$$

The attraction force induced by mv_i and acting on mv_j is defined as:

$$f_{attr}(i,j) = Ka_{ij} \frac{\left\| p_{mv_i} - p_{mv_j} \right\|^2}{l} p_{0_{ij}}, \tag{3}$$

if $i = j$, $f_{attr\,ij} = 0$.

Also the force of gravity attracts every node to the center of metagraph similar to the algorithm [7].

$$F_{gr}(i) = \left(1 + \frac{\deg(mv_i)}{2}\right) \cdot K_{grav} \frac{p_{mv_i} - B}{\left\| p_{mv_i} - B \right\|}, \tag{4}$$

where $B = \frac{1}{N_V + N_M} \cdot \sum p_{mv}$ - metagraph center,

$\deg(mv_i)$ - node degree or the number of edges connected to it,

$\left(p_{mv_i} - B\right)$ - the direction to the center of the metagraph.

Force of gravity keeps weakly connected nodes closer to whole metagraph. Its value depends on the node degree, so the nodes with many incoming or outgoing edges are located closer to the center. The presence of the force of gravity makes the overall arrangement more circular.

Then the total force acting on the node mv_i is calculated by the following formula:

$$F_{spring}(i) = \sum_j f_{rep}(i,j) + \sum_j f_{attr}(i,j) + F_{gr}(i).$$

The heuristics coefficients. The forces values depend on the coefficients Kr_{ij} and Ka_{ij}. The coefficients depend on the type of each node and their relationship in a pair mv_i and mv_j. They can be presented in two matrices: the attraction coefficient matrix and the repulsion coefficient matrix.

$$|Kr| = \begin{pmatrix} 0 & Kr_{12} & \cdots & Kr_{1n} \\ Kr_{21} & 0 & & \\ \vdots & & \ddots & \\ Kr_{n1} & \cdots & Kr_{nn-1} & 0 \end{pmatrix}, |Ka| = \begin{pmatrix} 0 & Ka_{12} & \cdots & Ka_{1n} \\ Ka_{21} & 0 & & \\ \vdots & & \ddots & \\ Ka_{n1} & \cdots & Ka_{nn-1} & 0 \end{pmatrix}.$$

Rows and columns of both matrices correspond to metagraph nodes. There are no repulsion force between pairs metavertex and inner vertex, so $Kr_{uv} = 0$.

Pairs $mv_i, mv_j \in m_q$ (inner vertex and inner vertex) have coefficient of repulsion:

$$Kr = Avg_{inner}.$$

Avg_{inner} is the average number of inner vertices in metavertices. The distances between inner vertices in metavertex should be approximately equal.

Other pairs have repulsion coefficient:

$$Kr_{ij} = \frac{W_{mv_i} + W_{mv_j}}{2} ,$$

where W_{mv_i} is the weight coefficient

$$W_{mv_i} = \begin{cases} 1 & , mv_i \in V \\ N_{m_q} & , mv_i = m_q \in M \end{cases} .$$

Thus, if some vertices mv_i and mv_j do not belong to any metavertex, then $Kr_{ij} = 1$. If mv_i or mv_j is the metavertex, repulsion force depends on its metavertex weight coefficient. Metavertex weight coefficient is a number of vertices included in it. If mv_i and mv_j are metavertices repulsion force depends on the average value of weight coefficients. Distance between metavertices with big number of inner vertices is larger.

If $mv_j \in M$ is the metavertex m_q and $mv_i \in m_q$ then attraction coefficient is defined by the formula:

$$Ka_{ij} = \frac{N_{m_q} + \deg(m_q) + 1}{1 + \frac{\deg(mv_i)}{2}} .$$

If $mv_i, mv_j \in m_q$ is the inner vertices for metavertex m_q:

$$Ka_{ij} = \frac{Avg_{inner}}{1 + \frac{\deg(mv_i)}{2}} .$$

If the some other nodes mv_i and mv_j are connected by an edge:

$$Ka_{ij} = \frac{W_{mv_i} + W_{mv_j}}{2 + \deg(mv_i)} .$$

If there is no edge between other nodes mv_i and mv_j : $Ka_{ij} = 0$.

Nodes with high degree move slower because of the expression $1 + \deg(mv_i)/2$ is in the denominator.

Coefficients matrices are normalized relative to the maximum value.

Therefore, elements in matrices less than or equal to one.

The algorithm of the method realization. The parameter ε is the accuracy of finding the equilibrium point; *stop* – the boolean parameter for determining the end of the algorithm, it is set as *true* in first iteration.

The proposed method for metagraph visualization consists of the following steps:

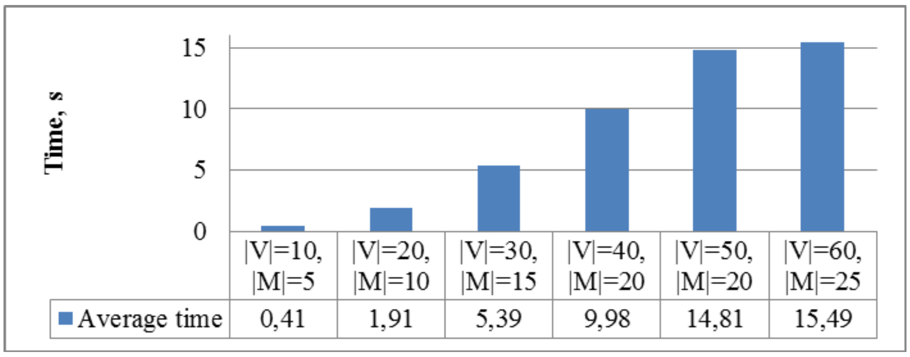
Step 1: Calculate the optimal edge length using the formula (2).
Step 2: Set the temperature *t* as maximum temperature.
Step 3: Calculate repulsion matrix and attraction matrix.
Step 4: Do steps 4.1 - 4.2 for each pair of nodes mv_i and mv_j
Step 4.1: Calculate the sum of attraction forces (1), repulsion forces (3) and gravity force (4).
Step 4.2: If $\left\|F_{spring}(u)\right\| > \varepsilon$ or reached the maximum iterations number, *stop = false*.
Step 5: If *stop = true* go to step 9.
Step 6: Do steps 6.1 - 6.4 for each node.
Step 6.1: Calculate new node position $p_{mv} = p_{mv} + \delta(t) \times \frac{F_{spring}(u)}{\left\|F_{spring}(u)\right\|}$.
Step 6.2: Prevent p_{mv} from being displaced outside frame *U*.
Step 7: Reduce the temperature.
Step 8: *stop = true*, go to step 4.
Step 9: Find F_{m_q} for each metavertex and set the metavertex position as the center of F_{m_q}.
Step 10: Find F_{e_h} for each edge.

In the suggested method the minimum convex hull was chosen as an optimal metavertices figure. To find convex hull Jarvis, Graham and Chan algorithms [5] can be used. The edges can be set as straight line connecting node shapes. In this case there are multiple intersections of edges and intersections metavertex figures. Therefore, it is necessary to calculate the shape of the edges as Bézier curve or B-spline [11, 13].

5 Experiment and results

The proposed method was tested on random metagraphs with up to 60 vertices and up to 25 metavertices and each metavertex had less than five intersections for any of its vertex. The tests were made on PC with Intel(R) Core(TM) i3 CPU 2x2.4 GHz, 3 Gb RAM.

It was observed that depending on the initial location of the nodes the result image was different. The time required to obtain a satisfactory image depends on the metavertices number and the number of inner vertices. The time is reduced with the improvement of the initial location of the nodes. Figure 3 illustrates the average time needed for random metagraph visualization.



	|V|=10, |M|=5	|V|=20, |M|=10	|V|=30, |M|=15	|V|=40, |M|=20	|V|=50, |M|=20	|V|=60, |M|=25
■ Average time	0,41	1,91	5,39	9,98	14,81	15,49

Fig. 3. Average time needed for random metagraph visualization

Visualization results for random metagraphs are presented in Figures 4, 5.

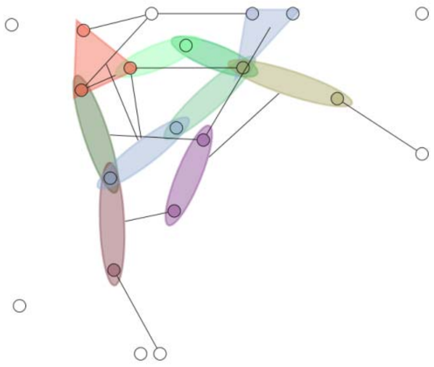


Fig. 4. Metagraph $S\langle |V| = 20, |M| = 10, |E| = 13 \rangle$ is drawn in 1.9s

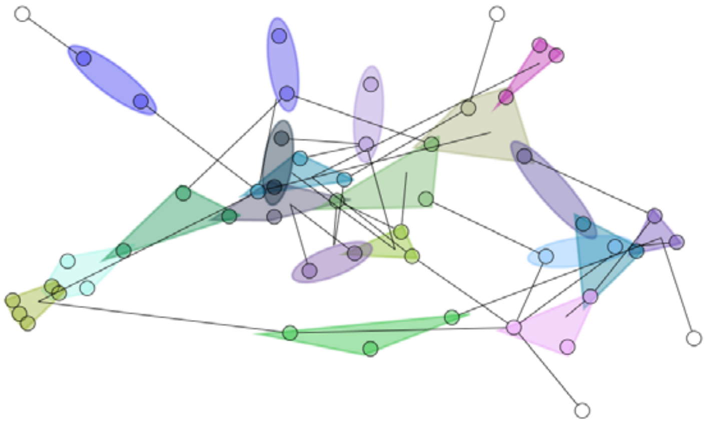


Fig. 5. Metagraph $S\langle |V| = 50, |M| = 20, |E| = 34 \rangle$ is drawn in 10.3s

6 Conclusion

The proposed method allows visualizing medium size metagraphs. Also there is a limit on the number of metavertices intersections. This can be explained by the fact that the Venn diagram analogue has been used. It could be difficult to discern metavertices if some of them are included in other and intersect. There were conducted several experiments that showed the usability of the proposed method.

In future revision of the method it is planned to work on the number of edges intersections minimization, a new model of multiple metavertex intersections determination, the occupation area minimization.

References

1. Archambault D., Munzner T., Auber D. TopoLayout: Multilevel Graph Layout by Topological Features: IEEE Trans Vis Comput Graph 13(2), 305-17 (2007)
2. Astanin S.V., Dragnish N.V., Zhukovsky N.K.: Nested metagraphs as models of complex objects. Inž. vestn. Dona, 4 (2012),http://www.ivdon.ru/magazine/archive/n4p2y2012/1434
3. Barnes J., Hut P.: A hierarchical O(N log N) force-calculation algorithm. Nature. 324(4), 446–449 (1986)
4. Basu A., Blanning R.W: Metagraph and Their Application. Springer. US, Integrated Series in Information Systems (2007)
5. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C.: Introduction to Algorithms (2nd. ed.). MIT Press and McGraw-Hill (2001)
6. Eades P.: A heuristic for graph drawing. Congressus Numerantium, 42 ,149–160 (1984)
7. Frick A., Ludwig A., Mehldau H.: A fast adaptive layout algorithm for undirected graphs. In Proc. Graph Drawing 1994. LNCS, vol. 894, pp. 389–403. Springer-Verlag (1995)
8. Fruchterman T. M. J., Reingold E. M.: Graph drawing byforce-directed placement. Software - Practice and Experience. 21(11), 1129–1164 (November 1991)
9. Hachul S., Junger M.: Drawing Large Graphs with a Potential-Field-Based Multilevel Algorithm (Extended Abstract). In Graph Drawing 2004. LNCS, vol. 3383, pp. 285-295. Springer-Verlag (2005)
10. Harel D., Koren Y.: A Fast Multi-scale Method for Drawing Large Graphs. In Graph Drawing 2000. LNCS, vol. 1984, pp. 183-196. Springer-Verlag (2001)
11. Holten D.: Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. IEEE Trans. Vis.Comput Graph. 12(5), 741–748 (2006)
12. Kamada T., Kawai S.: An algorithm for drawing general undirected graphs. InformationProcessing Letters, 31, 7-15 (1989)
13. Rogers D. F.: Procedural Elements for Computer Graphics (2nd. ed.). WCB/McGraw-Hill (1998)
14. Sugiyama K., Misue K.: Graph drawing by the magnetic spring model, J. Visual Languages Comput. 6 (3), 217- 232 (1995)
15. Walshaw C.: A multilevel algorithm for force-directed graph drawing. Journal of Graph Algorithms 7(3), 253–285 (2003)