Information Visualization Based on Attributed Hierarchical Graphs with Ports

Victor Kasyanov [1], Elena Kasyanova [1] and Timur Zolotuhin [1]

[1] A.P. Ershov Institute of Informatics Systems, Lavrentiev pr. 6, Novosibirsk, 630090, Russia

Abstract

Information visualization based on graph models has many applications in both real and theoretical fields. Since the information that it is desirable to visualize is constantly growing and becoming more complex, more powerful graph-theoretic formalisms are required and appear to represent structured information models with a hierarchical structure. One of these formalisms is the so-called attributed hierarchical graphs. It allows selecting in the original graph a set of such parts (so-called fragments) that all elements of each fragment deserve separate joint consideration, and all fragments of the selected set form a nesting hierarchy. Visual Graph is a visualization system which is constructed at the A.P. Ershov Institute of Informatics Systems to explore complex structured big data through their visual representations based on attributed hierarchical graphs. In many applications, objects modeled by graph vertices are complex and contain non-intersecting logical parts (so called ports) through which these objects are in a relationship modeled by arcs. In this paper a formalism of attributed hierarchical graphs with ports and new possibilities of the Visual Graph system for information visualization based on this graph formalism are considered.

Keywords

Attributed hierarchical graph, information visualization, graph model, port, visualization system, Visual Graph.

1. Introduction

Visualization of structured or relational data based on graph models has many applications in both real and theoretical fields [1, 2, 3]. Among them are physical communication networks and electrical networks, on the one hand, and compiler data structures and state transition diagrams, on the other. Therefore, at present, science-intensive software products using information visualization methods based on graph models (such as Cytoscape [4], Higres [5], Gephi [6], Graphviz [7], Tulip [8], yEd [9] and many others) are widely represented on the market.

Since the information that it is desirable to visualize is constantly growing and becoming more complex, more and more situations arise in which the organization of information is too complex to be modeled by a classical graph. More powerful graph-theoretic formalisms have been introduced to represent a hierarchical kind of diagramming objects, since hierarchy is the basis of numerous methods for visual processing of complex big data in various fields of applications [10, 11, 12]. One of these formalisms is the so-called hierarchical graphs and graph models [12]. The hierarchical graph model allows selecting in the given classical graph a set of such its parts (so-called fragments) that all elements of each selected fragment deserve separate joint consideration, and all fragments of the selected set form a nesting hierarchy. This fragment hierarchy provides a good tool for dealing with the complexity of visualizing structured data in applications that must deal with large amounts of complex structured data. It forms the basis for natural methods of "abstraction" and "reduction" that researchers can use to reduce the visual complexity of a large graph. The Visual Graph system has been constructed at the A.P. Ershov Institute of Informatics Systems by order of the Intel Company as a visualization system which is aimed to explore complex structured big data that occur in optimizing compilers through their visual representations based on hierarchical graph models [13].

GraphiCon 2022: 32nd International Conference on Computer Graphics and Vision, September 19-22, 2022, Ryazan State Radio Engineering University named after V.F. Utkin, Ryazan, Russia
EMAIL: kvn@iis.nsk.su (V. Kasyanov); kev@iis.nsk.su (E. Kasyanova); tzolotuhin@gmail.com (T. Zolotuhin)
ORCID: 0000-0002-4899-9429 (V. Kasyanov); 0000-0002-3412-0997 (E. Kasyanova); 0000-0002-9543-2353 (T. Zolotuhin)


In many applications, objects modeled by graph vertices are complex and contain non-intersecting logical parts through which these objects are in a relationship modeled by arcs. For example, when representing a data flow in a program in the form of a so-called information graph (or a program scheme with distributed memory), the graph vertices that model program statements are associated with sets of their so-called operands: information inputs (arguments) and information outputs (results) of statements [3]. Since data is exchanged between two program statements through their operands (from a result of the first statement to an argument of the second statement), information connections (data transfers) between program statements should be represented by directed arcs (edges) that connect the corresponding operands of the graph vertices. When representing graphs with vertices that model complex objects in existing graph description formats (see, for example, GraphML [14]), these different parts of complex objects are usually expressed using the so-called vertex ports, which in a graph drawing can be represented by different points of vertex images, in which the corresponding vertices are connected to the arcs incident to them.

In this paper a formalism of attributed hierarchical graphs with ports and new possibilities of the Visual Graph system for information visualization based on this graph formalism are considered. The rest of the paper is organized as follows. Section 2 presents a formalism of attributed hierarchical graphs with ports and ways of drawing such graphs on the plane. New possibilities of the Visual Graph system for visualization of large structured data based on attributed hierarchical graphs with ports are considered in Section 3. Section 4 is our conclusion.

2. Formalism of attributed hierarchical graphs with ports and ways of drawing such graphs on the plane

Let us recall some terms and notation from [3].

Let G be a graph of some type, for example, it can be an undirected (or directed) graph. The graph G is defined by two finite sets V and E, where the elements of V are the vertices of the graph G, and the elements of E are undirected (or directed) arcs of the graph G. G is a trivial graph if $|V|=1$ and $|E|=0$.

A graph C is called a fragment of the graph G and is denoted $C \subseteq G$ if C is a part of the graph G, i. e., it consists only of elements (vertices and arcs) of the graph G. A set of fragments F is called a hierarchy of nested fragments of the graph G if:

(1) F contains the graph G, and

(2) $C_1 \subseteq C_2$, $C_2 \subseteq C_1$, or $C_1 \cap C_2$ is empty for any two $C_1, C_2 \in F$.

For any two distinct fragments $C_1, C_2 \in F$, the fragment C_1 is immediately included in C_2 if $C_1 \subseteq C_2$ and there is no $C_3 \in F$ other than C_1 and C_2 such that $C_1 \subseteq C_3 \subseteq C_2$. A fragment $C \in F$ is elementary one if F does not contain a fragment immediately included in C.

A hierarchical graph $H=(G,T)$ consists of the graph G and a rooted tree T, which represents immediate inclusion relationship between elements of some hierarchy F of nested fragments of G. G is called the underlying graph of H. T is called the nesting tree of H.

A hierarchical graph H is called simple one if all of its fragments are generated subgraphs of the graph G (see Figure 1). It is easy to see that each cluster graph can be considered as a simple hierarchical graph $H=(G,T)$, such that G is an undirected graph, and the leaves of the tree T are exactly the trivial subgraphs of the graph G.

We define hierarchical graphs with ports as a subclass of hierarchical graphs as follows.

Let $H=(G,T)$ be a hierarchical graph and let $P \subseteq V$ be a set of so-called ports. A vertex $p \in P$ is called a port of a fragment $C \in F$ if C contains p and any fragment from F which is included in C does not contain p. It is easy to see that in any hierarchical graph $H=(G,T)$ with ports, each vertex $p \in P$ is a port of some fragment $C \in F$. Thus, the set of all vertices of a hierarchical graph H with ports splits into three non-intersecting sets: the set P of all ports of all fragments $C \in F$, the

set of all such vertices v of its underlying graph G that $v \notin P$, and the set F of all vertices of the nesting tree T.

For example, information connections between statements of a program can be represented as such a hierarchical graph $H = (G, T)$ with ports that the following two properties hold:

(1) G is a directed graph with $P = V$,

(2) each fragment $C \in F$ distinct from G is such an empty fragment (graph without arcs) that the set of its ports is divided into two disjoint subsets $A(C)$ and $R(C)$, such that there are no arcs outgoing from ports $A(C)$ or entering into ports $R(C)$.

So, the statements of the program are represented by the elementary fragments of the hierarchical graph $H = (G, T)$, the sets of arguments and results of a statement represented by any elementary fragment C are represented by the sets of ports $A(C)$ and $R(C)$, and all possible data transfers (information connections) between the results and arguments of the statements of the program are represented by the arcs of the underlying graph G.

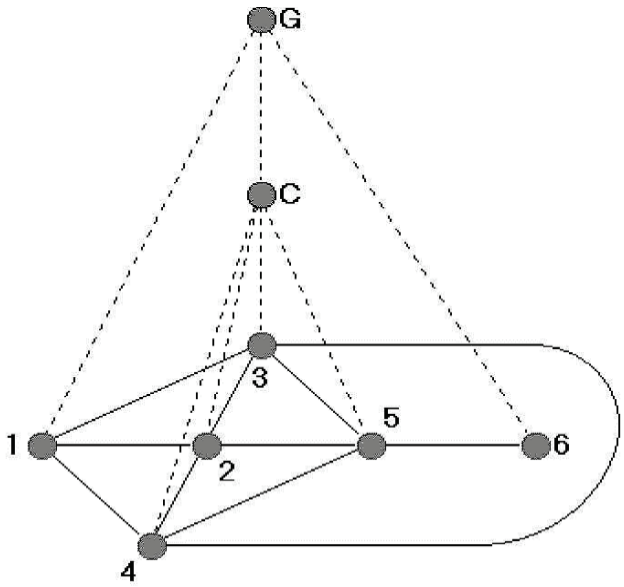

Figure 1: A simple hierarchical graph $H = (G, T)$ with two nontrivial fragments G and C

A drawing (or layout) of a hierarchical graph $H = (G, T)$ with ports is such a representation of its elements on the plane that the following properties hold.

1. Each vertex of the graph H is represented by some closed region (for example, a circle or a rectangle). The closed region is defined by its boundary (a simple closed curve in a plane) that divides the rest of the plane into two parts: an inner face and an outer face. In other words, the region representing the vertex of the graph H consists of its boundary and its inner face.

2. For any $C_1, C_2 \in F$ the intersection of regions of C_1 and C_2 is empty if and only if $C_1 \cap C_2$ is empty, and the region of each fragment $C \in F$ contain all points of region of any fragment that is included in the fragment C and all points of region of any port and vertex of the fragment C.

3. The region of any port and vertex of any fragment $C \in F$ does not contain points of regions of other port and vertex of the fragment C and points of regions of those fragments that are included in the fragment C.

4. Each arc of the graph G is represented by a simple curve (with an arrow if this arc is directed) connecting two points belonging to the boundaries of those two regions that represent the ports incident to this arc.

5. All arcs of any fragment $C \in F$ are represented inside the region of C.

Let a set W of objects called attributes be given, and let each element $w \in W$ be associated with a set $B(w)$ of objects called possible values of the attribute w. For example, a set of numbers, symbols or strings (sequences of symbols) can be used as the set $B(w)$. Let M denote the set of all such pairs (w,v) that $w \in W$ and $v \in B(w)$.

An attributed hierarchical graph with ports is a pair (H,L), where H is a hierarchical graph with ports, and L is an attribute function that associates each element (vertex, port, fragment and arc) h of the hierarchical graph H with such a subset $L(h) \subseteq M$ that $w_1 \neq w_2$ for any two $(w_1,v_1),(w_2,v_2) \in L(h)$.

When drawing an attributed hierarchical graph with ports, the attributes and their values for the elements of the hierarchical graph can either be expressed implicitly through certain properties of the way these elements are represented (for example, through the geometric shape of the vertex area, its dimensions, etc.), or be displayed explicitly in the form of a certain the type of marks of the corresponding elements. For example, an explicit representation of attributes can specify the location and appearance of attribute images as texts within vertex regions or near curves representing arcs.

3. New possibilities of the Visual Graph system for visualization of large structured data based on attributed hierarchical graphs with ports

The Visual Graph system [13] is intended mainly for visualization of complex structured data that occur in optimizing compilers (or in other program design systems), based on attributed hierarchical graphs of a sufficiently large size, and in the current version also with ports (see Figure 2). The system makes a multi-aspect layout of an input graph model that consists of separate drawings of only such its fragments which have been interested for user during graph model consideration and constructed on demand. The Visual Graph system provides also rich opportunities to navigate through big graph model, to make its structural analysis and to work with the attributes of its elements, as well as to extend and customize easily the system to specific needs of a concrete user.

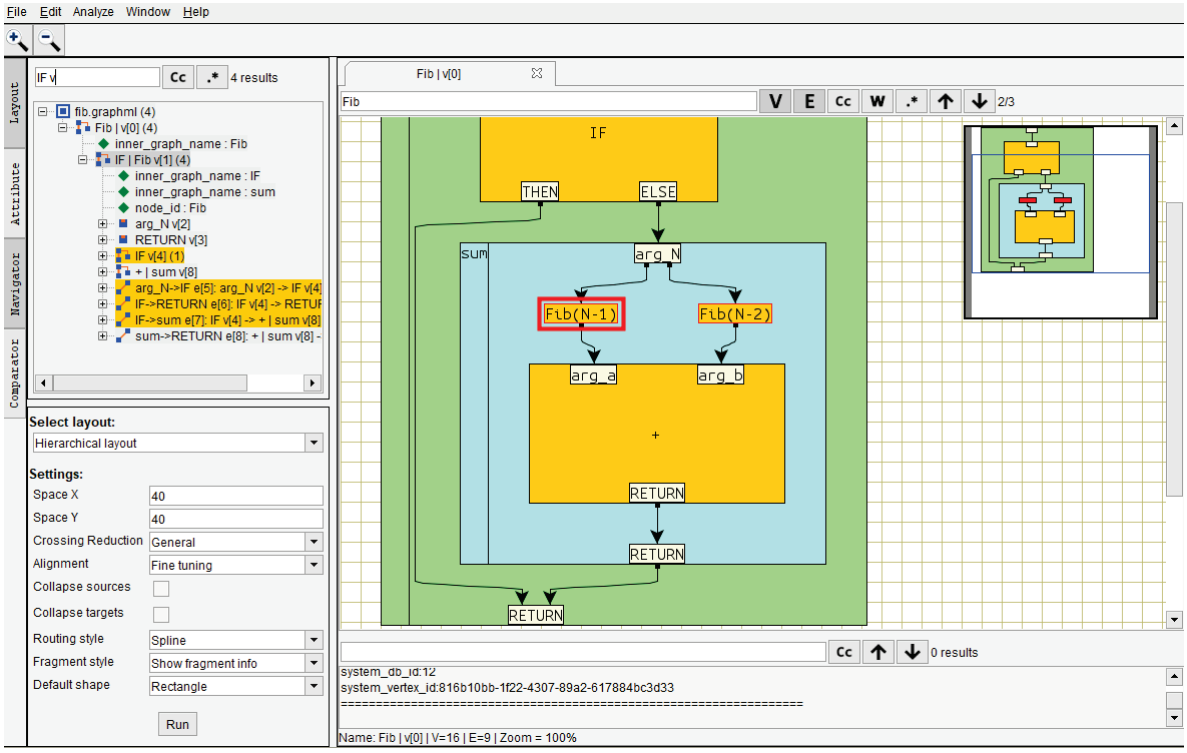

Figure 2: The Visual Graph system

It is assumed that the Visual Graph system is used as follows (see Figure 3). First, a compiler (or another program design system) itself or with an auxiliary program transforms a graph model arisen during compiling a source program from its internal representation into a file of one of the formats supported by the Visual Graph system, usually into the GraphML-file. Then the Visual Graph system will be able to read this graph model from the file, to visualize it and to provide a user with different navigation tools for its visual exploration allowing him to take the optimal decisions about compiler behavior.

In the current version of the Visual Graph system, the set of built-in algorithms for drawing graphs has changed due to its expansion with two algorithms: the algorithm for circular drawing of attributed hierarchical graphs with ports [15], and the algorithm for layer-by-level drawing of attributed hierarchical graphs with ports, being internal representations of functional programs [16]. For this purpose, JGraph library [17], which was originally used in the Visual Graph system for displaying and drawing graphs, has been replaced in the current version of the system with a specially developed module for displaying graph models. The fact is that the JGraph library has a number of limitations that made it difficult to fully implement work with hierarchical attributed graphs with ports. The main difficulties in its use are problems connected with the significantly increased number of graph elements due to ports, as well as with the required different ways of displaying such graph elements as ports, vertices and fragments.

There have also been changes to the user interface of the Visual Graph system, which still includes the desktop, the navigation bar, and the attribute bar.

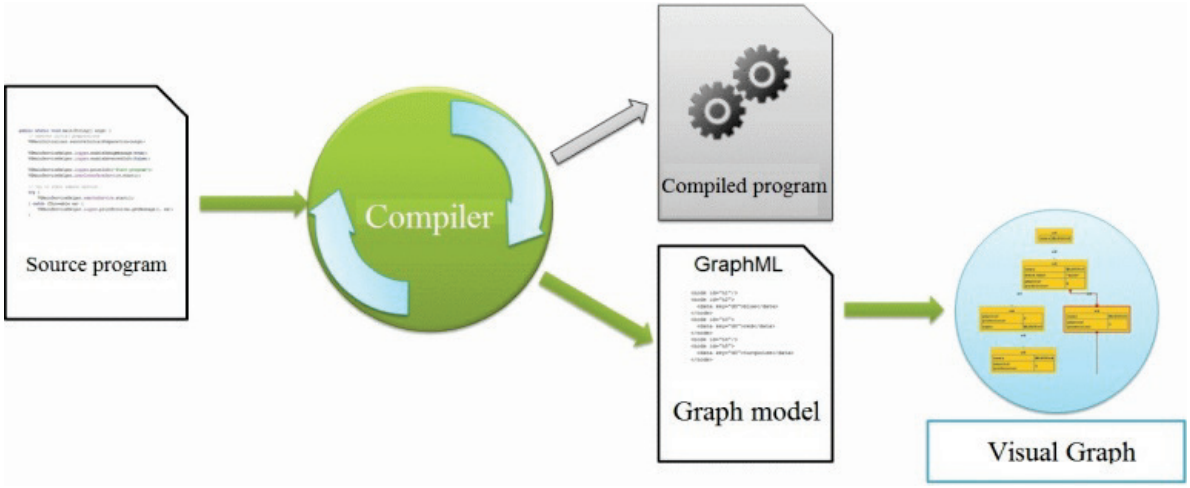

Figure 3: Application of the Visual Graph system

The desktop consists of a set of tabs opened by the user for visually represent selected fragments of a hierarchical graphical model in the form of their layouts on a plane, constructed using the drawing algorithms built into the system and analyzed using structural analysis tools. Structural analysis tools include various built-in graph model processing algorithms that help a user to select and visualize the structural information he needs in graph models. These include, for example, tools such as finding the shortest path, cycles, or strong predecessors in a graph, and finding the maximum common subgraph of two graphs. Each of these drawing algorithms and structural analysis algorithms has its own set of parameters that allow a user to control both the type of the graph model layout obtained as a result of its work and the process of its construction. To improve the layout obtained automatically, a user can change the shape of the vertices and arcs, the displayed attributes, the scale of the visible area and much more, and now he can use the grid when making these changes. Each tab has its own search string (filter), using which you can find and select those elements of the visible part of the graph model, the text attributes of which satisfy the regular expression that the user entered in the search string. Filter options have been expanded for its customizing what elements to search for. So, for example, you can search only among vertices or arcs. The mini-map allows a user to view the entire graph together with the selected part of it that is visible in the current tab, and also helps the user move and scale the visible

part of it. In the current version of the system, the mini-map was removed from the toolbar and became the upper right part of the current tab, and the ability to turn it off (not show) was added. In particular, this made it possible to increase the area of the current tab vertically, which is often useful during consideration of the constructed image of a graph fragment.

The navigation bar is a tool for visualizing all the graphs that a user is currently working with as an image using the indentation of the nesting trees of fragments of these graphs. Operations of folding and expanding images of nesting subtrees are supported, as well as highlighting fragments of interest to the user and opening them in new tabs. For a quick search through the trees, the search string has been improved, which allows the user to easily find the fragments of interest to him using regular expressions. Differences from the original version are the flexibility of specifying search conditions, as well as improved work with attributes containing a large amount of information.

The attribute panel is a tool that allows a user to control the visualization of attributes for selected vertices and arcs in the current tab. To do this, the user needs to select those vertices and arcs that are of interest to him in the graph from the current tab, and then tick the attributes that he wants to visualize for these elements in the attribute panel. The user can also use this tool to set a set of attributes that will be rendered for elements opened in a new tab. The attribute panel has been moved from the lower section of the user interface to the section on the left, which has increased the viewable area of the graph image. Attribute values have been removed from the attribute panel and moved to the graph image area, and now, when the user selects elements of the graph image, information about the selected elements automatically appears. This information is presented in the form of solid text, which can be searched, as well as copied and transferred to other third-party utilities. The user can always hide this menu if he wants.

The set of formats for representing graphs supported by the system has also been expanded. Now it is not only the standard graph description language GraphML, but also other widely used graph formats such as DOT [18] and GML [19].

4. Conclusion

The paper introduces the formalism of attributed hierarchical graphs with ports, and presents new possibilities of the Visual Graph system for visualization of large structured data based on this formalism. The system makes a multi-aspect layout of an input graph model with ports that consists of separate drawings of only such its fragments which have been interested for user during graph model consideration and constructed on demand. The Visual Graph system provides also rich opportunities to navigate through big graph model with ports, to make its structural analysis and to work with the attributes of its elements, as well as to extend the system and customize it to specific needs of a concrete user.

In the future, we plan to develop the Visual Graph system mainly in two directions: by expanding its set of built-in algorithms to include algorithms for drawing and structural analysis of attributed hierarchical graphs with ports, which are interesting not only for program design systems, but also for other applications, and also by giving the system the ability to provide corresponding web service.

5. Acknowledgements

This work was carried out under state contract with IIS SB RAS (FWNU-2021-0002).

6. References

[1] G. Di Battista, P. Eades, R. Tamassia, et all, Graph Drawing: Algorithms for Visualization of Graphs, Prentice Hall, 1999.

[2] I. Herman, G. Melançon, M. S. Marshall, Graph visualization and navigation in information visualization: a survey, IEEE Transactions on Visualization and Computer Graphics 6 (2000) 24-43. doi: 10.1109/2945.841119.

[3] V. N. Kasyanov, V. A. Evstigneev, Graphs in Programming: Processing, Visualization and Application. St. Petersburg, BHV-Petersburg, 2003. (In Russian).
[4] Cytoscape. URL: https://cytoscape.org.
[5] I. A. Lisitsyn, V. N. Kasyanov, Higres - visualization system for clustered graphs and graph algorithms, in: J. Kratochvíyl (Ed.) Graph Drawing, 7th International Symposium, GD 1999, volume 1731 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, 1999, pp. 82-89. doi: 10.1007/3-540-46648-7_8.
[6] Gephi. URL: https://gephi.org.
[7] Graphviz. URL: https://graphviz.org.
[8] Tulip, URL: https://tulip.labri.fr.
[9] yEd. URL: https://www.yworks.com/products/yed.
[10] Q. W. Feng, R. F. Cohen, P. Eades, Planarity for clustered graphs, in: P. Spirakis (Ed.) Algorithms - ESA 1995, Third Annual European Symposium, volume 979 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, 1995, pp. 213-226. doi:10.1007/3-540-60313-1_145.
[11] K. Sugiyama, K. Misue, Visualization of structured digraphs, IEEE Transactions on Systems, Man and Cybernetics 21.4 (1999) 876-892. doi: 10.1109/21.108304.
[12] V. N. Kasyanov, E. V. Kasyanova, Information visualization based on graph models, Enterprise Information Systems 7.2 (2013) 187-197. doi: 10.1080/17517575.2012.743188.
[13] V. N. Kasyanov, T. A. Zolotuhin, A system for visualization of big attributed hierarchical graphs, International Journal of Computer Networks & Communications (IJCNC) 10.2 (2018) 55-67. doi: 10.5121/ijcnc.2018.10206.
[14] The GraphML File Format, URL: http://graphml.graphdrawing.org.
[15] V. N. Kasyanov, A. M. Merculov, T. A. Zolotuhin, A circular layout algorithm for attributed hierarchical graphs with ports, Journal of Physics: Conference Series 2099 (2021) 012051. doi:10.1088/1742-6596/2099/1/012051.
[16] V. N. Kasyanov, T. A. Zolotuhin, D. S. Gordeev, Visualization methods and algorithms for graph representation of functional programs, Programming and Computer Software 45.4 (2019) 156-162. doi: 10.1134/S0361768819040030.
[17] JGraph. URL: http://dev.cs.ovgu.de/java/jgraph/tutorial/t1.html.
[18] DOT. URL: http://www.graphviz.org/doc/info/lang.html.
[19] GML. URL: http://openmis.ru/doc/clang/gml-tr.html.