Higres – Visualization System for Clustered Graphs and Graph Algorithms*

Ivan A. Lisitsyn and Victor N. Kasyanov

A. P. Ershov's Institute of Informatics Systems,
Lavrentiev av. 6, 630090, Novosibirsk, Russia
lis@iis.nsk.su

Abstract. We present the Higres system – a visualization tool, an editor for clustered graphs and a platform for execution and animation of graph algorithms. Higres can handle any attributed graphs and perform animated semantic processing of them. The semantics of a graph can be defined by the user in the graph editor. The user can also create new external modules to process graphs with defined semantics. In addition we provide a possibility to extend system with new graph drawing algorithms by using special type of external modules for this purpose. We provide an easy to use C++ library for implementation of both types of extensions.

1 Introduction

Graph models can be used in practice only along with support tools that provide visualization, editing and processing of graphs. For this reason many graph visualization systems, graph editors and libraries of graph algorithms have been developed in recent years. Examples of these tools include VCG [1], daVinci [2], Graphlet [3], GraVis [4], GLT & GET [5] and LEDA [6].

In some application areas the information is organized too complex to be modeled by a classical graph. In this case other graph-like structures should be taken in consideration. Hence, there is a need of tools that are capable of visualization of such structures. For example, GLT & GET can visualize graphs, vertices of which are associated with other graphs; D-ABDUCTOR [7] is a system designed for visualization of compound graphs.

One of recent graph formalisms is the *clustered graphs* . A clustered graph consists of a classical graph and a recursive clustering structure that partitioning vertices of this graph into hierarchically enclosed fragments. Clustered graphs are used in many areas where strong structuring of information is needed. Some theoretical results on clustered graphs with application to graph drawing are presented in [8,9,10,11]. A system that provides some kind of animated visualization and force-based allocation of clustered graphs is presented in [13].

Several general purpose graph visualization systems provide recursive folding of subgraphs allowing to create clusters. However, this feature is commonly used

* Supported by RFBR Grant N 98-01-748

J. Kratochvíl (Ed.): GD'99, LNCS 1731, pp. 82–89, 1999.

only to hide a part of information and not always applicable to the visualization of clustered graphs.

Usual graph editors either do not have support for attributed graphs or have rather weak support. Though the GML file format used by Graphlet can store arbitrary number of labels associated with each graph element, it is impossible to edit and visualize these labels in Graphlet's graph editor. Most editors allow only one text label for each vertex and optionally for each edge.

In this paper we present the Higres system, which is a visualization tool, an editor for attributed clustered graphs and a platform for execution and animation of graph algorithms. Proper support for attributed graphs allows us to define the semantics of a graph inside the graph editor. This feature can be particularly useful in conjunction with animation of graph algorithms, since in this case we can animate any kind of semantic processing.

Algorithms executed by Higres are implemented in external modules. The user can create his/her own modules with algorithms that process graphs with user defined semantics. In addition we provide special type of integration for graph drawing algorithms. Hence, the system can be extended with two types of modules. First type provides animation capabilities whereas second type is easier to use (user only needs to open a module and wait for the result). A new graph drawing method can be developed and tested with help of a module of the first type, then used in practice as a "quick" module without animation. We provide a special C++ library for implementation of both types of extension modules.

Higres is implemented in C++ with MSVC compiler and MFC library. The system works under MS Windows 95/98 and Windows NT.

2 Clustered Graphs in Higres

A clustered graph consists of vertices, fragments and edges, which we call objects. Vertices and edges form a classical graph that can be either directed or undirected. We allow multiple edges and loops. Each fragment is associated with a set of vertices. We say that these vertices belong to the fragment. If the set of vertices belonging to a fragment intersects with the set of vertices of another fragment, then either first fragment is a subfragment of second one or vise versa. The set of all vertices forms the *main fragment* of a clustered graph.

The semantics of a clustered graph is represented in Higres by means of object types. Each object in a graph belongs to an object type. A set of labels is defined for each object type. Each label has its data type, name and several other parameters. A set of values is associated with each object according to the set of labels defined for the object type to which this object belongs. These values along with partitioning of objects into types represent the semantics of a graph. Thus, user can define the semantics by creating new object types and adding new labels.

3 Visualization

In Higres each fragment is represented by a rectangle. All vertices that belong to a fragment and all its subfragments are located inside its rectangle. Fragments as well as vertices never overlap each other. Each fragment can be either closed or open. When a fragment is open, its content is visible to the user; when it is closed, it is drawn as an empty rectangle with only labels text inside it. A separate window can be opened to observe each fragment. Only the content of a fragment is shown inside its window, though it is possible to see this content inside windows of parent fragments, if the fragment is open. Fig. 1 shows a clustered graph with open and closed fragments.

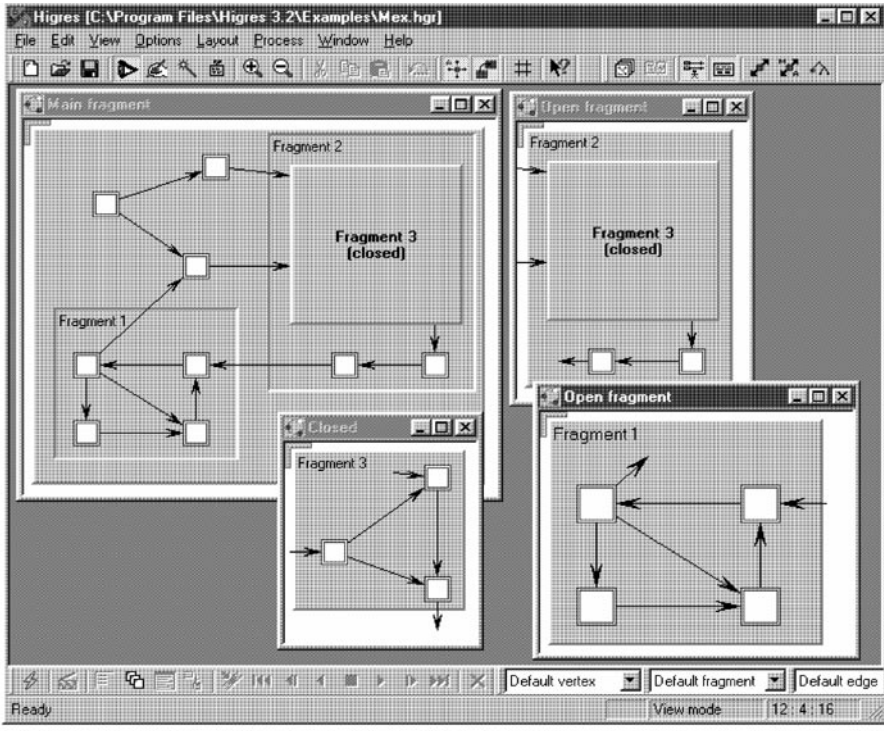

Fig. 1. Higres working with a clustered graph.

Most part of visual attributes of an object is defined by its type. This means that semantically related objects have similar visual representation. Higres uses flexible technique to visualize object labels. The user specifies a text template for each object type. This template is used to create labels text of objects of the given type by inserting in it labels values of a particular object.

Other visualization features include the following:
- various shapes and styles for vertices;
- polyline and smooth curved edges;
- various styles for edge lines and arrows;
- color selection for all graph components;
- possibility to scale graph image to arbitrary size;
- edge text movable along the edge line;
- external vertex text movable around the vertex;
- font selection for labels text;
- two graphical output formats;
- a number of options that control graph visualization.

Currently Higres uses three graph drawing algorithms for automatic graph allocation. First one is a force method, which is very close to original algorithm from [12]. Second one is our improvement of the first. Third method allocates rooted trees on layers. These algorithms are implemented in external modules.

4 User Interface

A comfortable and intuitive user interface was one of our main objectives in developing Higres.

The system's main window contains a several toolbars that provide quick access to frequently used menu commands, and object types switching for creation of new objects. The status bar displays menu and toolbars hints along with other useful information on current editor operation and graph processing status.

The system uses two basic modes: view and edit. In the view mode it is only possible to open/close fragments and fragment windows, but the scrolling operations are extended with mouse scrolling.

In the edit mode the left mouse button is used to select objects and the right mouse button displays the popup menu, in which the user can choose the operation he/she wants to perform with the selected objects or with whole graph. It is also possible to create new objects by selecting commands from that menu (fig.2).

The left mouse button can be also used to move vertices, fragments, labels texts and edge bends, and resize vertices and fragments. Thus, all edit operations are gathered in a single mode. To our opinion it is more useful approach (especially for inexperienced users), than division to several modes. However, for adherents of the latter we provide two additional modes. Their usage is optional but in some cases they may be useful: creation mode for object creation and labels mode for labels editing.

Other interface features include the following:
- cut/copy/paste operations available for any graph parts;
- almost unlimited number of undo levels;
- optimized screen update;
- automatic elimination of objects overlapping;
- automatic vertex size adjusting;

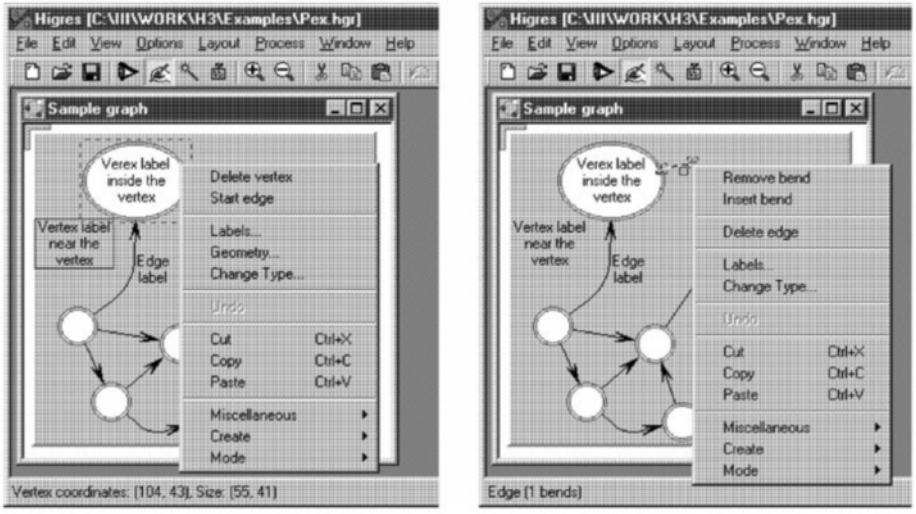

Fig. 2. Popup menus in the edit mode: a). menu for a vertex; b). menu for an edge.

- grid with several parameters;
- a number of options that configure user interface;
- online help available for each menu, dialog box and editor mode.

5 Algorithms Animation

To run an algorithm the user should select an external module in the dialog box. The system starts this module and optionally opens the process protocol window for textual process output. Higres provides both run-time animation of algorithms and sample caching for repeated and backward animation.

A set of parameters can be defined inside a module. These parameters can be changed by the user at any execution step. The module can input text and numbers from the user. It can also write any textual information to the protocol window.

A wide range of semantic and graph drawing algorithms can be implemented as external modules. Screen shots on fig. 3,4,5 give some examples of semantic algorithms executed on different graph models.

The animation feature can be used for algorithm testing and debugging, educational purposes and exploration of iteration processes such as force methods in graph drawing.

We provide a special C++ library that can be used to create external modules. This library includes functions for graph modification and ones that provide interaction with the system. It is unnecessary for programmer, who use this library, to know the details of the internal representation of graphs in Higres. This approach greatly simplifies the creation of new modules.

6 Conclusion

We presented the Higres system, which is a visualization tool, an editor for attributed clustered graphs and a platform for execution and animation of graph algorithms. The system is fully usable and has applications to education, research and practice. Our future plans in improving and extending the system concern the integration of additional graph drawing algorithms, including methods specially designed for allocation of clustered graphs.

The system is available on WWW at http://lis.iis.nsk.su/higres.

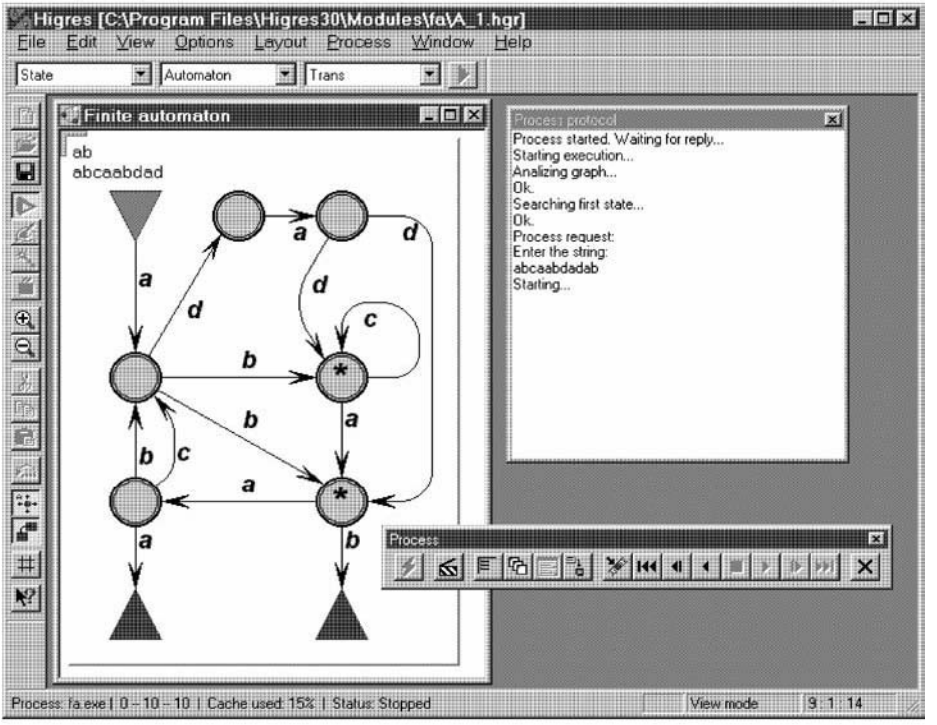

Fig. 3. The graph of a finite automaton. The external module emulates the work of this automaton. Asterisks show active states. Two lines in the top left corner are the tail of the string entered by the user and the part of that string that is already processed.

References

1. G. Sander. Graph layout through the VCG tool. Proc. of Graph Drawing '94, volume 894 of LNCS, pages 194-205, 1995.

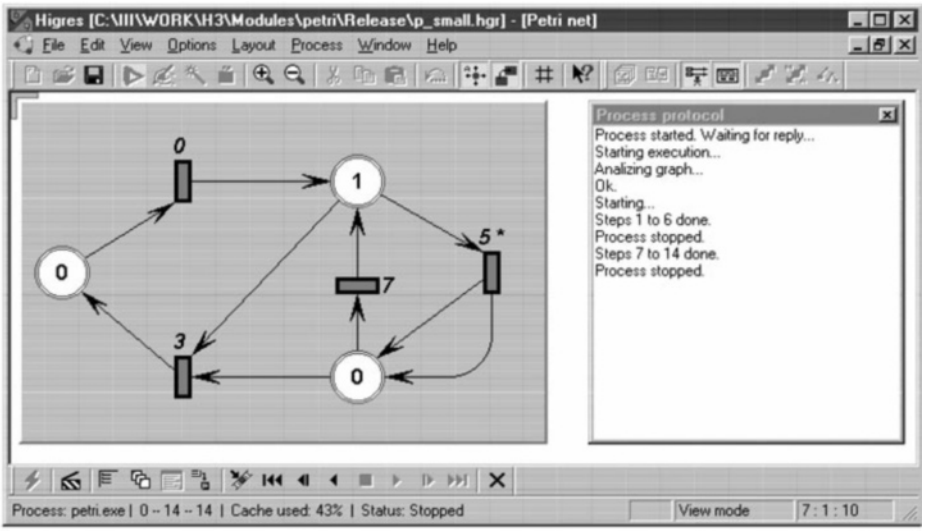

Fig. 4. A Petri net. The external module performs its simulation. Asterisk shows the transition that will fire on the next step. Numbers inside places are numbers of tokens. Numbers near transitions are their priorities introduced in order to make this Petri net determinate.

2. M. Fröhlich and M. Werner. Demonstration of the interactive graph visualization system daVinci. Proc. of Graph Drawing '94, volume 894 of LNCS, pages 266-269, 1995.
3. M. Himsolt. The Graphlet System (system demonstration). Proc. of Graph Drawing '96, volume 1190 of LNCS, pages 233-240, 1997.
4. H. Lauer, M. Ettrich and K. Soukup. GraVis – System Demonstration. Proc. of Graph Drawing '97, volume 1353 of LNCS, pages 344-349, 1997.
5. B. Madden, P. Madden, S. Powers and M. Himsolt. Portable Graph Layout and Editing. Proc. of Graph Drawing '95, volume 1027 of LNCS, pages 385-395, 1996.
6. K. Mehlhorn and S. Näher. LEDA: a platform for combinatorial and geometric computing. Commun. ACM, 38:96-102, 1995.
7. K. Siguyama and K. Misue. A Generic Compound Graph Visualizer/Manipulator: D-ABDUCTOR. Proc. of Graph Drawing 95, volume 1027 of LNCS, pages 500-503, 1996.
8. P. Eades, Q.W. Feng. Drawing Clustered Graphs on an Orthogonal Grid. Proc. of Graph Drawing 97, volume 1353 of LNCS, pages 182-193, 1997.
9. P. Eades, Q.W. Feng. Multilevel visualization of Clustered Graphs. Proc. of Graph Drawing 96, volume 1190 of LNCS, pages 101-112, 1997.
10. P. Eades, Q.W. Feng and X. Lin. Straight-Line Drawing Algorithms for Hierarchical Graphs and Clustered Graphs. Proc. of Graph Drawing 96, volume 1190 of LNCS, pages 113-128, 1997.
11. Q.W. Feng, R. Cohen and P. Eades. How to draw a planar clustered graph. In COCOON'95, volume 959 of LNCS, pages 21-31, 1995.
12. P. Eades. A Heuristic For Graph Drawing. Congressus Numerantium, 42, pages 149-160, 1984.

13. M.L. Huang, P. Eades. A Fully Animated Interactive System for Clustering and Navigating Huge Graphs. Proc. of Graph Drawing 98, volume 1547 of LNCS, pages 374-383, 1998.

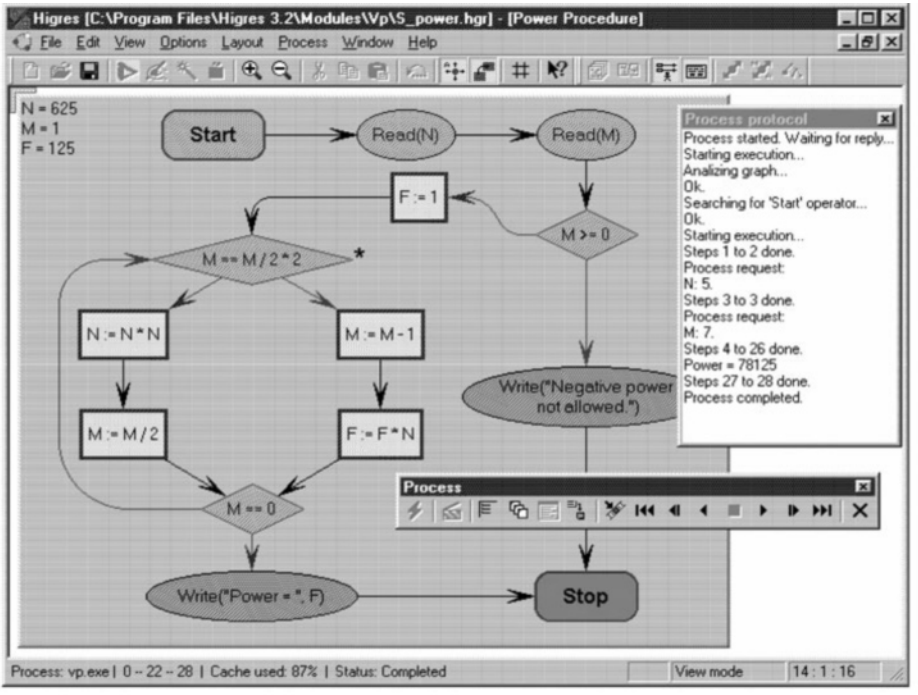

Fig. 5. Program scheme interpreted by external module. Asterisk shows which operator has control. Current values of variables are listed in the top left corner of the fragment rectangle. This screen shot was made after the completion of the process and rewinding several samples back.