BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **5**, No 2

Sofia . 2005

Applications

A Software System for Visual Processing of Graph Data

Ivan Moustakerov

Institute of Information Technologies, 1113 Sofia

Abstract: *There exist a large number of combinatorial problems defined on graphs (or networks). Many efficient algorithms for both directed and undirected graphs problems solving are known and all of them need input of the graph structure data. The current paper describes a software system for visual processing of graph data for the purposes of combinatorial graph problems solving. It can be used as a tool for teaching or learning the work of the combinatorial graph algorithms and for solving of practical graph problems.*

Keywords: *software system, Java-application, graph data, visual editing, shortest paths, Dijkstra's algorithm.*

I. Introduction

There exist a large number of combinatorial problems defined on graphs (or networks). For example a fundamental problem in network optimization is calculating of the shortest paths. Many efficient algorithms for both directed and undirected graphs tasks are known and all of them need input of the graph structure data. Specific visual representation of graphs naturally asks for visual graph data entering and processing. A graphical approach to creating graphs or networks and visual editing, repositioning and modifying of the corresponding data could be useful for all researchers, students, specialists, etc., dealing with different graph problems solving.

The current paper describes a software system for visual processing of graph data for the purposes of combinatorial graph problems solving. It can be used for creating and editing of graphs with edge costs (distances or weights). Using simple intuitive interface and mouse buttons, the user can create any kind of directed or undirected graph up to 999 vertices and unlimited connecting edges. The limitation of vertices was chosen because of typical computer screen size and from the size of most practical graph problems. That limitation is not restrictive one and the system could be modified to any practical need if such exists. The graphs elements – vertices,

edges, edges costs, source and destination vertices, can be visually (using mouse buttons) chosen, moved, deleted, inserted with correspondent graph data structure modifying. The created graphs can be saved as data files for loading, editing and subsequent processing or as pictures (PNG-format) for illustration purposes.

The described software system is programmed as Java-application and is (at least theoretically) independent of the computer platform. On its base different combinatorial graph algorithms can be implemented and different practical graph problems solver systems could be created.

As variant of the system for finding the shortest path for a single-source-destination problem with nonnegative edges costs using Dijkstra's algorithm will be shown here as an illustration for practical usage .

II. Functional Purposes

The main functional purposes of the system are:

- to serve as tool for visual (graphical) creating of any directional or unidirectional graph and for visual processing of its structure (editing and modifying – deleting, inserting, moving of its elements, scaling for better viewing, inserting boxes and explanatory texts, undoing last operation and allowing other operations known for graphical data processing and making sense for graphs creating,
- to save the graph structure as a system data file for subsequent loading and using,
- to save the current graph presentation as a image (in PNG-file format for using as illustration in a report or paper for example),
- to solve relevant combinatorial graph problems by using implemented different combinatorial graph algorithms (finding of the shortest path for example),
- to show visually working steps of the implemented algorithms for educational purposes.

It should be pointed out that main attention was paid to editing functionality of the system and to visualizing of the algorithm work. There exist many graph algorithms ready to be implemented. The main goal is using of the system from operations research teachers and students as a tool for using and studying the existing known combinatorial graph algorithms. To serve that goal a number of different known algorithms will be implemented gradually as the system evolves. It could be used also from researchers for fast visual creating of graphs and solving combinatorial graph problems.

III. System's menus and user interface

The user interface is traditional – using standard windows menus and buttons for mouse clicking. The used graphical pictograms for buttons were chosen to be self-explanatory and intuitive. The system windows also have standard look as known from mass operating systems (Windows and Linux for example). The interface look could be changed using Java possibilities for specific "look and feel" interface. Although not shown on black and white examples of the paper magazine copies, colors are widely used for clear pointing of the specific graph elements – source node is blue, destination node and the shortest path are red, etc.

The system main window with a graph example and shortest path found from node 1 to node 7, is shown on Fig.1. The buttons on the tool bar are mostly standard and will be explained later. The main drawing panel is scrollable when big graphs should be drawn and shown. The text panel on the left is for graph data structure in text format. It is automatically loaded and updated with each change of the graph on the drawing panel and is not editable in this version of the system. At the bottom of the window a status line is provided for system messages – for example on Fig. 1 it shows a message for shortest path found. The shortest path is drawn in red (visible as light grey on black and white Fig. 1), source node is marked with blue and destination node with red color. Next to the tool bar a text line for entering some explanatory texts, titles, etc., is provided.

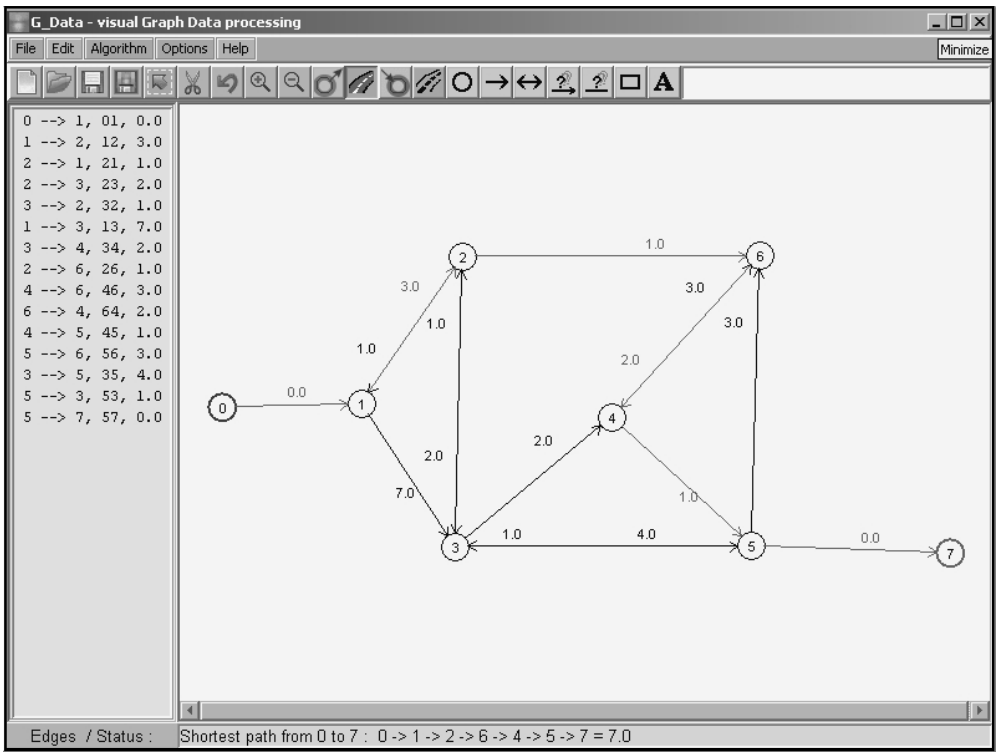

Fig. 1. Main window

The main functions of the system are activated by the following menus and toolbar buttons (Figs. 2-7).

Menu **"File"** is shown in Fig. 2.

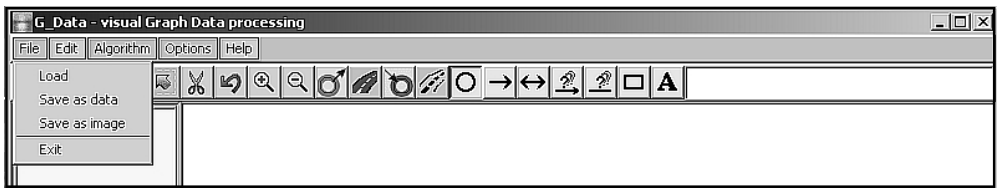

Fig. 2

• ***Load:*** Loads a data file in system format that contains graph structure data of a previously created by the system graph. The corresponding toolbar button is  .

• ***Save as data:*** Saves the created graph in system's data format. The user names the file and the system automatically adds the filename extension "*.grph*". *The corresponding toolbar button is*  .

• ***Save as image:*** Saves the graph on the display area as image *png*-file. The user gives the filename and the system automatically adds the filename extension "*.png*". The corresponding toolbar button is  .

• ***Exit:*** Quits the session and exit from the system.

Menu **"Edit"** is presented in Fig 3.

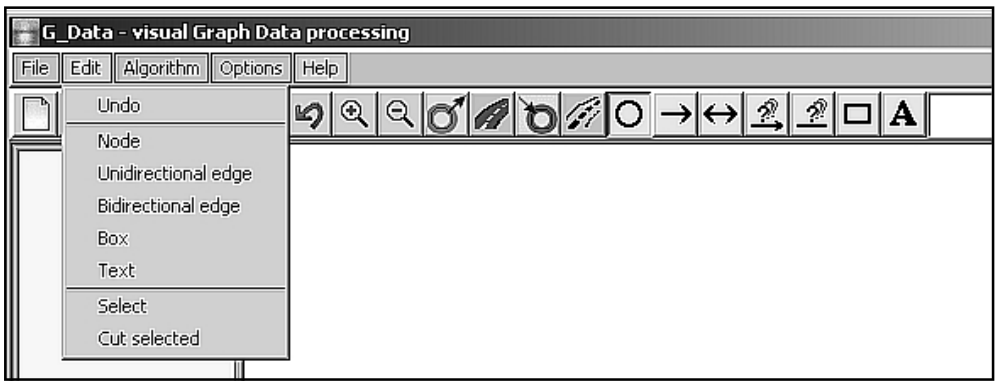

Fig. 3

• ***Undo:*** Reverts back the last operation repeatedly until that action makes sense. The corresponding toolbar button is  .

• ***Node:*** Inserts a node with fixed size and automatically numbers it in the center of the node (see Fig. 1), starting with 0 up to 999. The limit of 999 comes from fixed diameter of the node – if the practical need for number of nodes greater than 999 exists that option of the system will be fixed as needed. The corresponding toolbar button is  .

• ***Unidirectional edge:*** Inserts unidirectional edge with automatically positioning the edge on the line connecting both nodes centers and the both edge ends on the nodes circles and automatically drawing of the ending arrow at the ending node. It is impossible to create edge starting or ending from or to nowhere – just between different existing nodes. When the edge is valid the system asks (by pop-up window – Fig. 4) for cost of that edge – it is also impossible to accept edge without cost. Only positive numbers are accepted without theoretical upper limit. Clicking on button "NO" means that edge will not be created or in case of modification of the cost, the old cost will remain. The accepted cost is automatically position near the ending arrow of the edge.

Fig. 4

The corresponding toolbar button is  .

• ***Bidirectional edge:*** Same as for unidirectional edge with the difference that ending arrow is not drawn. That kind of edges are considered as passed in both directions with equal cost in both directions. When a bidirectional edge with different costs in different directions is needed it is created by inserting two unidirectional edges with different starting and ending node. That kind of edge looks as a single edge with arrows on both ends and two cost numbers around the both arrows. (for example see the edge 1<—>2 in Fig. 1). The corresponding toolbar button is  .

• ***Box:*** Allows box drawing with mouse dragging, mainly for some visual aesthetic purposes – Fig. 5. The corresponding toolbar button is  .

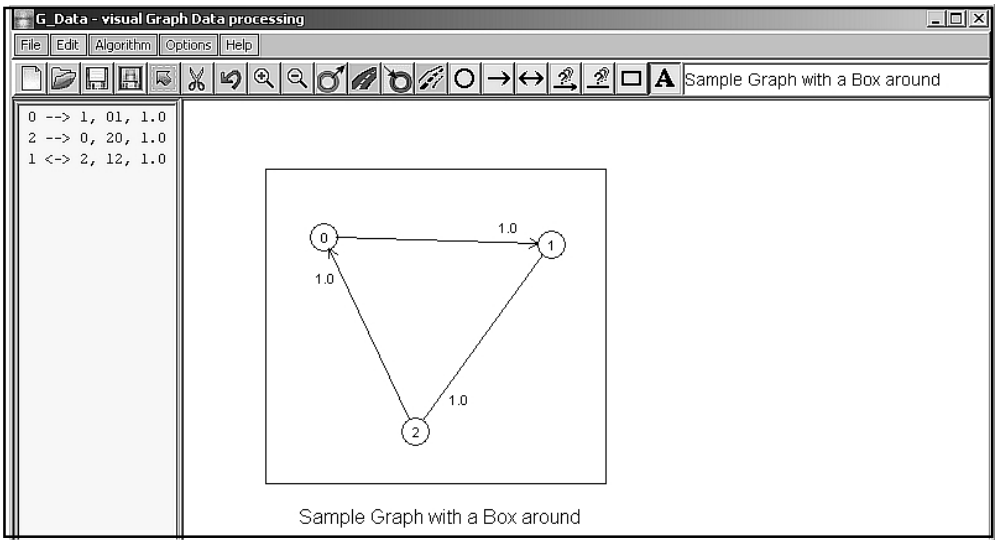

Fig. 5

• ***Text:*** Some explanatory text can be entered in the text line on the tools bar and can be inserted by clicking the mouse at the desired position (Fig. 5). The corresponding toolbar button is  .

• ***Select:*** This menu item is for selecting of graph elements for subsequent deleting (cutting). That operation is dependable of the graph structure specifics. For example – selecting the node selects also all adjacent edges to that node . It is possible to select multiple elements at once (see Fig. 6), or single edge or just edge cost (if deleting it and entering new cost is a goal). The corresponding toolbar button is  .

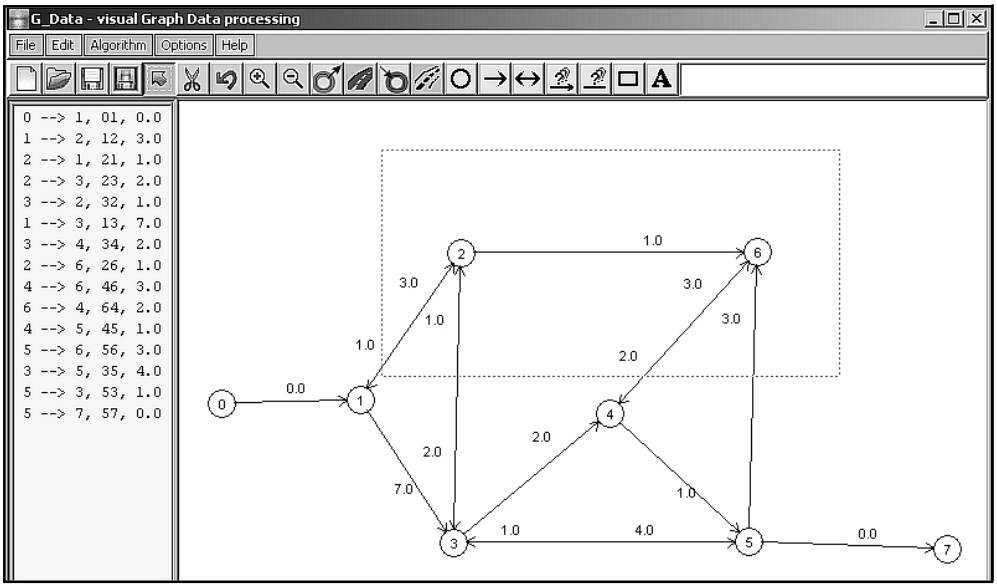

Fig. 6

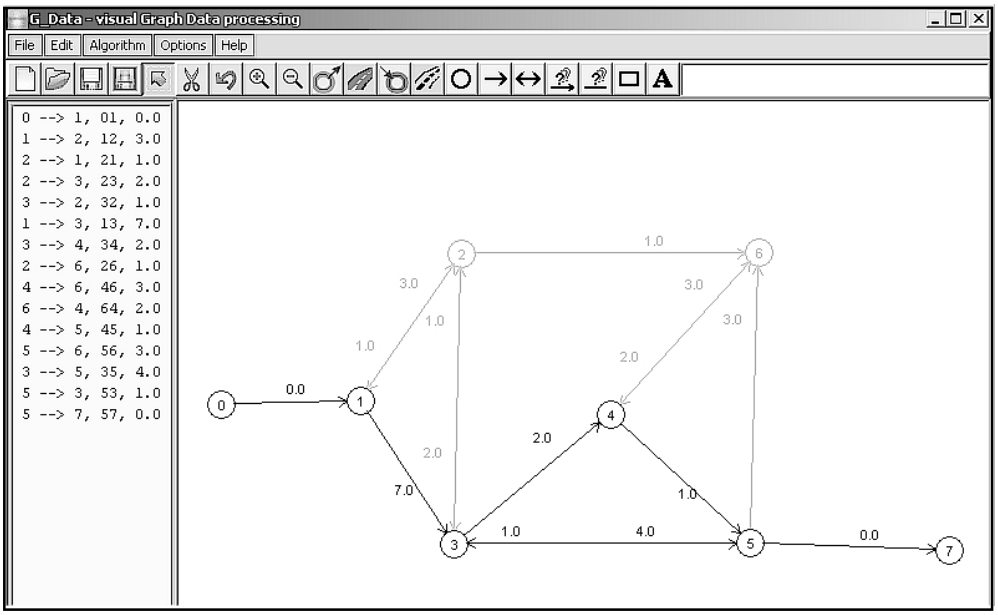

Fig. 7

- ***Cut selected:*** As the menu item suggests it cuts selected graph elements – for example shown on Fig. 7, the result will be the graph shown on Fig. 8. The text graph data on the left are automatically updated accordingly to the new graph structure.

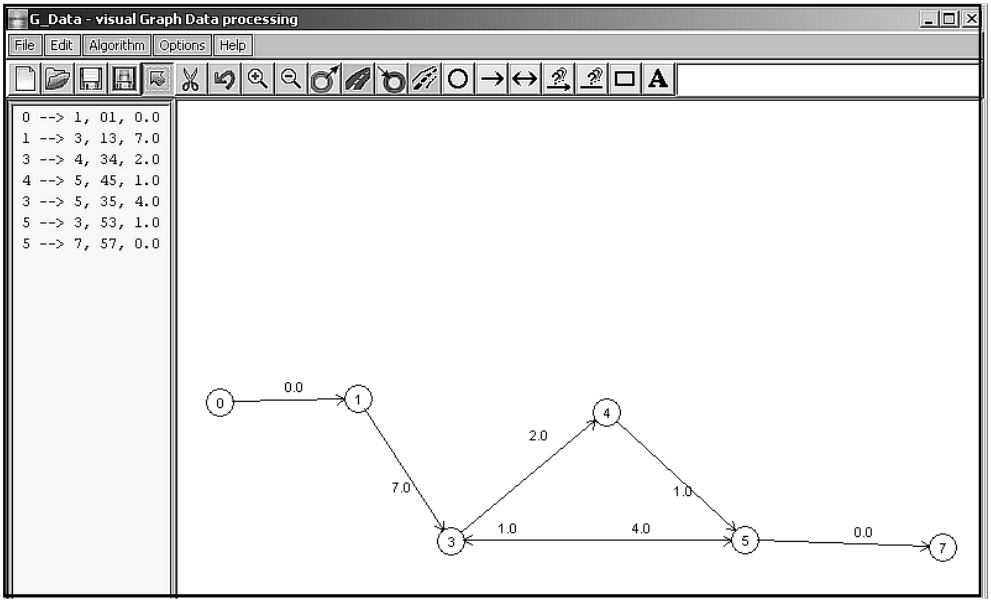

Fig. 8

The menu **"Algorithm"** (Fig. 1) will contain main known combinatorial graph algorithms. To start with, the **"Dijkstra"** algorithm is implemented as a typical shortest path problem solving in the current version of the system.

The menu **"Options"** (Fig. 1) contains the single submenu item for now – **"Font"** for changing the font of the entered explanatory text.

The last main menu **"Help"** contains standard items for **"Help"** and **"About"** information.

The implemented specific buttons without menu analogs are as follows:

 – Zoom in/out of the graph picture by 10% with each click.

 – For determination of source/destination nodes respectively.

 – Shortest path calculating and clearing it.

 - Moving (dragging) the cost numbers of directed and undirected edges to some new topological positions for better viewing of the graph structure. When costs for each edge are entered they are automatically positioned at some place "above" the edge near to the its ending point. Sometimes that position overlaps edge and should be manually changed.

As the system evolves some new menus and buttons would be implemented.

IV. Conclusion

The first variant of the system has been tested and has proven its functionality. It will be expanded with other shortest path algorithms and presumably with other

combinatorial graph algorithms. It is planned to use the system for teaching of operations research students for visualizing of the shortest path algorithms working procedures.

References

1. Moustakerov, I., B. Metev, V. Monov. "INFO1"– a File-based Management Information System. – Cybernetics and Information Technologies, Vol. **1**, 2002, No 1, 103-109.
2. Moustakerov, I. General purpose file-based information system. – Problems of Engineering Cybernetics and Robotics, **52**, 2001, 72-76.
3. OR-Objects Tutorials – Shortest Path, 1997-2000.
   http://opsresearch.com/OR-Objects/tutorials/index.html
4. Image processing with Java 2D.
   http://www.javaworld.com/javaworld/jw-09-1998/jw-09-media-p3.html
5. Java Tip 109: Display images using JEditorPane.
   http://www.javaworld. com/ javaworld/javatips/jw-javatip109.html
6. ICS 161: Design and Analysis of Algorithms. Lecture notes, February 8, 1996.
   http://www.ics.uci.edu/~eppstein/161/960208.html
7. McGill University: School of Computer Science Winter 1997 Class Notes for 308-251B – Data Structures and Algorithms, Topic #29: Shortest Path Algorithms.
   http://www.cs.mcgill.ca/~cs251/OldCourses/1997/topic29