PROMOTING OPERATIONS RESEARCH EDUCATION USING A NEW WEB – ACCESSIBLE DIDACTIC TOOL

BALOUKAS THANASIS*,
PAPARRIZOS KONSTANTINOS
SIFALERAS ANGELO

Department of Applied Informatics, University of Macedonia Greece, Thessaloniki

Abstract

The scope of this work is to present a new tool with clear didactic orientation, which will serve as a complementary means in teaching graphs algorithms. The proposed educational software can be used in courses like "Network Optimization" or "Graph Theory" and has been implemented using the Java Programming language. This Java applet is freely available, platform independent and highly interactive. The most important feature is that it is Web – accessible and can be used from any remote place, through a common Internet browser. It is a friendly application, for the Operations Research Scientist, as also for the novice student. It features visualizations of the Depth First Search (DFS), Breadth First Search (BFS), algorithms for determining graph connectivity (strong or not), topological ordering and the Dijkstra's shortest path algorithm. Furthermore, its use is analytically shown through an illustrative example. Benefits and drawbacks are thoroughly described in order to support the significance of this tool in Operations Research education and also possible future work is discussed.

Keywords: *Operation Research Education, Network Optimization, Animation, Visualization, Java*

1. INTRODUCTION

Algorithm visualization (AV) depicts the execution of an algorithm, as a sequence of graphical snapshots, the view of which is controlled by the user [1]. A variety of approaches can be used to deliver AV over the Web. The limiting factor in all these approaches was the system – dependent code in which the AV software was written [2]. The system independence and Web compatibility of the Java language [3, 4, 5, and 6] have changed the landscape significantly. Applets written in Java seem to be very promising for the Open and Distance Learning. In this paper, a next step in this evolving use of the Web for educational purposes is being made. A step made possible by Java and Java – enabled browsers.

The proposed applet focuses on algorithms for graphs and networks and presents animated their solutions. For a discussion of these algorithms the reader is referred to [7, 8 and 9]. This applet is not the only software of this kind. Many other educational programs exist for teaching graph theory or network optimization [10, 11, 12, 13, 14 and 15]; each of them of course, has its own merits and demerits. EVEGA [10] is an educational visualization environment written in Java which, in its current version, implements only standard graph search routines such as breadth first search (BFS), depth first search (DFS) and the maximum flow algorithm. BALSA II [11], TANGO [12] and XTANGO [13] haven't been implemented in Java. As a result they are dependent on a particular platform and not accessible through Java – enabled Web browsers. DIDAGRAPH [14] also is another useful, more recent, educational tool, but it is not web – accessible and platform independent. Moreover the Network – enabled solver for the assignment problem presented at [15] features visualizations only for the assignment problem and doesn't incorporate more graph algorithms. On the other hand our applet is easily extensible and more graph algorithms will be added in a future version. Moreover its comprehensive documentation enables its extension not only by the author but by any other programmer familiar with Java language. All the necessary data structures for the graph are declared and initialized near the top of the applet's class. So, all he has to do is to add the appropriate code inside the update method of the component, where the graphics rendering takes place. This way any new graph algorithm can be easily incorporated in the tool and be animated.

* Corresponding Author

There is extensive research in the Web that deals with the evaluation of the instructional efficacy of such educational tools. Some studies suggest that algorithm visualizations will not benefit "novice" students, who are just learning a new topic. On the other hand, the visualizations will probably benefit the more advanced students. These "experienced" students may use the visualization to refine their understanding of a particular algorithm. Novice students would benefit more by actually constructing an algorithm visualization, rather that viewing a predefined one [16]. However, although having students code their own visualizations can be very valuable, it can also be very time – consuming endeavor.

The work, described in this paper, focuses on an environment for delivering predefined visualizations over the World Wide Web. Moreover, some researchers maintain that it would be better for the student to be more "active" in watching the visualization. Lawrence [18] found that students, who constructed their own input data sets for the algorithm being viewed, scored significantly higher on a post test than students who watched the visualization passively. Besides, some studies claim that accompanying textual explanations are absolutely essential to the effective instructional use of Algorithm Visualization. They recommend that it would be better to give the student textual information at every algorithm step and not only to present students with a non-stop animation.

Taking the above considerations into account, this Java applet was decided by us, to have the following characteristics:

- to feature two modes for presenting the visualizations to the students. Firstly, the algorithm animations may run throughout at one step, allowing the user to set a delay (in milliseconds) between consecutive steps. Secondly, it can show several running animations by sequences of discrete snapshots.
- to allow students to input their own fully customizable data sets for the algorithm being animated. By doing so, it is hoped that students would get more learning benefits, rather than presenting them a mere predefined input data set. It is believed that when someone experiments with different input data every time he runs an algorithm animation, he will eventually be more familiar with the specific algorithm's behavior.
- to present students with textual information inside the applet. At every step of the algorithm, the student is provided with all the necessary textual information regarding the state of the variables used by the algorithm (i.e. what is the stack content in a given iteration). It should be pointed out that documentation is dynamically produced simultaneously with the algorithm execution.

2. DESCRIPTION OF THE APPLET'S GRAPHICAL USER INTERFACE (GUI)

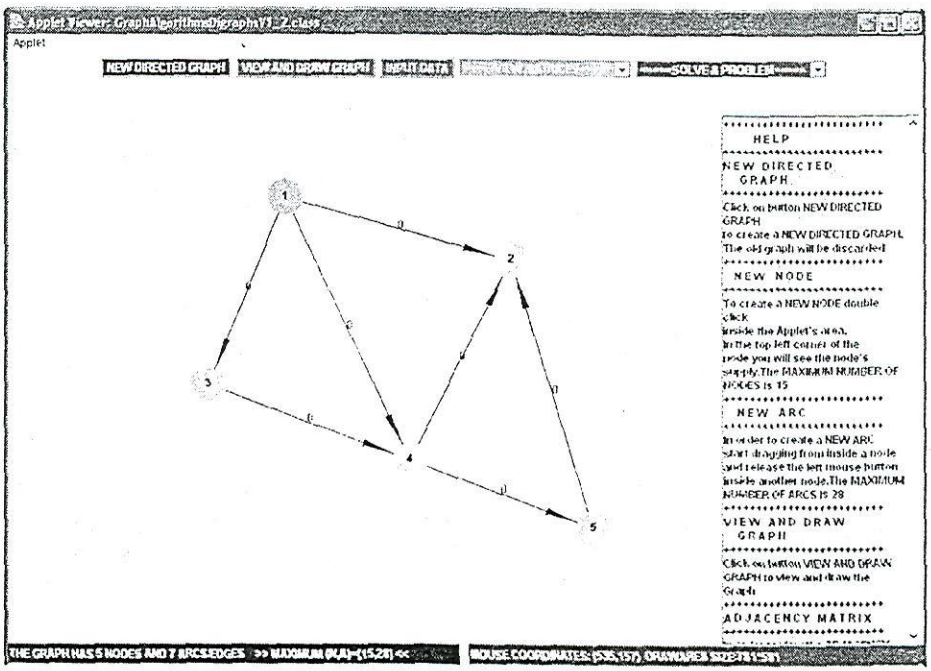

Figure 2.1 The GUI

The applet's GUI is divided into four regions, as it is shown in Figure 2.1.

The top region consists of the applet's menu. The user can:

- Create a new digraph. All the graph nodes get a random color every time the user double – click inside the graph drawing area.
- View and draw an existing digraph. When a user has already created a graph, he will always be able to draw it again and modify the existing one (i.e. add new nodes and modify arc costs).
- Input a data set, namely input arc costs that would be used from network algorithms like Dijkstra.
- View the matrices which store the digraph. The node – node incidence (adjacency) matrix, as also the node – arc incidence matrix graph representations have been used in the proposed tool.
- Solve a problem. The approach, used in this applet, is problem oriented and not algorithm oriented. The user can choose to solve a problem (i.e. search problem or shortest path problem by selecting the corresponding algorithm which solves the specific problem).
- Choose to execute an algorithm. The user can execute an algorithm either by non-stop animation providing optionally a time delay in milliseconds, or step by step. Whichever case is selected, all the necessary textual information that accompanies the algorithm execution is depicted inside the blue vertical strip.
-

The blank region at the left hand side – which takes up the biggest area of the applet, is actually the Graph editor. Graph editing guidelines are provided inside the blue vertical strip that lies at the right hand side of the applet.

The blue vertical strip at the right hand side has three uses:

- Its first use is to give help to the user regarding the graph editor, while he is editing the graph.
- Its second use is to enable the user to input the data set, to be precise to input the arc costs (this point will be clarified later by giving an example).
- Last, its third use is to depict textual content – sensitive information regarding the algorithm being animated.

Finally, **the bottom horizontal strip** which consists of two labels (in Java), gives feedback to the user. More specifically it informs him of how many nodes, or arcs, have already been created and warns him when something might go wrong (i.e. an alphabetic character instead of a number is typed inside a text box whose value will be assigned to an arc's cost). Moreover, while the user draws the graph, it informs him about the mouse position coordinates inside the blank drawing area. In order to make the applet as configurable as possible, three parameters were added:

- nodeDiameter, the diameter of each node in pixels.
- nodeToNodeDistance , the desirable distance between nodes in pixels.
- nodeDistanceFromCanvasBorder, the desirable distance of graph from the four borders of the drawing area in pixels.

3. IMPLEMENTATION ISSUES

The proposed applet has been implemented in Java, compiled with Java 2 SDK 1.4.2 package and executed using the J2SE Runtime Environment (JRE) 1.4.2_03. The applet has been tested extensively on a Windows XP Professional system, with Service Pack 2 and Java Runtime 1.4.2_03 installed. The Java 2 Runtime Environment allows you to run applications written in the Java programming language and can be freely downloaded from Sun Microsystems [17].

In order for someone to browse the proposed applet, he must have the Java Runtime Environment previously installed on his system. The AWT package has been mainly used for all graphics rendering. Furthermore, the interfaces have been created with the package java.util. Finally, Collection and Map classes have been used extensively since algorithms like Dijkstra use sets of Graph Nodes.

4. AN ILLUSTRATED EXAMPLE: DIJKSTRA'S ALGORITHM

A demonstration of the applet using an example that implements the single source shortest path algorithm in networks with non negative arc lengths (Dijkstra) will be done. The question is to find the shortest paths from node 1 to all the other nodes of the graph.

The user begins by clicking the button <<NEW DIRECTED GRAPH>> in order to create a new digraph. To create a NEW NODE one has to double – click inside the applet's area. In order to create a NEW

ARC he has to start dragging from inside a node and release the left mouse button inside another node. After having created the digraph the user may input the arc costs by clicking on button <<INPUT DATA>> and giving the costs inside the text boxes, that lie within the blue vertical strip at the right hand side of the applet's area, see Figure 4.1.

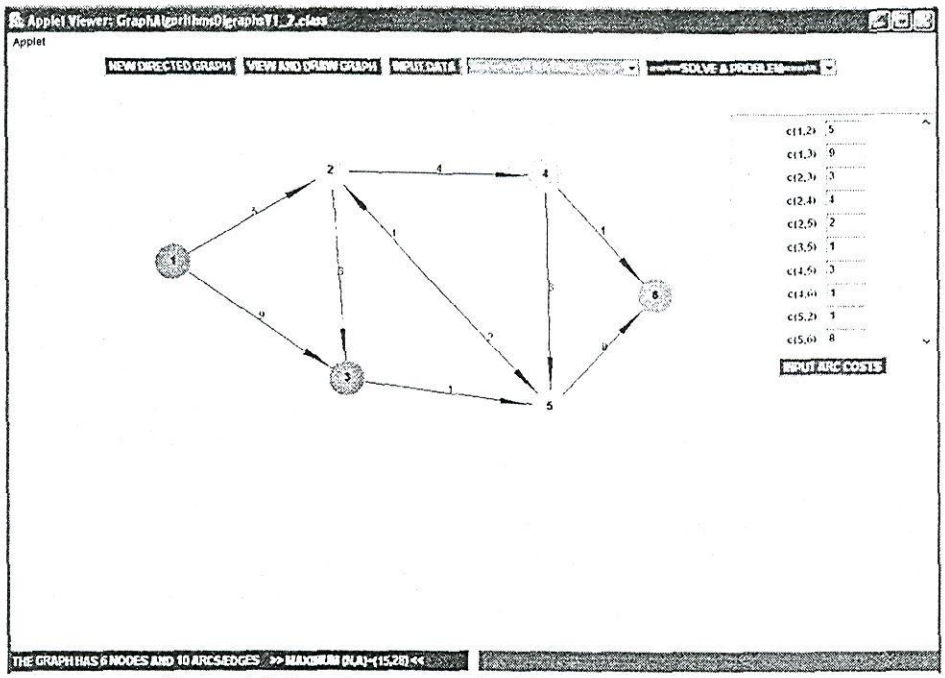

Figure 4.1 Creating a network with non negative arc costs

The network depicted in Figure 4.1 has non negative arc costs (lengths) so we can apply the Dijkstra's algorithm. At this point the pseudo – code of the Dijkstra's algorithm will be presented, in Table 4.1, in order to remind the algorithm's steps to the reader. The symbol "–" means set difference and the symbol "U" means set union. Moreover, the words, written in italics font style, mean comments of the code. Finally, we suppose that the source node is node 1.

Input: the graph the arc costs c_{ij} Output: d, the array of the shortest path lengths p, the vector which stores the parent of each node

s=1
d(s) =0
d(j) = ∞ *for each node* $j \neq s$
p(s) = s , *the parent of source node equals to the start node*
S= {s}, *the set of nodes in S consists initially only of the source node*
l = s
S'= N – S
while S' $\neq \emptyset$ and {(i,j): i $\in$ S , j $\in$ S' } $\neq \emptyset$
for each arc (l,j) : j $\in$ S'
if d(j) > d(l)+ c_{lj}
d(j) = d(l) + c_{lj}
p(j) = l
end if
end for
d(l) = min{d(i):j $\in$ S'}
S=S U {l}
S'=S'– {l}
end while

Table 4.1 Dijkstra's algorithm pseudo – code

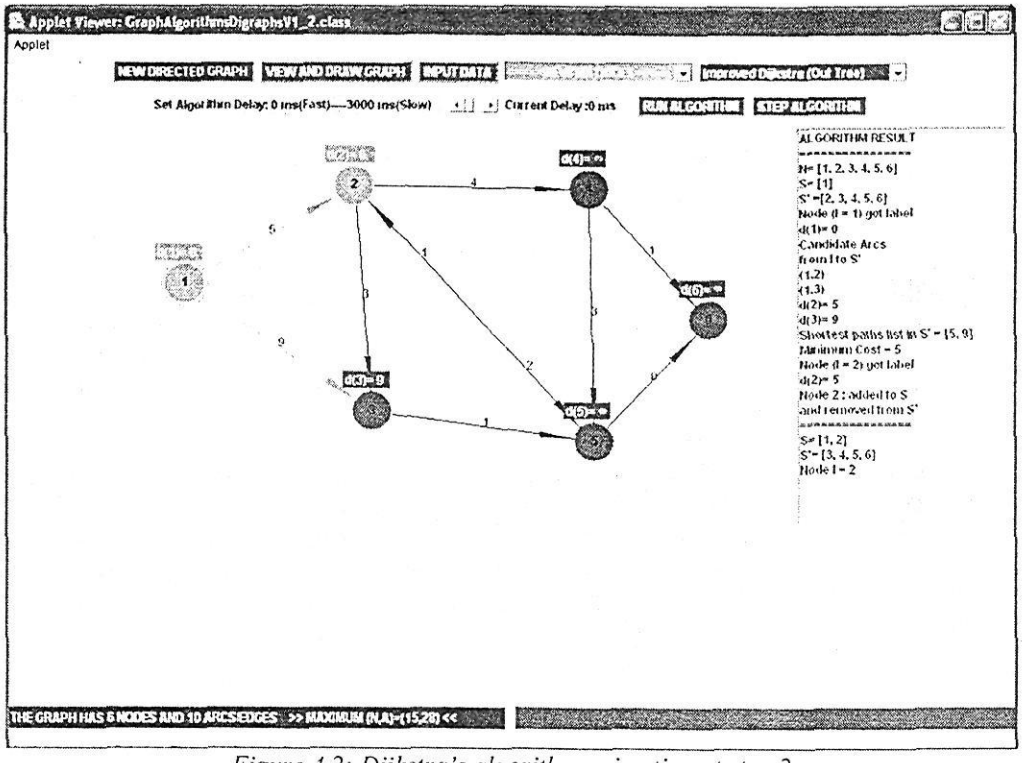

Figure 4.2: Dijkstra's algorithm animation at step 2

Now, the user must select <<Dijkstra (Out tree)>> from the choice <<SOLVE A PROBLEM>>, to start the animation. From that point, the program allows the user to either run the algorithm at once (allowing him to set a time delay in milliseconds between consecutive steps) or to run the algorithm step by step. In Figure 4.2 the animation progress is depicted, just after the user having pressed the button <<STEP>> two times.

It should be mentioned that textual explanation, for the specific algorithm, is depicted inside the blue window which lies at the right hand side of the applet. The textual information assists the user absorbing the algorithm, being taught, in a better way. For each algorithm's step it was decided to depict the following textual information for a better user understanding of the algorithm:

- The values of sets S and S'.
- The node l.
- The candidate arcs from node l to set S'.
- The shortest path array values d(j) for all nodes j whose d(j) are greater than $d(l) + c_{lj}$.
- The list of costs from which the minimum cost and hence the new node l is selected.
- Information that node l has added to set S and removed from set S'.

Moreover the nodes, that belong to set S, are colored green, whereas the nodes that belong to set S' red. In addition, at the top of each node, its current shortest path length from the source node is written. Whenever a node joins the shortest path tree a green arc is drawn. Using suitable colors, the user's understanding of a specific algorithm can be further improved. Furthermore, certain sound clips have been used at the following circumstances:

- To inform the user if he tries to create a new node, or arc, when the maximum number of nodes, or arcs, has been reached. For the educational purposes of our program the maximum dimensions for a new graph, have been set to 15 nodes and 28 arcs.
- To inform the user if he tries to type an invalid character inside a text box, whose value will be assigned to an arc's cost.
- To inform the user if he tries to run an algorithm step by step, that the algorithm's execution has reached to the end.

It is believed that using sound clips, inside similar educational software, might improve the user understanding of an algorithm.

5. CONCLUSION – FUTURE WORK

To sum up, such applets can prove to be very helpful to all the students, in general, who cannot attend a course in a University class. Every student who wishes to learn one of the algorithms, implemented in this tool, may use the proposed applet from any remote place, probably his home. It doesn't matter what his system is. It only requires that the Java Virtual Machine, (JVM), is installed into his system. It should be pointed out that JVM is freely available for download from Sun Microsystems.

Moreover there is an intention to enhance the graph editor, in order that the user would be able to draw undirected graphs, or networks, as well. Yet this applet could be further enriched, and possibly visualize more algorithms for digraphs – directed networks such as the Primal Simplex Algorithm which solves the Minimum Cost Network Flow Problem (MCNFP).

Besides more algorithms will be added for undirected graphs such as the Kruskal's and Prim's algorithms which solve the Minimum Spanning Tree (MST) problem. In a future release the applet will be modified, in order to be executed as standalone Java application as well.

In addition, certain questionnaires will be given to the students of our department to be filled in, in order to get feedback. This is very important, to improve and evaluate the teaching effectiveness of our applet in real instruction environment. Students will also be encouraged to construct their own applets. Finally, the applet being described in this paper can be accessed or downloaded from http://eos.uom.gr/~thanasis/GraphAlgorithms.htm.

BIBLIOGRAPHY

[1] Naps T. (1997), "Algorithm visualization on the World Wide Web—the difference Java makes!", in Proceedings of the 2nd conference on Integrating technology into computer science education (ITICSE), Uppsala, Sweden;

[2] Naps T. (1996), "Algorithm Visualization Delivered Off the World Wide Web – Why and How" in Proceedings of the Association for Computing Machinery's SIGCSE/SIGCUE Conference on Integrating Technology into Computer Science Education, Barcelona, Spain;

[3] Bates B. and Sierra K. (2003), "Head First Java", published by O'Reilly;

[4] Cohen S. et al (1996), "Professional Java Fundamentals", published by Wrox Press Inc;

[5] Sikora Z. (2003), "Java Practical Guide for Programmers", published by Morgan Kaufmann;

[6] McBride P. (2002), "Java Made Simple", published by Made Simple;

[7] Ahuja, K. R., Magnanti, L. T. and Orlin, B. J. (1993), "Network Flows: Theory, Algorithms and Applications", Prentice Hall, Englewood Cliffs, NJ;

[8] Paparrizos K., Samaras N. and Sifaleras A. (2005), "Network Optimization", published by University of Macedonia Press;

[9] Waite M. and Lafore R. (1998), "Data Structures & Algorithms in Java", published by Waite Group Press; Book and CD Rom edition;

[10] Khuri S. and Holzapfel K. (2001), "EVEGA: An educational visualization environment for graph algorithms", ACM SIGCSE Bulletin, 33/3, 101-104;

[11] Brown M. (1998), "Exploring algorithms using Balsa-II", Computer, 21/5, 14-36;

[12] Stasko J. (1990), "TANGO: A framework and system for algorithm animation", Computer, 23/9, 39-44;

[13] Stasko J. (1992), "Animating algorithms with XTANGO", SIGACT News, 23/2, 67-71;

[14] Dagdilelis, V. and Satratzemi, M. (1998), "DIDAGRAPH: Software for Teaching Graph Theory Algorithms", ITICSE '98, Dublin, Ireland;

[15] Andreou D., Paparrizos K., Samaras N. and Sifaleras A. (2005), "Application of a New Network-enabled Solver for the Assignment Problem in Computer-aided Education", Journal of Computer Science, 1/1, 19-23;

[16] Stasko J., Badre A. and Clayton L. (1993), "Do algorithm animations assist learning? An empirical study and analysis", in Proceedings of the INTERCHI '93 conference on Human factors in computing systems, Amsterdam, The Netherlands, 61 – 66;

[17] Sun MicroSystems, J2SE Runtime Environment (JRE) available from http://java.sun.com;

[18] Lawrence, A.W. (1993), "Empirical Studies of the Value of Algorithm Animation in Algorithm Understanding", Doctoral Thesis, Georgia Tech University.