OPEN AND DISTANCE LEARNING ENDORSEMENT THROUGH A WEB-BASED VISUALIZATION SOFTWARE

ΠΡΟΩΘΗΣΗ ΤΗΣ ΑΝΟΙΚΤΗΣ ΚΑΙ ΕΞ ΑΠΟΣΤΑΣΕΩΣ ΕΚΠΑΙΔΕΥΣΗΣ ΜΕΣΩ ΕΝΟΣ ΛΟΓΙΣΜΙΚΟΥ ΟΠΤΙΚΟΠΟΙΗΣΗΣ

Baloukas Thanasis*
Dept. of Applied Informatics, University of Macedonia 156 Egnatia Str., 54006, Thessaloniki, Greece,
Ph.D candidate, Dipl. Eng. In Computer Engineering & Informatics
E–mail: thanasis @ uom.gr

Paparrizos Konstantinos
Dept. of Applied Informatics, University of Macedonia 156 Egnatia Str., 54006, Thessaloniki, Greece,
Professor in Applied Informatics
E–mail: paparriz @ uom.gr

Rossiou Eleni
Dept. of Applied Informatics, University of Macedonia 156 Egnatia Str., 54006, Thessaloniki, Greece,
Ph.D candidate, M.Sc. in Computer Science
E–mail: rossiou @uom.gr

Περίληψη

Στην παρούσα εργασία παρουσιάζουμε ένα καινούριο εκπαιδευτικό λογισμικό για τη διδασκαλία αλγορίθμων γραφημάτων. Το πιο σημαντικό χαρακτηριστικό του λογισμικού – το οποίο είναι γραμμένο στη γλώσσα προγραμματισμού Java – είναι ότι μπορεί να χρησιμοποιηθεί τόσο τοπικά όσο και από απόσταση με τη βοήθεια ενός φυλλομετρητή του διαδικτύου. Συνεπώς είναι κατάλληλο για την ανοιχτή και εξ αποστάσεως εκπαίδευση.

Abstract

In this work we present a new software with clear educational direction, which assists in teaching graph algorithms over the Web. The most important feature of the Java applet being presented is that can be accessed both locally and remotely through a common Web Browser. Therefore, the proposed software is appropriate for the Open and Distance Learning.

Keywords

Computer Science Education, Web-based Education, Open Education, Algorithm Visualization, Graph Algorithms.

1. Introduction

Because of the importance of Graph Algorithms in Computer Science curricula, researchers have been trying to find out ways to teach them in a way that enhances student comprehension. Algorithm visualization (AV) is a depiction of an algorithm using an integrated set of multimedia tools such as graphics, text, color, sound, code, animation and video. AV could be used to attract students' attention during lectures, explain concepts in visual terms and encourage a practical learning process. Moreover, since systems that incorporate algorithm visualizations run interactively, it is possible for students to try out their own example problems, and thus increasing their understanding of a specific algorithm.

* Corresponding Author

A variety of approaches have been used in the past in order to deliver AV over the Web. The restrictive matter in all these approaches was the system – dependent code in which the AV software was written (Naps, 1996). The system independence and Web compatibility of the Java language (Bates et al., 2003; Cohen et al., 1996; Sikora, 2003 and McBride, 2002) have changed the landscape significantly. Applets written in Java seem to be very promising for the Open and Distance Learning, since they could be executed remotely using a common Web Browser. In this paper, a next step in this evolving use of the Web for educational purposes is being made. This step is made possible by Java and Java – enabled browsers. The proposed applet demonstrates algorithms for graphs and networks and explains their operation through visualization. The following algorithms have been implemented: Depth First Search (DFS) and Breadth First Search (BFS) algorithms, that determine graph connectivity (strong or not), topological ordering algorithm that determines whether there exist directed cycles inside a digraph, Dijkstra's shortest path algorithm implemented in two ways and Minimum Spanning Trees Algorithms due to Prim and Kruskal. For a discussion of these algorithms the reader is referred to (Ahuja et al., 1993; Paparrizos et al., 2005 and Waite et al., 1998).

This applet is not the only software of this kind. Increasingly, many algorithm visualization tools have been developed and presented at conferences (Khuri et al., 2001;Brown, 1998;Stasko, 1990;Stasko, 1992;Dagdilelis et al., 1998;Andreou et al., 2005 and Naps et al., 2000), each of them having its own merits and demerits. EVEGA (Khuri et al., 2001) 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. In addition, the visualization system proposed in (Müldner et al., 2004) that is based on Macromedia Flash MX doesn't incorporate visualizations for graph algorithms. BALSA II (Brown, 1998), TANGO (Stasko, 1990) and XTANGO (Stasko, 1992) haven't been implemented in Java. As a result, they are dependent on a particular platform and are not accessible through Java – enabled Web browsers. DIDAGRAPH (Dagdilelis et al., 1998) 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 (Andreou et al., 2005), features visualizations only for the assignment problem and doesn't incorporate more graph algorithms. Besides, in JHAVE (Naps et al., 2000) the context-sensitive documentation has been written statically, that is to say, before the execution of the algorithm that it tries to explain.

On the other hand, we think that the tool being proposed overcomes the limitations mentioned above since it has the following characteristics:

- *It demonstrates visualizations for directed and undirected graphs:* we have animated six algorithms up to now and more are going to be animated in a future version. As a consequence the applet could be used as accompanying teaching and learning material for lessons such as "analysis and design of algorithms" as well as "network optimization".
- *It has been implemented in Java as applet:* as a consequence it is accessible remotely by any Java enabled Web browser.
- *It is highly interactive*: incorporating a convenient graph editor, it allows the student to draw his own graph. What is more, it allows him to always modify the existing graph (i.e. to add new node/arcs and to delete existing node/arcs). All he has to do is to click on the button "VIEW AND DRAW GRAPH" and to follow the graph editing guidelines that are depicted inside the blue vertical strip that lies at the right hand side of the applet. Moreover by clicking on button "INPUT DATA" he can

easily input or modify the node supplies and arc/edge costs. The latter data are going to be used from graph algorithms like Dijkstra, Prim, Kruskal and the Network Simplex Method. The later algorithm is going to be added in a future version of the applet.

There is extensive research in the Web that deals with the assessment 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 (Stasko, 1993). However, although having students write their own visualization programs can be very important, 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 a student to be more "active" in watching the visualization. Lawrence found (Lawrence, 1993) 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 for the effective instructional use of Algorithm Visualization. They recommend that it would be better to provide the student with textual information at every step of the algorithm instead of providing them with a non-stop animation.

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

- to incorporate two modes for presenting the visualization to the students. Firstly, the algorithm animations could be run throughout at one step, allowing the user to set a delay (in milliseconds) between successive 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. Thus, 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 context-sensitive textual information in parallel with the visualization. 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)

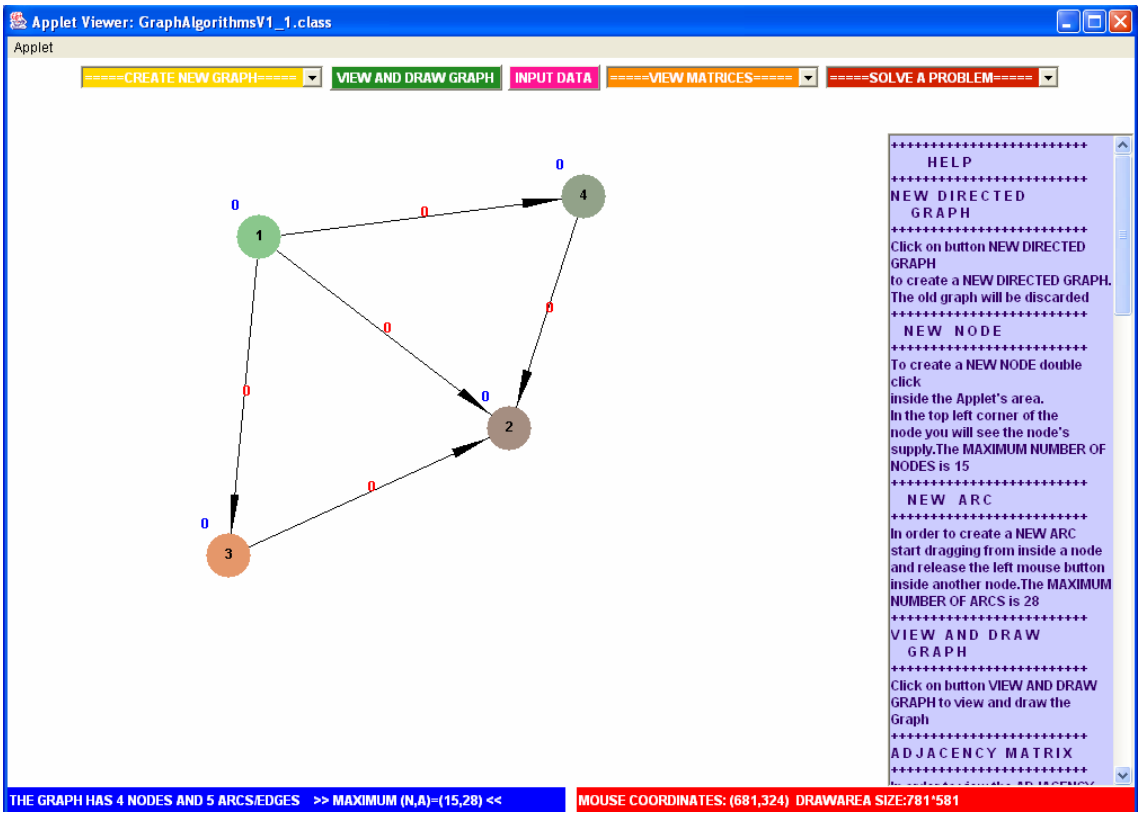

Figure 1: The GUI

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 graph.* By clicking on this choice the user can choose to create a new directed or undirected graph. All graph nodes get a random color every time the user is doubleclicking inside the graph editor (will be described comprehensively below). Moreover, the initial value of a newly created node's supply is depicted. When the user drags from inside a node and releases the left mouse button inside another node an arc and its corresponding cost are created.
- *View and draw an existing graph.* 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, add new arcs).
- *Input a data set*, namely input arc costs that would be used from network algorithms like Dijkstra.
- *View the matrices* which store the graph. 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. graph searching or shortest paths) 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. Any 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 arcs' costs (this point will be clarified later by giving an example).
- *Lastly, 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.

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. Finally, the Collection and Map classes belonging to java.util package have been used extensively, since algorithms like Dijkstra make use of sets of Graph Nodes.

3.1 Use of sound clips

Sound clips have been used in the proposed applet at the following circumstances:

- to inform the user who is trying to create a new node/arc, that the maximum number of nodes/ 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 who is trying to type an invalid character inside a text box, whose value will be assigned to an arc's cost.
- to inform the user who is trying 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.

4. An illustrated example: Breadth First Search Algorithm

A demonstration of the applet using an example that implements a variation of the Breadth First Search algorithm in directed graphs will be provided. **The question is to determine whether a digraph is connected or not.** The algorithm starts with an arbitrary node (we started with node 1), marks it as visited, gives it a label and adds it to the Queue. Thereafter, removes the node in the front of Queue and examines whether its neighbors (nodes) have been visited. If not, the algorithm marks, visits and adds them in the Queue. This is repeated until the Queue becomes empty. In order to determine that the graph is connected one has to examine if all nodes have been visited after the algorithm is finished.

At this point the pseudo – code of the algorithm in question will be presented in Table 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.

Algorithm BFS
Input: *a directed graph G* Output : **order(j)**, *the order nodes j are marked* **marked (j)**, *array of Boolean. For each node j, marked(j) is yes or no depending on whether j has been visited*

counter = 1 , *auxiliary variable*
s = 1 , *start search with arbitrary node. We start from 1*
Queue Q = Ø , *initialize Q to empty*
marked(s) = yes , *mark node s*
order(s) = counter , *visit node s and give it a label*
Q = [s] , *enqueue node s*
marked (j) = no ,∀ j≠s , *all other nodes are initially unmarked*
while Q ≠ Ø
i = node in front of queue Q , *dequeue(Q)*
Q = Q ~ {i}
for each arc (i,j) or (j,i) , *such that, node j is unmarked*
marked (j) = yes , *mark node j*
counter = counter + 1
order(j) = counter , *visit node j and give it a label*
Q = Q ∪ {j} , *enqueue node j*
end for
end while

Table 1: Breadth First Search pseudo – code

In order to create a new digraph the user chooses <<DIRECTED GRAPH>> from the choice <<CREATE NEW GRAPH>>. To create a NEW NODE he 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.

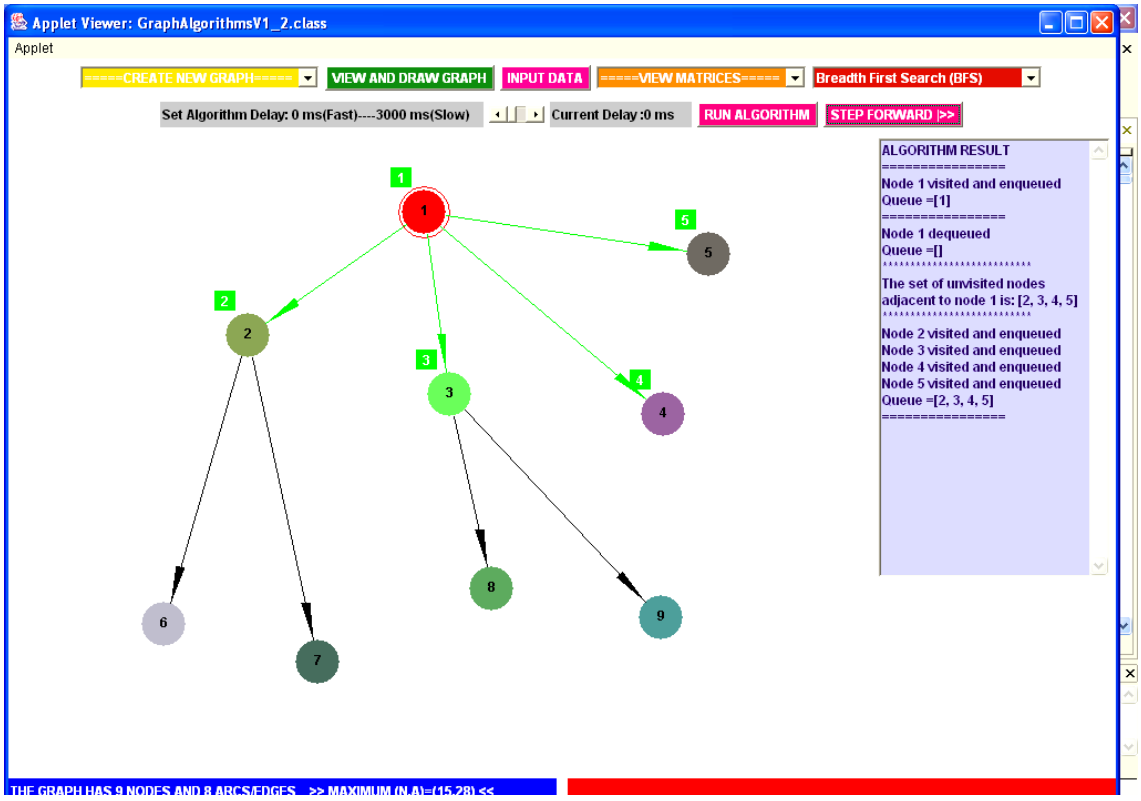

Figure 2: Breadth First Search visualization – step 2

Now, the user must select <<Breadth First Search>> from the choice <<SOLVE A PROBLEM>>, to start the visualization. From that point, the program allows the user to either run the algorithm at once (optionally allowing him to set a time delay in milliseconds between consecutive steps) or to run the algorithm step by step. In Figure 2 the visualization progress is depicted, just after the user having pressed the button <<STEP>> two times. Above each node there exists a green rectangle, displaying the order in which that node has been visited. Nodes that have been removed from the Queue are displayed in red color. Nodes that are currently into the Queue are these nodes that have a green label above them and are not displayed in red color.

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 queue contents.
- the set of adjacent unvisited nodes of the node being visited
- the fact that a node is being visited and enqueued

Finally, with algorithm completion the program informs the user about whether the graph is connected.

6. Conclusion – Future Work

In a nutshell, 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 implemented algorithms may use the proposed applet from any remote personal computer independently of the underlying operating system. The only requirement is that the Java Virtual Machine (JVM), should be installed. JVM is freely available for download from Sun Microsystems.

Moreover this applet could be further enhanced, and possibly visualize more graph and network algorithms such as the Primal Simplex Algorithm which solves the Minimum Cost Network Flow Problem (MCNFP). Besides, in a future release the applet will be modified, in order to be executed as standalone Java application as well.

In addition, we are going to assess the teaching effectiveness of the proposed software in real instruction environment. This could be done by dividing students into two groups after having been taught some algorithms: the first group will not be demonstrated algorithm visualizations whereas the second group will be. Then we are going to examine both groups in a post test and we will investigate the results. Students will also be encouraged to construct their own applets because it has been demonstrated that this increases the educational effectiveness (Stasko et al. 1993). Finally, the applet being described can be browsed from the link : http://eos.uom.gr/~thanasis/ICODL2005.htm.

References

Ahuja, Magnanti, Orlin (1993). 'Network Flows: Theory, Algorithms and Applications'. Prentice Hall, Englewood Cliffs, NJ.

Andreou, D., Paparrizos, K., Samaras, N. , Sifaleras, A. (2005). 'Application of a New Network-enabled Solver for the Assignment Problem in Computer-aided Education'. *Journal of Computer Science*, 1(4), pp. 19-23

Bates, Sierra (2003). 'Head First Java'. O'Reilly.

Brown, M., (1998). 'Exploring algorithms using Balsa-II'. *Computer*, 21(5), pp. 14-36

Cohen et al (1996). 'Professional Java Fundamentals', Wrox Press .

Dagdilelis, V. and Satratzemi, M. (1998). 'DIDAGRAPH: Software for Teaching Graph Theory Algorithms'. ITICSE '98 - Dublin, Ireland.

Khuri S., Holzapfel K. (2001). 'EVEGA: An educational visualization environment for graph algorithms'. *ACM SIGCSE Bulletin*, 33(3), pp. 101-104

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

McBride, P. (2002) 'Java Made Simple', Made Simple.

Müldner, T., Shakshuki, E. and Merill, J. (2004). 'Selecting Media for Explaining Algorithms'. AACE Proceedings of EDMEDIA'04 - Lugano, Switzerland, pp. 2048-2053.

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

Naps, T., Eagan, G. and Norton, L. (2000). 'JHAVE: An Environment to Actively Engage Students in Web-based Algorithm Visualizations'. Proceedings of the SIGCSE Session - Austin, Texas, pp. 109-113.

Paparrizos , Samaras , Sifaleras (2005). 'Network Optimization', Thessaloniki, University of Macedonia Press.

Sikora, Z. (2003) 'Java Practical Guide for Programmers', Morgan Kaufmann.

Stasko, J., (1990). 'TANGO: A framework and system for algorithm animation'. *Computer*, 23(9), pp. 39-44

Stasko, J., (1992). 'Animating algorithms with XTANGO'. *SIGACT News*, 23(2), pp. 67-71

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

Waite, Lafore (1998). 'Data Structures & Algorithms in Java', Waite Group Press.