E – learning technologies: employing matlab web server to facilitate the education of mathematical programming

P. KARAGIANNIS, I. MARKELIS, K. PAPARRIZOS,
N. SAMARAS[*] AND A. SIFALERAS.

Department of Applied Informatics, University of Macedonia
156 Egnatia Str., Thessaloniki, Postal Code 540 06, Greece

This paper presents new Web-based educational software (WebNetPro) for Linear *Network Programming*. It includes many algorithms of *Network Optimization* problems, such as shortest path problems, minimum spanning tree problems, maximum flow problems and other search algorithms. Therefore, WebNetPro can assist the teaching process of courses such as *Graph Theory* or *Network Optimization, Algorithms* or *Data Structures*. Since the Matlab Web server is part of the underlying infrastructure, the end user needs not use Matlab directly. Our approach exploits transparently the benefits of the Matlab Programming Environment through a straightforward Web interface. Furthermore, WebNetPro is platform independent. Thus, it can be viewed as a powerful supplement to traditional instruction techniques and can be used without significant difficulties in Distance Education. Benefits and drawbacks are thoroughly described in order to support the significance of this tool in Distance Learning.

1. Introduction

Mathematical Programming is a vast area, which addresses the problem of developing efficient algorithmic solutions. A special subfield of Mathematical Programming is Linear Programming, which only deals with strictly linear problems. Following the previous categorization, Linear Network Programming can be viewed as a subfield of Linear Programming. Linear Network Programming focuses primarily on linear problems which can be modelled using special data structures such as graphs or networks. The Linear Network Programming research area has been widely expanded during the last decades and includes a remarkable range of problems; see for example [1, 2].

Nowadays, most curricula in Computer Science Departments contain courses related to Linear Network Programming, Graph Theory or Combinatorial Optimization. Graph algorithms and especially their complexity aspect, is regarded as a difficult topic in Theoretical Computer Science. The development of a Web-based application, capable of presenting a visual implementation of many different graph algorithms, constitutes a useful teaching tool. Therefore, the implementation of such an educational tool was a strong motivation. To the best of our knowledge, it is the first time that such a large set of implemented algorithms, for customized problems, are set into an integrated graphical environment over the Internet.

[*] Corresponding author. Email: samaras@uom.gr

It is well known that the teaching process can be highly effective when assisted by visual and interactive tools. This way, students can obtain a deeper understanding of the subject being taught, rather than abstract theoretical knowledge. For these reasons, e – learning technologies have become a very active research area for many scientists [3, 4].

This paper presents new Web-based educational software tool for Linear Network Programming. Most of the existing on-line software applications deal with a small set of algorithms. This set becomes even smaller in cases of network representations. The number of graph algorithms included in WebNetPro surpasses the number of algorithms included in the existing freely available web-based educational solvers by a considerable amount. Moreover, it focuses on a variety of abstract data structures such as adjacency (or node – node) matrices, node – arc matrices, lists of arcs and star representations (forward, reverse or compact). WebNetPro constitutes an extension of the NetPro project [5] which is a limited stand alone version of this tool. NetPro's main drawback is that in order to be used it should have been previously installed in the users' computer. Furthermore, the set of implemented algorithms has been expanded. The new tool features algorithms for solving shortest path problems in Acyclic Networks, in nonnegative Cyclic Networks and in networks with nonnegative arc lengths. Moreover, new features, such as a set of solutions pertaining to maximum flow problems and implementations of various algorithms for the minimum spanning tree problem have been added. Prim's and Kruskal's for instance is an example of the latter case. There are also search algorithms which compute in trees or out trees of a directed graph, determine connectivity (strong or not) or find directed cycles or topological ordering. The user can also convert one network representation to another and choose between an adjacency matrix, a list of arcs, forward star representation, reverse star representation, or in some cases compact star representation [6]. Furthermore, WebNetPro gives students the ability to save their problem's definitions as files. These files can be uploaded by the students to WebNetPro at some other time of choice. This functionality is analytically described in Section 6.

This tool could be adapted for use in universities. Computer scientists, who include Linear Network Programming among their research interests, can use this method of presentation. Since the tool is platform independent every student eager to learn a specific algorithm may use it, regardless of the operating system or the browser being used. JavaScript support is the only requirement. Moreover, it is our belief that such new teaching aids and techniques can prove themselves helpful to students who cannot physically attend a course.

Following this section, technical details about the related work in this field shall be discussed. This continues, in the third section, where technical details about the implementation will be given. Details regarding the solution process are provided in the fourth section. In the fifth section, the graphical user interface will be discussed through an annotated example. Furthermore, WebNetPro's functionalities of saving user's defined problems, (in appropriate files) and uploading them later will be explained in the sixth section. Finally, the seventh section concludes the paper and poses for future directions.

2. Related work

It is a fact that a considerable number of software applications for educational purposes are available. In all probability, should specific aspects of a research field do not discourage incorporating distance learning, instructive software tools are already present in order to assist students. However, there is limited availability of on-line solvers for customized network problems, especially of those that use multiple input data types, rather than a set of prepared examples. This is the point in which WebNetPro makes an actual difference. Through the proposed software tool, instructors can assist students in order to obtain a deeper understanding of various mathematical data structures and algorithms.

One of the best available optimization software resources is *Netflo* [7]. Netflo can be used to solve minimum cost network flow problems. The user submits the problem in either the DIMACS standard format or the Netflo native format. Problems can be submitted through the World Wide Web (http or ftp

URL's) or electronic mail. Nevertheless, Netflo's primary target is large scale optimization, not education. Furthermore, its data type support is inadequate and it covers only one problem category; the minimum cost network flow problem.

Web sites, containing educational tools, are referenced in [8, 9]. The first presents an online tool which a student can use to verify the solution of minimum cost network flow problems, using the classical network simplex algorithm. The second presents a different online tool which a student can use to verify the solution of shortest path problems, using the Dijkstra algorithm. Both are implemented in Java and the user specifies the network's nodes and arcs by a pair of mouse clicks. Nevertheless, the user cannot select between graph representations, nor has the choice of selecting a specific algorithm among a number of alternatives. Another useful web resource is depicted in [10]. However, this work only provides the animated solution of a shortest path problem, using the Dijkstra algorithm. The proposed tool is also implemented in Java and the user specifies nodes and arcs in the network by a pair of mouse clicks.

A similar software tool is DIDAGRAPH [11]. DIDAGRAPH implements a wide set of algorithms and is capable of solving various customized problems. Nevertheless, it is not accessible through the Internet, and lacks the ability to work with multiple data structures. Finally, [12] depicts an on-line tool which uses animation techniques for solving customized problems. However, its primary scope is implementing Linear Programming algorithms instead of algorithms for network flow problems. As in all the aforementioned cases, there has been no consideration for multiple data structures.

3. Implementation issues

WebNetPro uses the Matlab Web server, version 1.2.3. All the algorithms have been fully implemented in Matlab Version R14, with Service Pack 3. During recent years, Matlab has become very popular among the scientific community and has been used extensively in Colleges and Universities [13, 14]. However, WebNetPro students are not required to learn Matlab. This way our tool exploits transparently all the benefits of the huge set of mathematical intrinsic functions inherent in the Matlab Programming Environment, without further requirements.

Moreover, it is platform independent and can be used efficiently in Distance Education. The proposed software is running entirely on the Matlab Web Server and can be accessed any time by any machine with Internet access by simply opening a Web browser. The technical details that follow shouldn't concern end users, except those who are interested in deploying Matlab applications, via the Web.

In order to function properly, Matlab Web server requires installation and proper set up of a typical Web server (Apache, IIS etc.). Currently, Apache Web server, version 2.0.55, is being used. There were also certain needs for acquiring input data from the user, sending them to the Matlab Web server and vice versa. This task, having considered all the security – related issues, was carried out using PHP, version 5.03. Finally, all the pop up menus of the graphical user interface were implemented using JavaScript.

4. The design of the proposed educational tool

As already mentioned WebNetPro uses the Matlab programming language. The current version of Matlab is quite sophisticated and provides a plethora of useful tools and functions. Furthermore, Matlab combines graphics and animation [15] and provides the ability to build graphical user interfaces (G.U.I.) in a simple manner.

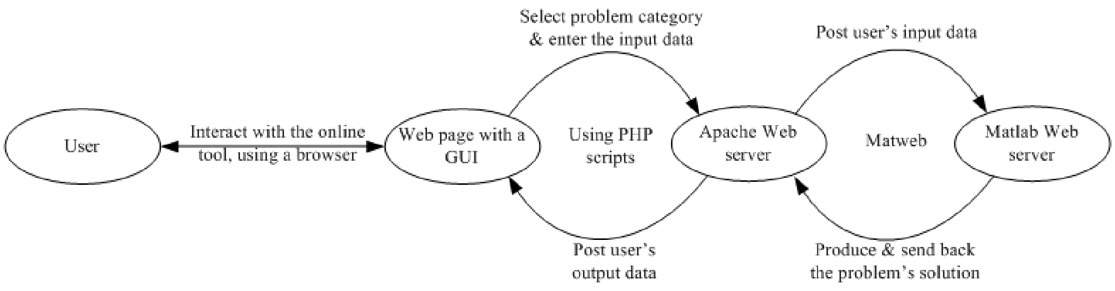

User
Interact with the online tool, using a browser
Web page with a GUI
Select problem category & enter the input data
Using PHP scripts
Post user's output data
Apache Web server
Post user's input data
Matweb
Produce & send back the problem's solution
Matlab Web server

Figure 1: Solution Process.

The whole process of solving a student - defined problem is described in figure 1. The student first accesses the on-line tool using a Web browser. Then an algorithm should be chosen through pop up menus. The graphical user interface (G.U.I.) will be sufficiently covered in the next section. In fact, the student interacts with the tool, since one can enter the necessary data input of the problem, or change them later and watch how the solution is affected. Afterwards, PHP scripts post the input data from the html form to the Apache Web server. The Apache Web server uses a specific file called *matweb* to communicate with the Matlab Web server. At this point, the solution is calculated entirely by the Matlab Web Server, not by the client computer. Then, the output data are posted backwards to the Apache Web server and then back to the Web page of the tool. In this way, the user gets the output data. Furthermore, the user is able to make changes to the input data and watch how the solution is being affected (not again from scratch).

The following Web address is the starting point, in order to use the on-line tool:

http://eos.uom.gr/~sifalera/NetPro

Consequently, our approach makes use of all the capabilities of state-of-the-art Dynamic Web technologies, such as PHP and JavaScript, which combined with the Apache Web server and Matlab produce very promising results.

5. Solving problems using WebNetPro's G.U.I.

The graphical user interface consists of a main menu bar, which comprises several other pop up menus, as illustrated in figure 2.

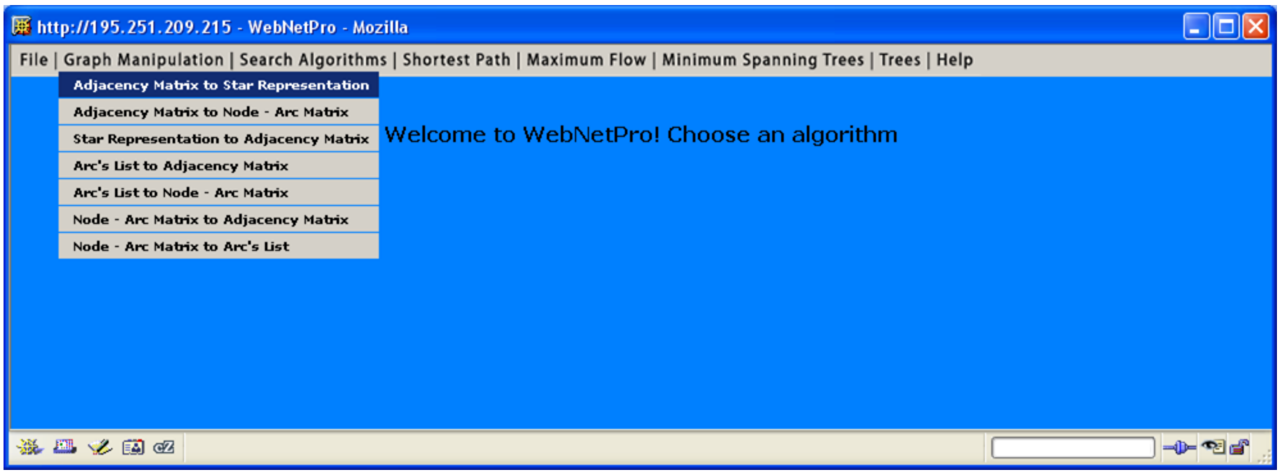

http://195.251.209.215 - WebNetPro - Mozilla
File | Graph Manipulation | Search Algorithms | Shortest Path | Maximum Flow | Minimum Spanning Trees | Trees | Help
Adjacency Matrix to Star Representation
Adjacency Matrix to Node - Arc Matrix
Star Representation to Adjacency Matrix
Arc's List to Adjacency Matrix
Arc's List to Node - Arc Matrix
Node - Arc Matrix to Adjacency Matrix
Node - Arc Matrix to Arc's List
Welcome to WebNetPro! Choose an algorithm

Figure 2. Main menu bar.

WebNetPro is an application easy to use not only for the Mathematician - Operations Research Scientist, but for the student as well. It performs quickly and efficiently the scheduled task while on the other hand the user can easily explore and get accustomed with its graphical environment.

At this point we depict a full example of a conversion between two abstract data types. An adjacency matrix conversion into a forward star representation follows. Assume that the network of figure 3 is given:

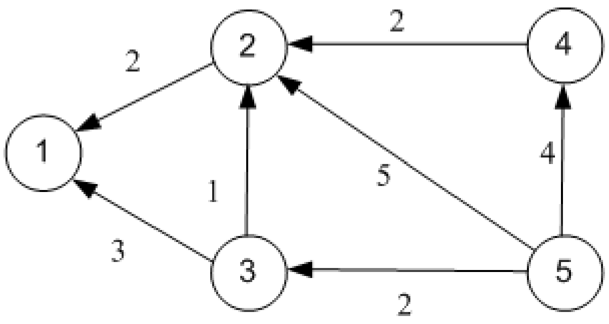


Figure 3. Network example.

The nodes can be regarded as cities, while the arcs can be regarded as one direction roads between them. The numbers next to each arc denote road length. Suppose that a company wants to transfer certain amounts of its products between different cities. The products are stored in the fifth city (supply node) and they must be distributed as soon as possible to other cities.

According to theory, the Adjacency matrix representation of a network or directed graph, is a *n* by *n* array of Boolean values, (where *n* is the number of nodes) with entry in row *i* and column *j* defined to be 1 if there is an edge from node *i* to node *j* in the graph, and 0 otherwise. In our example, the adjacency matrix is the following:

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

The user also provides a second matrix *C*, with dimensions that equal the dimensions of *A*, in order to enter the costs associated to each arc. Every arc of the network corresponds to a cost which is denoted by an element in the second matrix, exactly in the same position as in *A*. The following is the matrix with the cost coefficients:

$$C = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 5 & 2 & 4 & 0 \end{pmatrix}$$

The forward star representation of a network is similar to the adjacency list representation. This similarity relies on the fact that the forward star representation stores the node adjacency list of each node. This kind of representation stores these lists as a single array. In order to represent a network, through a forward star representation, each arc corresponds to a unique sequence number. This way, an ordering of the arc list is developed. Afterwards, all the arcs are numbered in a specific order; firstly those emanating from the first node then those emanating from the second node, and so on. The arcs emanating from the same node are numbered in an arbitrary fashion and all the information concerning each arc is stored sequentially in the arc list. For example, if arc *(g, h)* is the ninth arc, the tail, head, cost e.t.c. are stored in the array positions *tail(9)*, *head(9)* and *cost(9)*. Moreover, a pointer is associated with each node *i*, denoted by *K(i)*. The *K(i)* pointer indicates the smallest - numbered arc in the arc list, which emanates from the i^{th} node. In case the i^{th} node hasn't any outgoing arcs, then we set $K(i) = K(i + 1)$. Consequently, the forward star representation stores the outgoing arcs of the i^{th} node at positions $K(i)$ to $(K(i + 1) – 1)$ in the arc list. Also, for consistency it is set $K(1) = 1$ and $K(n + 1) = m + 1$, (we denote by m the number of the arcs). More detailed information about the benefits or drawbacks regarding different types of network structures can be found in [1].

Now, the solution of this problem, via the graphical menu, will be presented. The user must first enter the previously mentioned Web site and then from the menu bar one must select *Graph Manipulation → Adjacency Matrix to Star representation*. At this point the following window (figure 4) will demand the student enter the dimensions of the adjacency matrix:

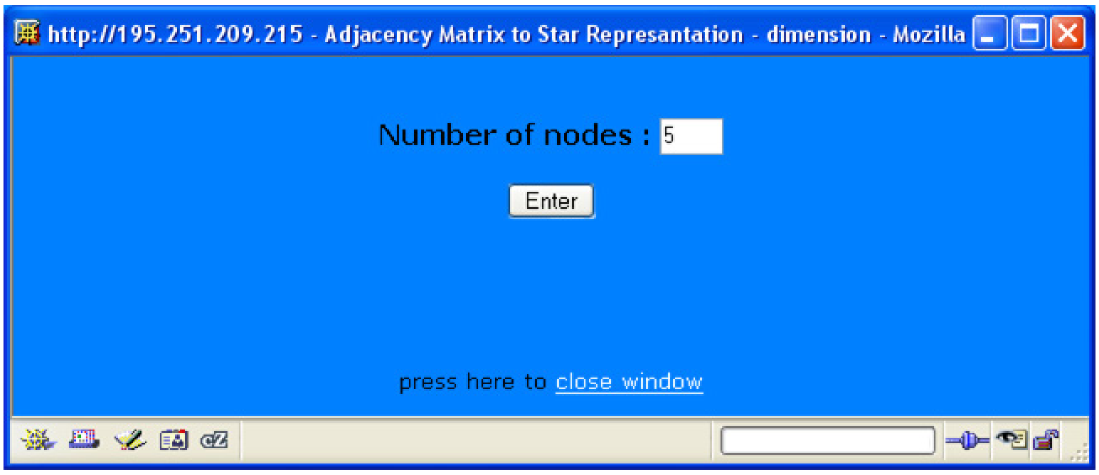
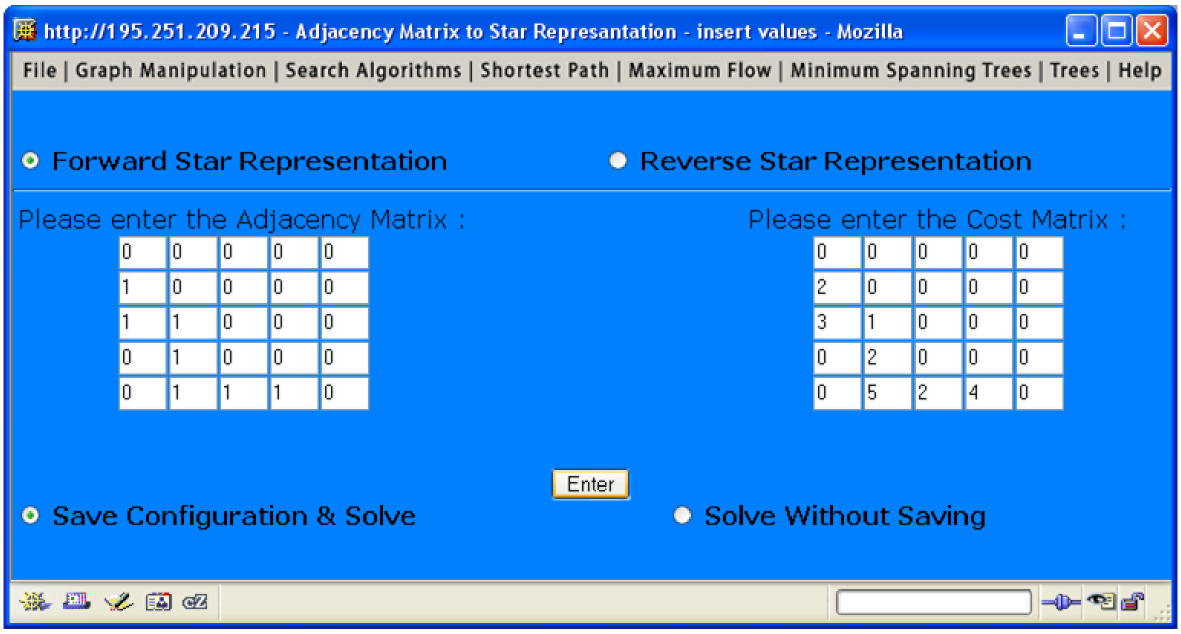

http://195.251.209.215 - Adjacency Matrix to Star Represantation - dimension - Mozilla
Number of nodes : 5
Enter
press here to close window

Figure 4. The user must enter the dimension of the adjacency matrix.

After entering the value 5 as the dimension for this example, a new window (figure 5) will appear for choosing the type of star representation and also fill the two matrices mentioned above:


http://195.251.209.215 - Adjacency Matrix to Star Represantation - insert values - Mozilla
File | Graph Manipulation | Search Algorithms | Shortest Path | Maximum Flow | Minimum Spanning Trees | Trees | Help
Enter

[✓ Forward Star Representation] Reverse Star Representation

Please enter the Adjacency Matrix :

0	0	0	0	0
1	0	0	0	0
1	1	0	0	0
0	1	0	0	0
0	1	1	1	0

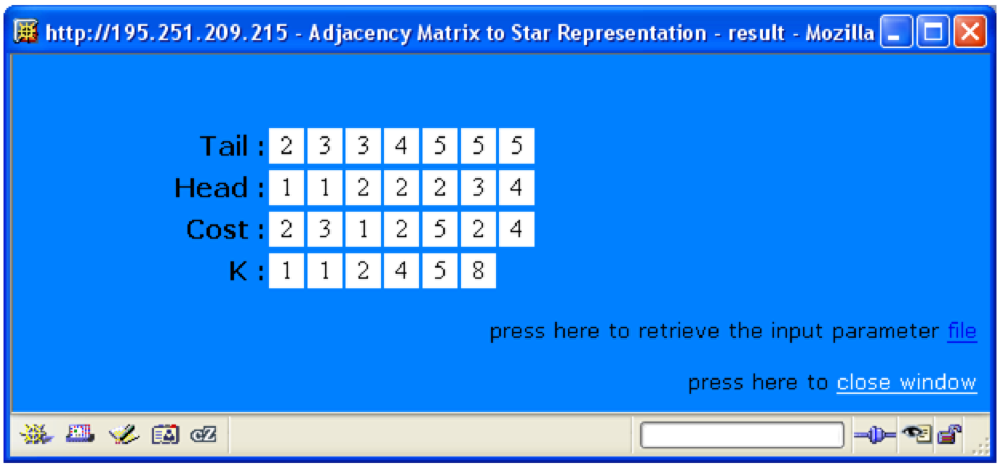
Please enter the Cost Matrix :

0	0	0	0	0
2	0	0	0	0
3	1	0	0	0
0	2	0	0	0
0	5	2	4	0

[✓ Save Configuration & Solve] Solve Without Saving

Figure 5. The user must fill the appropriate cells.

The outputs are going to be displayed in a different window. In figure 6, the output vectors are visible. Specifically, there are the “Tail”, “Head”, “Cost” and the previously mentioned pointer vector “K”.


http://195.251.209.215 - Adjacency Matrix to Star Representation - result - Mozilla

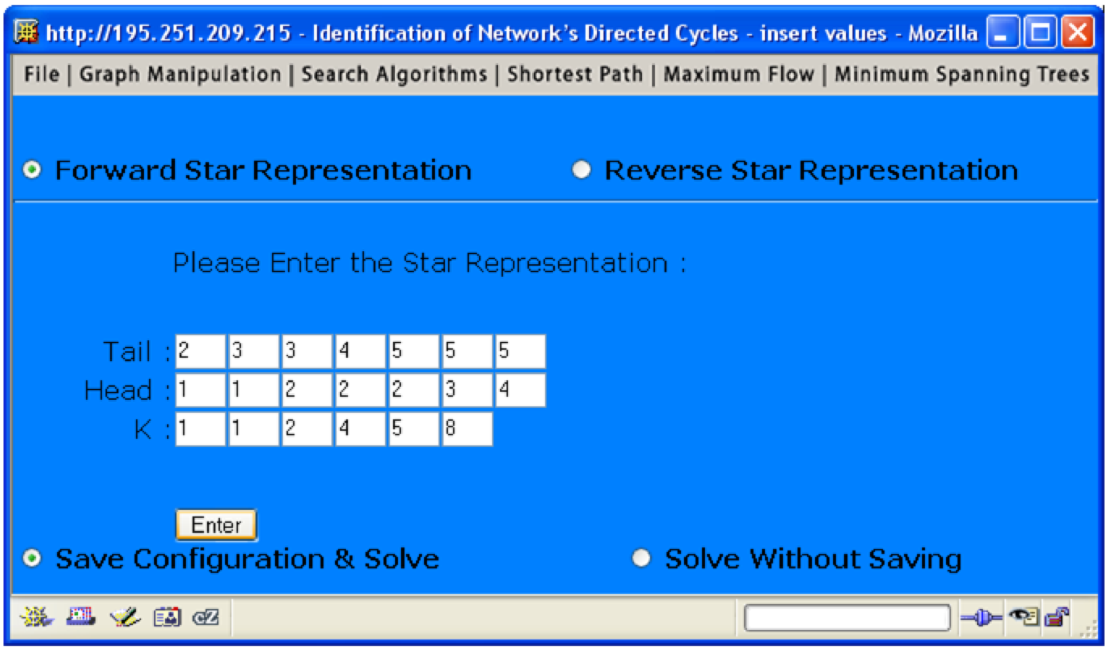
Tail :	2	3	3	4	5	5	5
Head :	1	1	2	2	2	3	4
Cost :	2	3	1	2	5	2	4
K :	1	1	2	4	5	8	

press here to retrieve the input parameter file

press here to close window

Figure 6. The output forward star representation.

The user may change and resubmit all the entered values in the matrices and the outputs will change instantly. In this way, the student can experiment with various problems. There is also the option to use the outputs in order to solve another problem. Assume the student wants to identify the existence of a directed cycle in the previous network. In that case, the student selects from the menu bar *Search Algorithms* → *Identification of a Network's Directed Cycle*. As previously, the student selects the data type (Star representation), the dimension of the problem and the specific type; Forward star representation. At this point the previous output should be entered as the new input data, see figure 7, (now there is no need to enter the cost of each arc).


http://195.251.209.215 - Identification of Network's Directed Cycles - insert values - Mozilla
File | Graph Manipulation | Search Algorithms | Shortest Path | Maximum Flow | Minimum Spanning Trees
Enter

[✓ Forward Star Representation] Reverse Star Representation

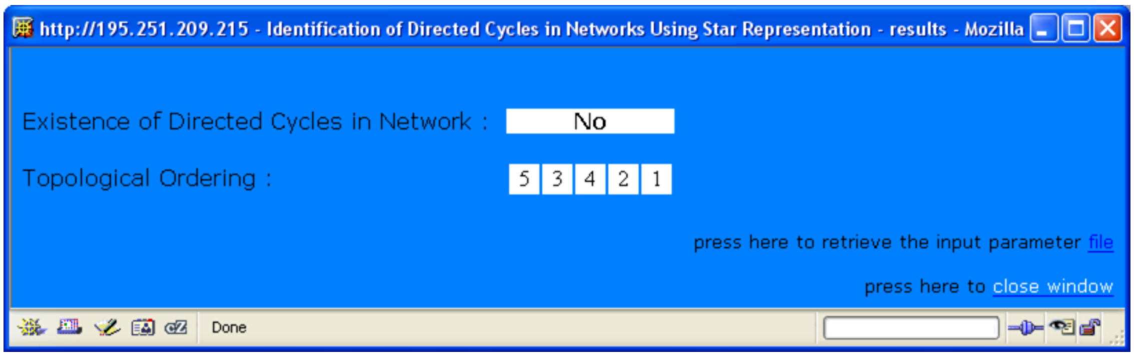
Please Enter the Star Representation :

Tail :	2	3	3	4	5	5	5
Head :	1	1	2	2	2	3	4
K :	1	1	2	4	5	8	

[✓ Save Configuration & Solve] Solve Without Saving

Figure 7. The previous outputs become now the new input data.

The user can verify that no directed cycle exists in the graph. Therefore, there exists a topological order for the network and the user can browse it in the output window of figure 8.


http://195.251.209.215 - Identification of Directed Cycles in Networks Using Star Representation - results - Mozilla
Done

Existence of Directed Cycles in Network : No

Topological Ordering : 5 3 4 2 1

press here to retrieve the input parameter file

press here to close window

Figure 8. The output window which informs the user about the potential topological order.

A node's order, $(order(i),\ i = 1, 2, \ldots, n)$, is called a *topological order*, iff $order(i) < order(j),\ \forall (i, j) \in A$. The network mentioned in this example is acyclic, since for any arc (i, j) it holds that $order(i) < order(j)$. For example, the 5th node has been labelled first and has $order(5)=1$, the 3rd node has $order(3)=2$, the 4th node has $order(4)=3$, e.t.c.

WebNetPro has been developed for educational purposes, rather than large scale optimization purposes. Therefore, the user may only submit problems with no more than 10 nodes and 45 arcs. This is done so that the results can be displayed properly on the user's screen. The imposed constraints are controlled using JavaScript (client side) and assisted by the source code of the implemented algorithms (server side).

Using this constraint violation checking method, the filled values of all elements of the Adjacency matrix or the list of arcs, namely the tail, the head and K vector are also checked. For example it is checked whether all values of the elements of the Adjacency matrix are either 0 or 1. Also it is checked that the arc costs (input) to Dijksta algorithm are non negative. Moreover, it is checked whether the values of the elements of the tail, head vectors are greater than 0 and less or equal than n.

However, not all possible scenarios can be examined. For example, consider the cases in which someone uses an algorithm for finding the shortest paths corresponding to an acyclic network, and the entered values are erroneous (which correspond to a cyclic network). In this case run time errors are unavoidable. Accordingly, assume that a student wishes to use a star representation and fails to enter the values of the *K* pointer vector correctly. This could also lead to run time errors.

Furthermore, the user may draw a directed rooted tree. The necessary input are a vector of the nodes parents, the preorder vector and a vector describing the orientation of the arcs. In order to draw a tree, the user selects the “*Tree visualization*” command from the “*Trees*” menu as it is shown in figure 9.

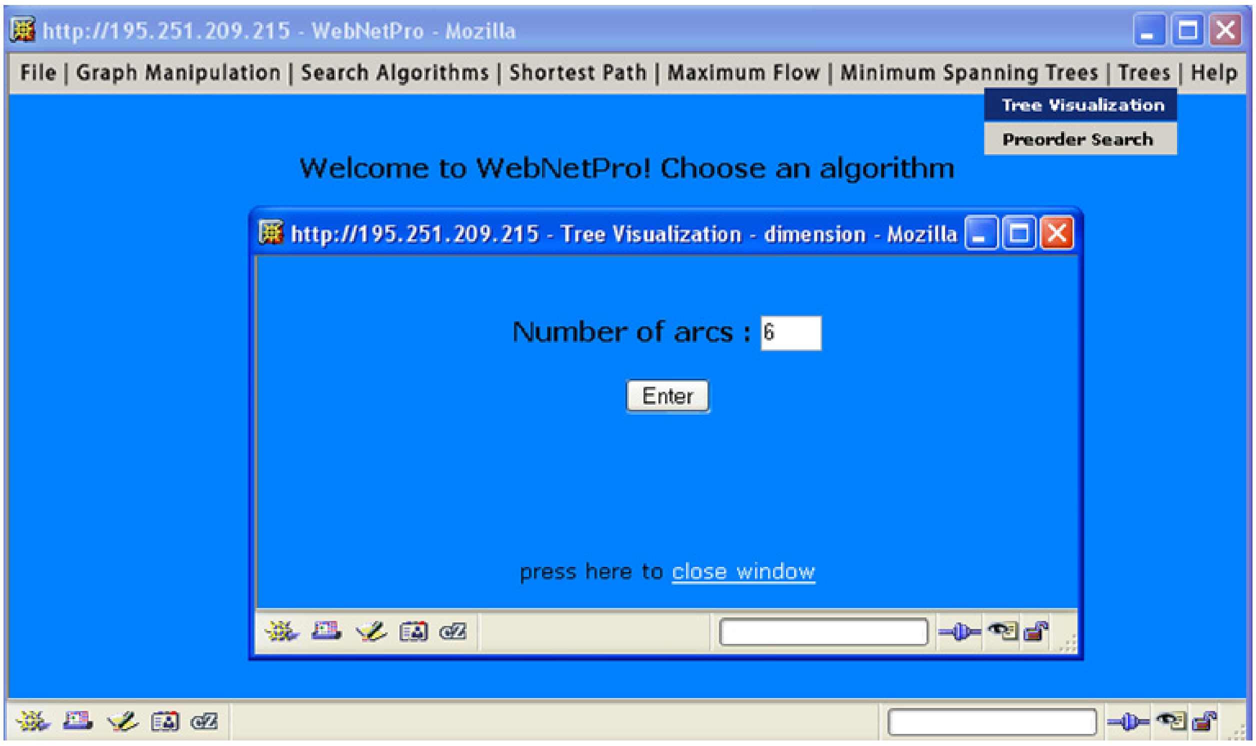
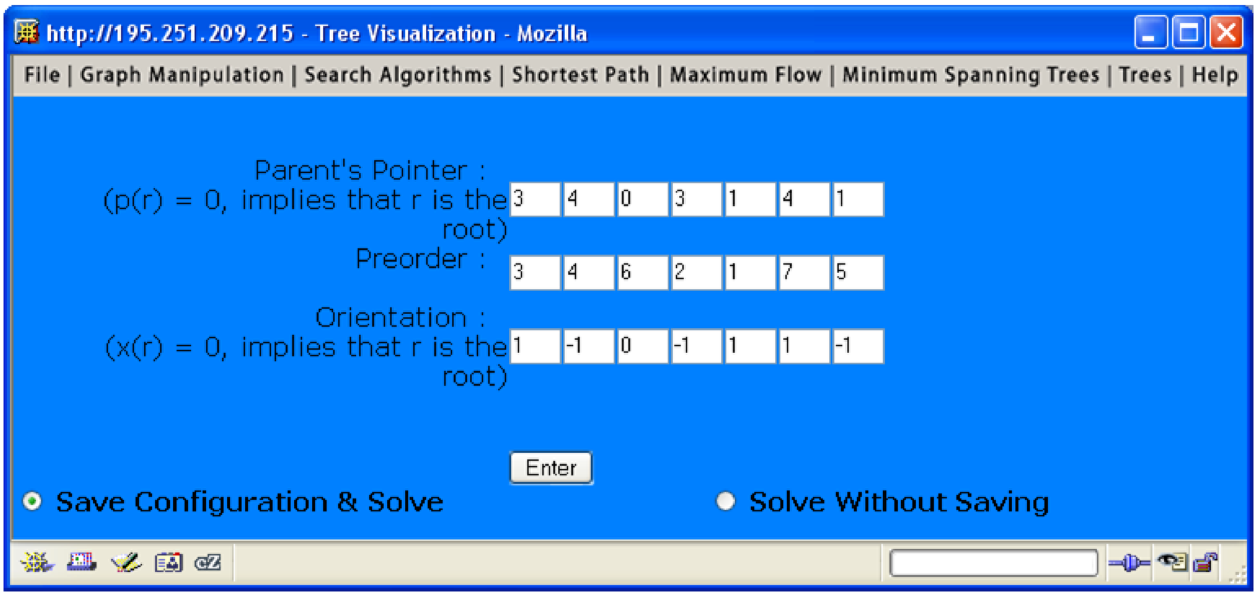

http://195.251.209.215 - WebNetPro - Mozilla
File | Graph Manipulation | Search Algorithms | Shortest Path | Maximum Flow | Minimum Spanning Trees | Trees | Help
Tree Visualization
Preorder Search
Welcome to WebNetPro! Choose an algorithm
http://195.251.209.215 - Tree Visualization - dimension - Mozilla
Number of arcs : 6
Enter
press here to close window

Figure 9. Dimension of the tree to be depicted.

Then the user may fill the values of the tree vectors as it is shown in figure 10. He should only pay attention to enter correctly the elements. The root node for the first vector, as also for the third vector, is implied with a value equal to zero. Therefore, if $p(3) = 0$ (1^{st} vector), and $x(3) = 0$ (3^{rd} vector) then it implies that the 3^{rd} node is the root of the tree.

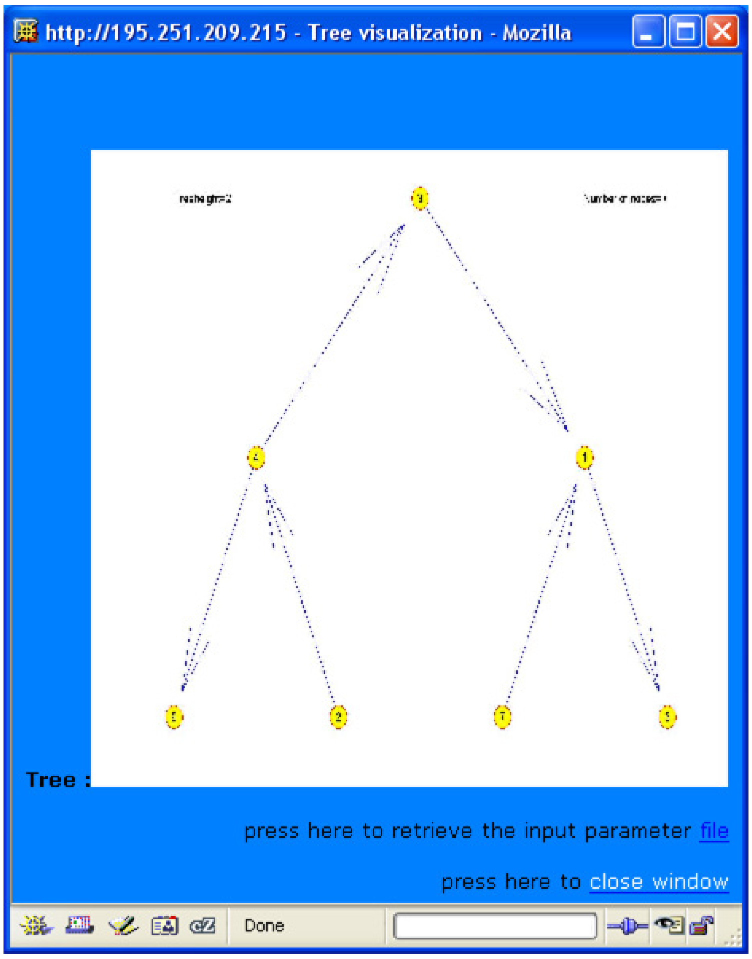

http://195.251.209.215 - Tree Visualization - Mozilla
File | Graph Manipulation | Search Algorithms | Shortest Path | Maximum Flow | Minimum Spanning Trees | Trees | Help
Enter

Parent's Pointer : (p(r) = 0, implies that r is the root)	3	4	0	3	1	4	1
Preorder :	3	4	6	2	1	7	5
Orientation : (x(r) = 0, implies that r is the root)	1	-1	0	-1	1	1	-1

[✓ Save Configuration & Solve] Solve Without Saving

Figure 10. The user must fill the appropriate cells for the tree.

The final tree visualization is depicted in figure 11. The user may save the figure, which is *.jpg format, into his client PC.



press here to retrieve the input parameter file

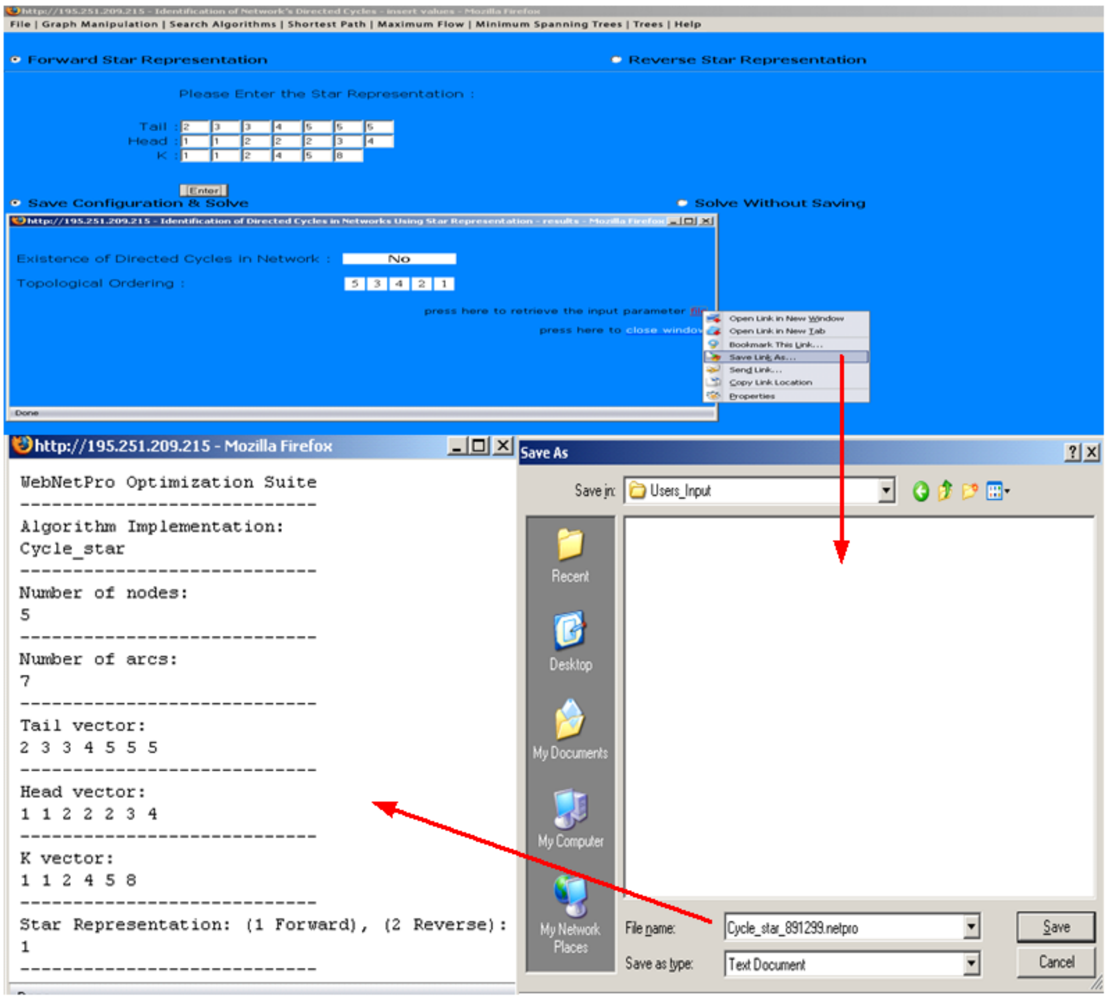
press here to close window

Figure 11. Tree visualization.

Finally, it must be mentioned that there is a help file in HTML format, which describes scholastically all the possible menu choices and walks the student through analytical examples.

6. Saving and uploading user's problem files.

WebNetPro gives user the possibility to save his own customized problem set. This way the user can re-submit any problem he created previously, without the need to re-enter all the values in the text boxes. For example, suppose the user in figure 7, had selected the radio button "*Solve and save the configuration*". Then, in the next figure 8, the user can now retrieve his problem input file by right clicking on the hyperlink "*file*" at the lower right part of the output window. By choosing "*Save link as*", (or "*Save as*" if the users browser is the Microsoft Internet Explorer) he may save his customized problem data to his client PC. The stepwise process is depicted in figure 12. Also in figure 12, anyone can see the internal structure of the file.


http://195.251.209.215 - Identification of Network's Directed Cycles - insert values - Mozilla Firefox
File | Graph Manipulation | Search Algorithms | Shortest Path | Maximum Flow | Minimum Spanning Trees | Trees | Help
Forward Star Representation
Reverse Star Representation
Please Enter the Star Representation :
Enter
Save Configuration & Solve
Solve Without Saving
http://195.251.209.215 - Identification of Directed Cycles in Networks Using Star Representation - results - Mozilla Firefox
press here to retrieve the input parameter file
press here to close window
Open Link in New Window
Open Link in New Tab
Bookmark This Link...
Save Link As...
Send Link...
Copy Link Location
Properties
Done
http://195.251.209.215 - Mozilla Firefox
Save As
Recent
Desktop
My Documents
My Computer
My Network Places
Save
Cancel

Tail :	2	3	3	4	5	5	5
Head :	1	1	2	2	2	3	4
K :	1	1	2	4	5	8	

Existence of Directed Cycles in Network : No

Topological Ordering : 5 3 4 2 1

```
WebNetPro Optimization Suite
----------------------------
Algorithm Implementation:
Cycle_star
----------------------------
Number of nodes:
5
----------------------------
Number of arcs:
7
----------------------------
Tail vector:
2 3 3 4 5 5 5
----------------------------
Head vector:
1 1 2 2 2 3 4
----------------------------
K vector:
1 1 2 4 5 8
----------------------------
Star Representation: (1 Forward), (2 Reverse):
1
----------------------------
```

Save in: Users_Input

File name: Cycle_star_891299.netpro

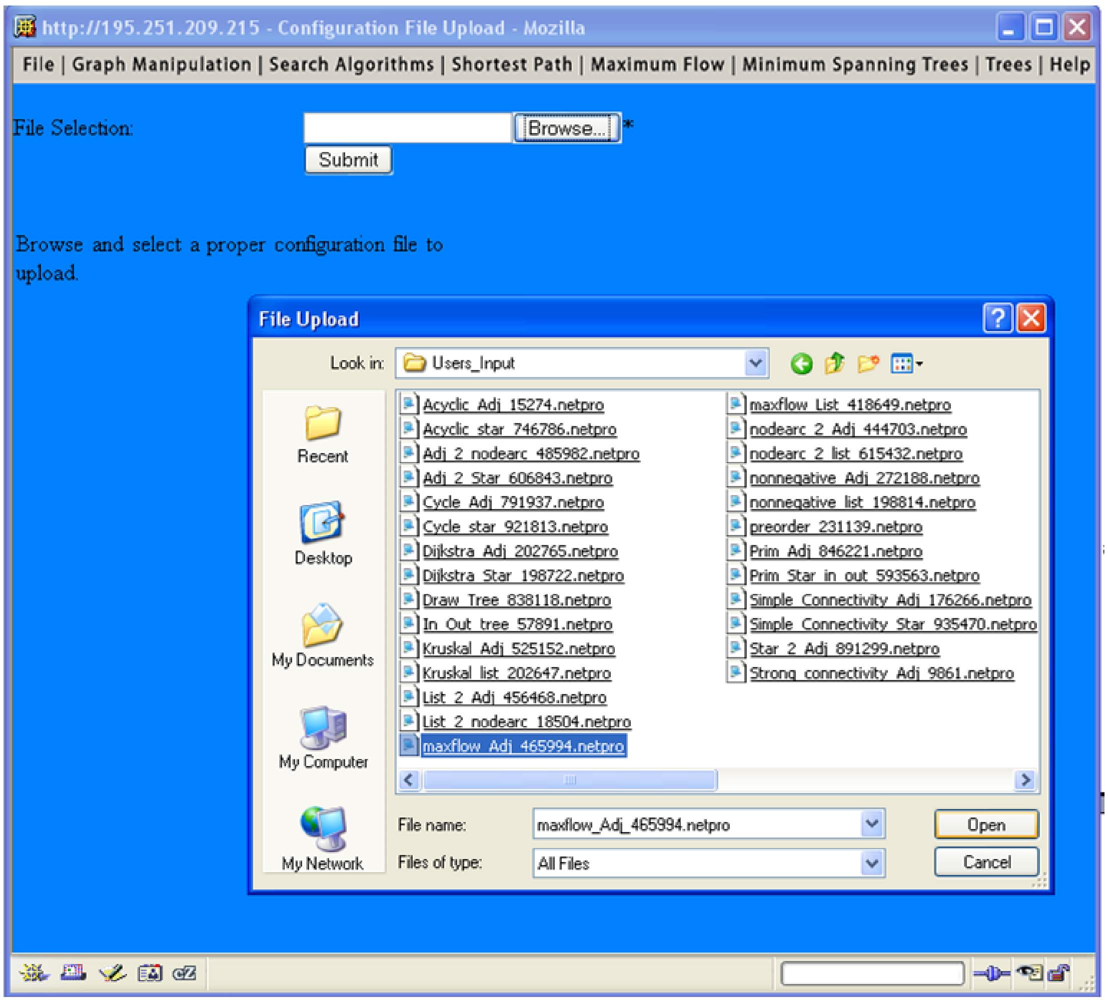
Save as type: Text Document

Figure 12. Description of the file uploading process.

WebNetPro is designed to parse the user's problem file and find out which algorithm was selected by the user in order to produce the specific file. Suppose that the user had created a problem file using the topological order algorithm. After uploading the file, the html form, corresponding to the topological order algorithm, will load again and contain the values included in the file. This is helpful, in order to prevent any error from the user. If for example a user saves an adjacency matrix consisting of 5 rows and columns and a starting node for the Dijkstra algorithm, he cannot upload it into an html form corresponding e.g. to a maximum flow algorithm implementation which uses a list of arcs or even an adjacency matrix with different dimension.

The same user – centred approach was also applied in the naming of each problem file. More specifically, each file name begins with certain keywords to assist the user in remembering the problem parameters. The last part of each file name is a random 6 digit number, just to prevent any file duplication. Finally, all the files have a *.netpro extension and therefore can be distinguished from other users data files.

Uploading files constitutes a quite simple and straight forward process. The user selects form the main Menu bar the sub – option "*Load file*" from the option "*File*" and afterwards he must find the appropriate problem file browsing his PC using the "*Browse*" button, see figure 13. Once the desired problem file has been found, the user presses the "*Submit*" button.


http://195.251.209.215 - Configuration File Upload - Mozilla
File | Graph Manipulation | Search Algorithms | Shortest Path | Maximum Flow | Minimum Spanning Trees | Trees | Help
File Selection:
Browse...
Submit
File Upload
Look in:
Recent
Desktop
My Documents
My Computer
My Network
Open
Cancel

Browse and select a proper configuration file to upload.

Look in: Users_Input

Acyclic_Adj_15274.netpro	maxflow_List_418649.netpro
Acyclic_star_746786.netpro	nodearc_2_Adj_444703.netpro
Adj_2_nodearc_485982.netpro	nodearc_2_list_615432.netpro
Adj_2_Star_606843.netpro	nonnegative_Adj_272188.netpro
Cycle_Adj_791937.netpro	nonnegative_list_198814.netpro
Cycle_star_921813.netpro	preorder_231139.netpro
Dijkstra_Adj_202765.netpro	Prim_Adj_846221.netpro
Dijkstra_Star_198722.netpro	Prim_Star_in_out_593563.netpro
Draw_Tree_838118.netpro	Simple_Connectivity_Adj_176266.netpro
In_Out_tree_57891.netpro	Simple_Connectivity_Star_935470.netpro
Kruskal_Adj_525152.netpro	Star_2_Adj_891299.netpro
Kruskal_list_202647.netpro	Strong_connectivity_Adj_9861.netpro
List_2_Adj_456468.netpro	
List_2_nodearc_18504.netpro	
maxflow_Adj_465994.netpro	

File name: maxflow_Adj_465994.netpro

Files of type: All Files

Figure 13. Selection of the file to upload.

Now, the problem file has been uploaded to the server and it only needs to parse the file and pass the data parameters into the appropriate application html forms. This is done, after the user presses the "*Configure*" button. Notice that the previous "*Submit*" button has now been replaced with the "*Configure*" button. At this point the, previously saved, problem set has been successfully loaded and the user may solve it again or modify it if he wants.

7. Conclusions and future work

In this paper we present an on - line software application which constitutes a powerful educational tool. The main characteristics of this tool are that it is user - friendly and has a flexible G.U.I. Its capabilities make it attractive in the teaching of Linear Network Programming. The design of this tool is completely customized and can be utilized in various scientific areas.

The proposed tool does not demonstrate the computational steps inside an algorithm. It helps a student to verify his/her calculated solutions or convert one network representation to another. However, animation is not included because it was out of the scope of this work. Furthermore, all applications that incorporate animation require additional pre-installed proprietary software. On the contrary, WebNetPro relies on Matlab web server and PHP scripts, which run exclusively on the server PC, posing not further requirements. Updates or newer versions are completely transparent to the student as using WebNetPro is similar to visiting an ordinary web page.

It is our belief, that the discussed software should be best used by the students in cooperation with their instructor. It is better to combine a lecture regarding the theoretical aspects of any algorithm and a supplementary tool to verify the understanding of an algorithm. However, it is easier to find a book explaining either the workings inside a graph algorithm or the diversifications of the variety of network representations, than to have a free, easy to use, on - line supplementary tool which can assist any student.

In order to estimate the educational value of this tool, an on - line questionnaire could be embedded in the application main web page. Following this direction, WebNetPro is schedule to be thoroughly evaluated inside the classrooms during the lectures, for the needs of our Department’s courses. Students form WebNetPro’s main target group. Thus, their comments and suggestions are of invaluable importance for further improving WebNetPro.

It is more important to determine how the remote users - students, (probably from their home), make use of this specific software, rather than just explaining to them what might be the potential benefits, see [16, 17]. Other researchers show how to use algorithm animations in learning environments, see [18]. This technology seems to be very promising for Distance Learning. However, it should be mentioned that research has proved that pedagogical advantages can only partially be attributed to Algorithm Visualization Technology, see [19].

A future work is to integrate *LinPro* [20]. *LinPro* is also implemented in the Matlab computing environment and it is an educational platform for Linear Programming. Both the packages can constitute a useful Mathematical Programming suite, mainly for educational purposes.

7. Acknowledgements

The authors would like to thank the anonymous referees for helpful suggestions and comments.

Research supported by European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II), and particularly the program HRAKLEITOS.

References

[1] Ahuja, K.R., Magnanti, L.T. and Orlin, B.J., 1993, *Network Flows: Theory, Algorithms and Applications* (Prentice Hall).

[2] Gross, J.L. and Yellen, J., 1998, *Graph Theory and its Applications* (CRC Press).

[3] Smith, G.G. and Ferguson, D., 2004, Diagrams and math notation in e-learning: growing pains of a new generation, *International Journal of Mathematical Education in Sciences and Technology*, **35**(5), 681–695.

[4] 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 .

[5] Dosios, K., Paparrizos, K., Samaras, N. and Sifaleras, A., 2003, NetPro, an Educational Platform for Network Optimization, in Proc. of 16th National Conference of Hellenic Operations Research Society (HELORS), Larissa, Greece, **1**, 287–295.

[6] Evans, J.R. and Minieka, E., 1992, *Optimization Algorithms for Networks and Graphs*, 2nd Edition, (Marcel Dekker).

[7] Kennington, R.V. and Helgason, J.L., 1980, *Algorithms for Network Programming* (John Wiley), 244–256. Available software at http://www-neos.mcs.anl.gov/neos/solvers/LNO:NETFLO

[8] Available at http://campuscgi.princeton.edu/~rvdb/JAVA/network/nettool/netsimp.html**,** Vanderbei, R., Personal web page, (Princeton University).

[9] Available at http://www.princeton.edu/~rvdb/JAVA/CIV201/shortpaths/shortpaths.html, Vanderbei, R., Personal web page, (Princeton University).

[10] Available at http://carbon.cudenver.edu/~hgreenbe/sessions/dijkstra/DijkstraApplet.html,

[11] Dagdilelis, V. and Satratzemi, M., 1998, Didagraph: A Software for Teaching Graph Theory Algorithms, *SIGSE Bulletin*, (ACM), **30**(3), 64–68.

[12] Lazaridis, V, Paparrizos, K., Samaras, N. and Sifaleras, A., Visual LinProg: A Web-based Educational Software for Linear Programming, Accepted for publication in *Computer Applications in Engineering Education*.

[13] Colgan, L., 2000, Matlab in first year engineering mathematics, International *Journal of Mathematical Education in Sciences and Technology*, **31**(1), 15–25.

[14] Dunn, P.K. and Harman, C., 2002, Calculus demonstrations using MATLAB, *International Journal of Mathematical Education in Sciences and Technology*, **33(**4), 584–596.

[15] Marchand, P. and Holland, O.T., 2002, *Graphics and GUIs with MATLAB*, 3rd. Edition, (CHAPMAN and HALL/ CRC).

[16] Hundhausen, D.C., Douglas, A.S. and Stasko, T.J., 2002, A meta-study of algorithm visualization effectiveness, *Journal of Visual Languages and Computing*, **13**(3), 259–290.

[17] Lawrence, A.W., Badre, A.N. and Stasko, J.T., 1994, Empirically evaluating the use of animations to teach algorithms, in Proc. of the IEEE Symposium on Visual Languages, U.S.A.

[18] Kehoe, C., Stasko, J.T. and Taylor, A., 2001, Rethinking the evaluation of algorithm animations as learning aids: an observational study, *International Journal of Human-Computer Studies*, **54**(2), 265–284.

[19] Byrne, M.D., Catrambone, R. and Stasko, J.T., 1999, Evaluating animations as student’s aids in learning computer algorithms, *Computers and Education*, **33**(4), 253–278.

[20] Dosios, K., Paparrizos, K., Papatzikos, N. and Sifaleras, A., 2002, LinPro, an Educational Informational System for Linear Programming, presented at the 15th National Conference of Hellenic Operations Research Society (HELORS), Tripoli, Greece.