Visualization software of the network exterior primal simplex algorithm for the minimum cost network flow problem

D. Andreou[†], K. Paparrizos[*], N. Samaras[*‡] and A. Sifaleras[*]

Abstract

The Minimum Cost Network Flow Problem (MCNFP) constitutes perhaps the most important of the research area of Network Optimization. Recently a new category of algorithms for the MCNFP have been developed. These algorithms belong to a special "exterior type" simplex category and they feature significant tree modifications. This paper presents a new didactic tool for the teaching of this type of algorithms. The proposed educational software can be used in courses like "Graph Theory" or "Combinatorial Optimization". This tool has been implemented using the Java Programming language and it is platform independed. It constitutes a friendly application, for the instructor, as also for the novice student. Furthermore, its use is analytically shown through an illustrative example. Benefits and drawbacks are thoroughly described in order to support the significance of this visualization tool in computer-aided education and also possible future work is discussed.

Keywords: Operational Research, Combinatorial Optimization, Web-based Educational Software.

1. Introduction

The development of efficient algorithmic solutions is the main scope of the research area of Mathematical Programming. Linear Programming constitutes a special subfield of Mathematical Programming which only deals with strictly linear problems. Following the previous classification, Linear Network Programming can be viewed as a subset of Linear Programming. More precisely, Linear Network Programming focuses mainly on linear problems which can be modeled using special data structures such as networks or graphs. The MCNFP constitutes a wide category of problems; perhaps the most important of the research area of Linear Network Programming. There exists a plethora of every-day applications, in Informatics,

[†] Department of Computer Science and Engineering, of University of Crete, P.O. Box 2208, Heraklion, Crete, GR-714 09 Greece, andreou@csd.uoc.gr.
[*] Department of Applied Informatics, of University of Macedonia, Postal Code 54006, 156 Egnatias Str., Thessaloniki, Greece, {paparriz,samaras,sifalera}@uom.gr.
[‡] Corresponding author, Tel: +30-2310-891866.

Constructions, Telecommunications sector e.t.c, which can be mathematically formulated using the MCNFP. Many of them can be found in [Ahuja et. al. (1995)] and in [Glover et. al. (1992)]. Furthermore, most curricula in Computer Science Departments contain courses related to Graph Theory or Combinatorial Optimization. However, Network Optimization algorithms and especially their complexity aspect, is regarded as a difficult topic in Theoretical Computer Science.

Recently a new algorithm, belonging to a special "exterior simplex type" category, for the MCNFP was developed by [Paparrizos et. al., (2008)]. Instead of the long expression "Network Exterior Point Simplex Algorithm for the MCNFP", the discussed algorithm will be referred with the acronym "NEPSA". The development of a new educational software capable of providing an animating step-by-step solution would constitute a useful teaching tool. Therefore, the implementation of such a didactic tool for NEPSA was a strong motivation. This is the first time this visualization is being made.

It is well known that exposure to interactive visualization tools impact student comprehension of the material in a positive manner. Is the teaching process assisted by visual and interactive tools, then the students obtain a deeper understanding of the subject being taught rather than abstract theoretical knowledge. Due to these reasons, eLearning technologies have become a very active research field for many scientists like [Andreou et. al. (2005)], [Lazaridis et. al. (2003)], [Lazaridis et. al. (2007)], and [Smith and Ferguson (2004)]. Moreover, nowadays the development of the broadband services and infrastructures, see [Varkas et. al. (2005)], open new horizons and change the way of communication. They can efficiently facilitate eLearning applications, for example as the above web - based applications, and educational programmes in general.

Algorithm Visualization (AV) presents an algorithms execution, using sequentially graphical snapshots, the depiction of which is controlled by the user, as described in [Naps (1997)]. Specifically, AV usually displays the state of the problem at each iteration of the algorithm execution and helps someone to understand the actions need to be made between two successive iterations. A plethora of approaches exists to deliver AV over the Web. The restrictive factor in all those approaches was the system dependency of the AV software, as described in [Naps (1996)]. The capability of data structure visualizations of the Java language, see for example [Hendrix et. al. (2004)], and the system independence have nowadays override these difficulties and seem to be very promising for the Open and Distance Learning.

This pedagogical application is not the only software of this kind. Many other educational programs exist for teaching several mathematics and computer science topics. For example there have been developed educational software for graph theory by [Paparrizos et. al. (2004)], for network optimization by [Dosios et. al. (2003)] and by [Papamanthou et. al. (2005)], for number theory by [Sinclair et. al. (2004)], for mathematical proofs by [Luengo (2005)] and for mathematical programming by [Karagiannis et. al. (2006)]; each of them of course, has its own merits and demerits. NetPro project by [Dosios et. al. (2003)], is a stand alone application for network optimization. NetPro's main drawback, as also in [Paparrizos et. al. (2004)], is that in order to be used, MATLAB should have been previously installed in the users' computer. Number Worlds on the other hand, by [Sinclair et. al. (2004)] is a computer-based learning environment called, that was designed to support the exploration of elementary number theory concepts. Moreover, the Cabri-Euclide program, by [Luengo (2005)], is a didactic microworld for learning proof construction.

There is extensive research in the Web that deals with the evaluation of the pedagogic efficiency of such learning tools, as in [Almstrum et. al. (2005)]. Some studies suggest that algorithm visualizations will not benefit “novice” students, who just started to learn a new topic, but will rather benefit the more advanced students. Recent studies, for example by [Hundhausen et. al. (2002)] and [Saraiya et. al. (2004)], show the effective features of Algorithm Visualizations. On the other hand, although having students program their own visualizations and animations can be very valuable, it can also be very time - consuming endeavor.

The work, described in this paper, enables the students to be more “active” in watching the visualization. Cordova in his paper [Cordova (2003)] created appropriate quizzes to measure the student's understanding of the concepts before and after they had participated in certain visualization's exercises. The results shown that students, who participated in interactive visualization exercises, have scored significantly higher on the quizzes than students who didn't participate. Furthermore, recent studies by [Saraiya et. al. (2004)], show that accompanying textual explanations are absolutely crucial to the effective instructional use of AV. Their recommendation is, that it would be better to provide the student textual information at every step of the algorithm and not merely to present students with a non-stop animation.

Taking the above considerations into account, this Java application was designed to have certain pedagogical features. First of all, it features two modes for presenting the algorithm visualization to the students. The first mode is that the animation may run throughout at one step, while the second mode is step by step. Using the latter mode, students can take deep insight of all the necessary computations for the solution of a MCNFP. Furthermore, the proposed software allows students to input their own fully customizable problem sets. This way, students can get more learning benefits, rather than presenting them only a predefined MCNFP instance. Moreover, textual information is presented inside the application, as for example the cost of each arc next to the specific arc, or the supply of each node next to the specific node, e.t.c. At every step of the algorithm, the student can see all the necessary textual information regarding the state of the variables used by the algorithm (i.e. what is the current basis tree solution of the problem, the dual variables and more). All the data structure values are dynamically produced simultaneously with the algorithm execution. Finally, the algorithm pseudo code is also provided on the right part of the user screen. It should be pointed out that at every iteration of the algorithm, the appropriate line of the pseudo code is highlighted. This is a significant factor because it help's the user to relate state changes directly to the algorithm's process.

The paper is organized as follows. Following this introduction, in Section 2 we give some necessary notations and definitions of the algorithm being animated. An analytic description of NEPSA is presented in Section 3. In Section 4 we describe the graphical interface and discuss about implementation issues and how can any instructor use it. An illustrated example is given in Section 5, while in Section 6 we conclude and possible future work is mentioned.

2. Algorithmic Notations and Definitions

Let $G = (N, A)$ be a directed network with n nodes and m arcs, where N and A are the sets of nodes and arcs respectively. Each arc $(i,j) \in A$ has a cost c_{ij} which denotes the unit shipping cost along arc (i,j). In this paper, NEPSA algorithm is applied to un-capacitated MCNFPs. This means that $l_{ij} \leq x_{ij} \leq u_{ij}$, where $l_{ij} = 0$ and $u_{ij} = +\infty$. Associated with each arc (i,j) is also an amount x_{ij} of flow on the arc. Contrary to the classical network simplex algorithm, x_{ij} might become negative. We associate with each node $i \in N$ a number b_i which indicates its available amount of supply or demand. Node i will be called a source, sink or transhipment node, depending upon whether $b(i) > 0$, $b(i) < 0$ or $b(i) = 0$ respectively. We make the assumption that G is a balanced network, that is $\sum_{i=1}^{n+1} b(i) = 0$. The mathematical formulation of the MCNFP is the following:

$$\begin{aligned} &\min \sum_{\{(i,j)\in A\}} c_{ij}x_{ij} \\ &\text{s.t} \sum_{\{k:(i,k)\in A\}} x_{ik} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b_i, \quad i \in N \\ &x_{ij} \geq 0, \quad (i,j) \in A \end{aligned} \qquad \textbf{(NP.1)}$$

In the above formulation, constraints of type $\sum x_{ki} - \sum x_{jk} = b_k$ are known as the flow conservation equations, while constraints of type $0 \leq x_{ij} \leq +\infty$ are called the flow capacity constraints. A vector of dual variables for the network G is a vector w with n components, where w_i is associated with node i. For each vector w, the reduced costs s_{ij} are defined as $s_{ij} = c_{ij} - w_i + w_j$. Network Simplex type algorithms compute basic solutions x and (w,s) which are complementary.

Furthermore, in this paper, notations such as paths, directed paths, cycles, directed cycles and trees, are defined as in [Bazaraa et. al. (2005)]. The basic solution for the MCNFP can be represented using a tree. Each basis tree will be denoted by *T*. Each tree of a MCNFP consists of a flow vector *x*, which is the unique solution of the problem (NP.1). The basic flow of each *T*, will be denoted by *x(T)*. We say that *T* is a feasible tree, iff $x(T) \geq 0$, otherwise *T* is an infeasible tree.

3. Algorithm Description

NEPSA firstly chooses a leaving arc and secondly an entering arc. For simplicity reasons, from now on the leaving arc will be denoted with *(k,l)* while the entering arc with *(g,h)*. We also denote by C^t the cycle that would have been created at the tth iteration, if prior to the leaving arc, the entering arc have joined the basic tree *T*. When an arc *(k,l)* leaves the basis tree, then the current *T* is divided into two sub-trees. The sub-tree which includes the *k* node will be denoted by T^+, while the other by T^-. All the nonbasic arcs *(i,j)* will either not join these two sub-trees, or they will join them with two ways. Firstly, the node $i \in T^+$and the node $j \in T^-$ and secondly vice versa. The value of a ψ variable at the tth iteration will be denoted by $\psi^{(t)}$.

At *Step 0*, an initial feasible tree can be constructed using the well known big *M* method. The *A* set of arcs is partitioned in two subsets. The first subset corresponds to the basic arcs, while the second subset corresponds to the non basic arcs. The latter subset will be further partitioned in the following two sets *P* and *Q*:

$$P = \{(i,j)\}, (i,j) \notin T : s_{ij} < 0, \qquad Q = \{(i,j)\}, (i,j) \notin T : s_{ij} \geq 0 \quad \textbf{(1)}$$

The initial *A* set of arcs is finally partitioned as *A = [T P Q]*. Also vector h_{gh} will denote the representation of each non basic arc, for example *(g,h)*, in terms of the basic arcs. Each non basic arc *(g,h)* corresponds to a column vector h_{gh}. Each element of this vector corresponds to a basic arc, for example *(k,l)*. More details about the calculation of h_{gh} can be found in [Bazaraa et. al. (2005)]. Furthermore, vector *d* will denote a "complemental" flow which assists the algorithm to compute the second feasible but not always basic flow. According to the partition of *A*, vector *d* is partitioned as $d = \{d(T), d(P), d(Q)\}$. The elements of *d* are computed using the following relations, where |*P*| denotes the cardinality number of the *P* set.

$$d(T) = -\sum_{(i,j)\in P} h_{ij}, d(P) = (\lambda_1, \lambda_2, ..., \lambda_{|P|}), d(Q) = (0,0,...,0) \quad \textbf{(2)}$$

Elements of vector $\lambda = (\lambda_1, \lambda_2, ..., \lambda_{|P|})$ can have any value, so long as vector *d* is appropriate to compute the second feasible but not always basic flow. In our implementation, λ is a unit vector. Following in Table 1, there is a formal description of NEPSA.

Table 1. Algorithm's pseudo – code.

```
Algorithm Network Exterior Point Simplex Algorithm for the MCNFP.
 1  Step 0 (Initializations)
 2  Construct a feasible tree $T$ (Big M method)
 3  Compute $x(T)$, $w$, $s$
 4  Find the sets $P$ and $Q$
 5  Compute $d$, using relation (2)
 6  Step 1 (Test of Optimality)
 7  do while ($P \neq \varnothing$)
 8      if ($d(T) \geq 0$) then
 9          EXIT (The problem is unbounded)
10      else
11          Step 2 (Selection of leaving arc)
12          Compute $a$, using relation (3)
13          Choose the leaving arc $(k,l)$
14          Compute feasible flow $y$
15          Step 3 (Selection of entering arc)
16          Compute $\theta_1$, $\theta_2$, using relations (4)
17          Choose the entering arc $(g,h)$
18          Step 4 (Pivoting)
19          Update $T, P, Q, x, s, d$.
20      end if
21  end do
22  EXIT (The problem is optimal)
```

At Step 1 there is the optimality check, while at *Step 2* there is the selection of the leaving arc, using the following minimum ratio test:

$$a = \frac{x_{kl}}{-d_{kl}} = \min\left\{ \frac{x_{ij}}{-d_{ij}} : (i,j) \in T, d_{ij} < 0 \right\} \tag{3}$$

Afterwards, the second flow y is calculated using the relation $y = x + ad$. This second flow remains feasible, but not always basic, since y corresponds to network and not to tree. At *Step 3*, the entering arc is chosen using the following minimum ratio tests:

$$\theta_1 = -s_{p_1p_2} = \min\left\{-s_{ij} : h_{ij}(k,l) = 1, (i,j) \in P\right\},\ \theta_2 = s_{q_1q_2} \min\left\{s_{ij} : h_{ij}(k,l) = -1, (i,j) \in Q\right\} \tag{4}$$

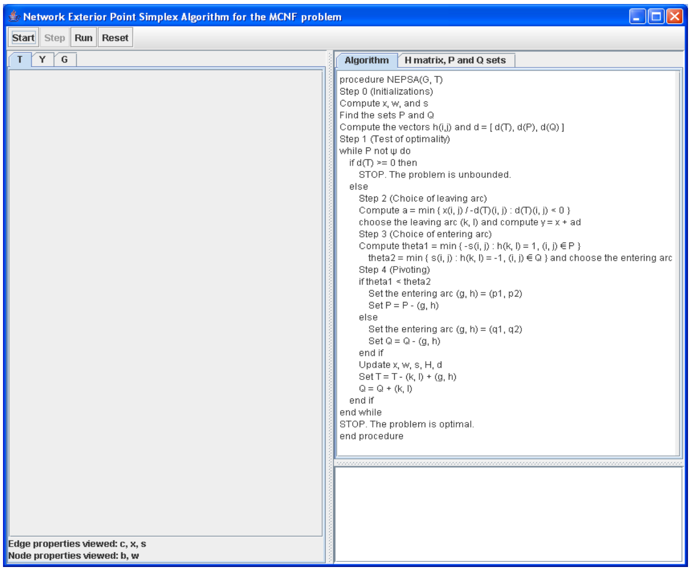
If $\theta_1 \leq \theta_2$, then the pivot will be called "*Type A iteration*", while otherwise the pivot will be called "*Type B iteration*". Afterwards, all the vectors x_{ij}, s_{ij}, h_{ij} and d_{ij} are updated. More analytical description and also all the analytical proofs of correctness of the NEPSA algorithm, can be found in [Paparrizos et. al., (2008)].

4. Description of the Graphical User Interface (GUI) and Implementation Issues

This is the first time that such an analytical animation of the NEPSA is being implemented. All the iterations as also the creation of the initial feasible tree are fully animated. The main goal is the development of an educational interactive tool which may be used firstly for the teaching inside the classroom/laboratory and secondly as an educational supplement at the student's home. The application gives the possibility to the student to reach the arithmetic solution of the MCNFP, while avoiding many hard calculations. This way he can focus on working on the thorough understanding of the solution procedure. Furthermore, there is the capability of solving many and difficult user defined MCNFP's, while eliminating the numerical errors. Emphasis was given at the final design of the application, to adopt an easy graphical interface for the user and not burden him with the additional learning of a complicated environment. Also, the user should have the ability to easily execute the proposed application, without to install anything in his PC.

The main screen of the NEPSA animation software is depicted below in Figure 1. The proposed on-line tool can be accessed via the following web address:

http://macedonia.uom.gr/~sifalera/ORIJ


Network Exterior Point Simplex Algorithm for the MCNF problem
Start
Step
Run
Reset
T
Y
G
Algorithm
H matrix, P and Q sets
Edge properties viewed: c, x, s
Node properties viewed: b, w

```
procedure NEPSA(G, T)
Step 0 (Initializations)
Compute x, w, and s
Find the sets P and Q
Compute the vectors h(i,j) and d = [ d(T), d(P), d(Q) ]
Step 1 (Test of optimality)
while P not ψ do
  if d(T) >= 0 then
    STOP. The problem is unbounded.
  else
    Step 2 (Choice of leaving arc)
    Compute a = min { x(i, j) / -d(T)(i, j) : d(T)(i, j) < 0 }
    choose the leaving arc (k, l) and compute y = x + ad
    Step 3 (Choice of entering arc)
    Compute theta1 = min { -s(i, j) : h(k, l) = 1, (i, j) ∈ P }
      theta2 = min { s(i, j) : h(k, l) = -1, (i, j) ∈ Q } and choose the entering arc
    Step 4 (Pivoting)
    if theta1 < theta2
      Set the entering arc (g, h) = (p1, p2)
      Set P = P - (g, h)
    else
      Set the entering arc (g, h) = (q1, q2)
      Set Q = Q - (g, h)
    end if
    Update x, w, s, H, d
    Set T = T - (k, l) + (g, h)
    Q = Q + (k, l)
  end if
end while
STOP. The problem is optimal.
end procedure
```

Figure 1. The main window of the proposed Java application

In the Menu bar, user can press the "Start" button in order to start a problem by defining the appropriate network. After the successful insertion of a problem the user may choose either to execute the animated solution step by step, by pressing the "Step" button, or execute the algorithm throughout at one step, by pressing the "Run" button. Finally, there is also the capability to reset the problem and define a new one, by pressing the "Reset" button.

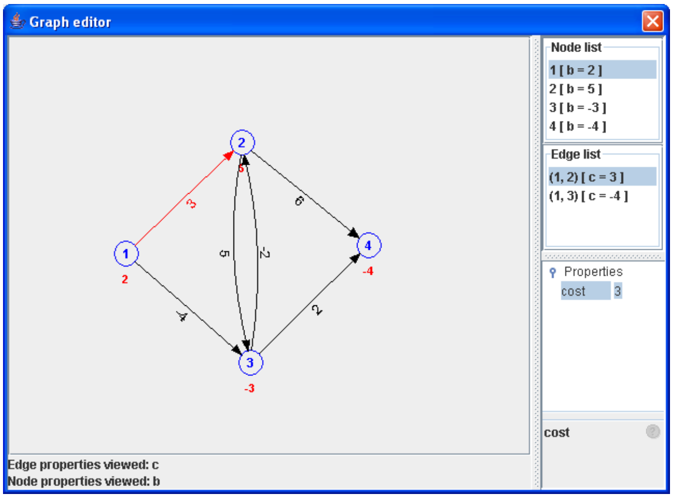
All the animations take place in the left frame of the main window, which is divided into three tabs. The first tab contains the basis tree *T*. The middle tab contains the feasible flow *Y* and finally in the third tab the initial network G is shown. The right frame of the main window is divided into two sub – frames; the upper and the lower the upper frame is again divided into two tabs. In the first tab the student can see the pseudo code proprietary highlighted in yellow color. In the second tab the *P* and *Q* sets are presented, as also the values of the column vectors h_{gh} corresponding to each non basic arc *(g,h)*. In the lower frame the student can see all the computations needed for the data structures as also all the variables updates. After the user selects "Start", the Graph Editor opens as it is shown in Figure 2.

The student may enter any node by double left clicking on this window. An (i,j) arc can be inserted by a simple drag and drop operation. Dragging the mouse, using the right mouse button, from the tail node *i* to the head node *j*. Any node, as also it's connecting arcs, may be moved to another place in the window just by left clicking on the node and dragging it to the desired position. Finally, all the necessary numerical input like the arc's costs or node's supplies or demands may be typed in the right windows. More specifically, after selecting the specific node or arc from the "Node list" or the "Edge list", then the values are filled in its properties window just below the previous two panels. These input are immediately "loaded" on the graph *G*. More details about the usage of the NEPSA visualization will be presented in the following Section, by solving an illustrative example.

Finally, the proposed Java application has been implemented in Java, compiled with Java 2 SDK 1.5.0_04 package and executed using the J2SE Runtime Environment (JRE) 1.5.0_04. This animation Java program has been tested extensively on a Windows XP Professional system, with Service Pack 2 and Java Runtime 1.5.0_04 installed. The Java 2 Runtime Environment allows anyone 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 standalone Java application, s/he must have the Java Runtime Environment previously installed on his/her system. The Swing package has been mainly used for all graphics rendering.

5. An Illustrated Example

In this section a demonstration of an animated solution will be presented, using the following MCNFP which is shown in Figure 2. The numbers next to the arcs denote their corresponding cost c_{ij}, while the numbers next to the nodes denote their corresponding product supply or demand b_i.



Node list
1 [b = 2]
2 [b = 5]
3 [b = -3]
4 [b = -4]

Edge list
(1, 2) [c = 3]
(1, 3) [c = -4]

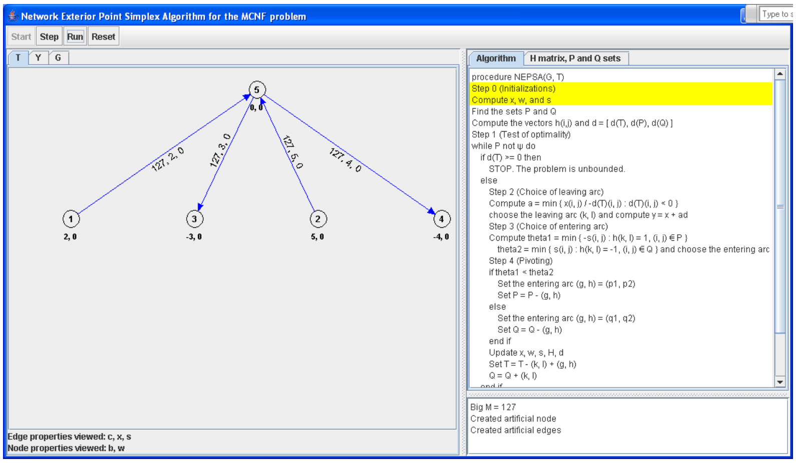
cost 3

cost

Edge properties viewed: c
Node properties viewed: b

Figure 2. The initial graph in the Graph Editor

After the entering of the network *G*, the user exits the Graph Editor using the Windows upper right "exit" button being at the title bar. For simplicity reasons we assume that the user chooses the step by step solution. At *Step 0*, the big M method is applied to the initial network. At the first click of the "step" button a new artificial node, with label "5", is created. Later at the second "click" the four new artificial arcs are created and the big M as also the *x*, *w* and *s* vectors are computed. At the third "click" the network is balanced as in the following Figure 3.



```
procedure NEPSA(G, T)
Step 0 (Initializations)
Compute x, w, and s
Find the sets P and Q
Compute the vectors h(i,j) and d = [ d(T), d(P), d(Q) ]
Step 1 (Test of optimality)
while P not ψ do
   if d(T) >= 0 then
      STOP. The problem is unbounded.
   else
      Step 2 (Choice of leaving arc)
      Compute a = min { x(i, j) / -d(T)(i, j) : d(T)(i, j) < 0 }
      choose the leaving arc (k, l) and compute y = x + ad
      Step 3 (Choice of entering arc)
      Compute theta1 = min { -s(i, j) : h(k, l) = 1, (i, j) ∈ P }
         theta2 = min { s(i, j) : h(k, l) = -1, (i, j) ∈ Q } and choose the entering arc
      Step 4 (Pivoting)
      if theta1 < theta2
         Set the entering arc (g, h) = (p1, p2)
         Set P = P - (g, h)
      else
         Set the entering arc (g, h) = (q1, q2)
         Set Q = Q - (g, h)
      end if
      Update x, w, s, H, d
      Set T = T - (k, l) + (g, h)
      Q = Q + (k, l)
   end if
```

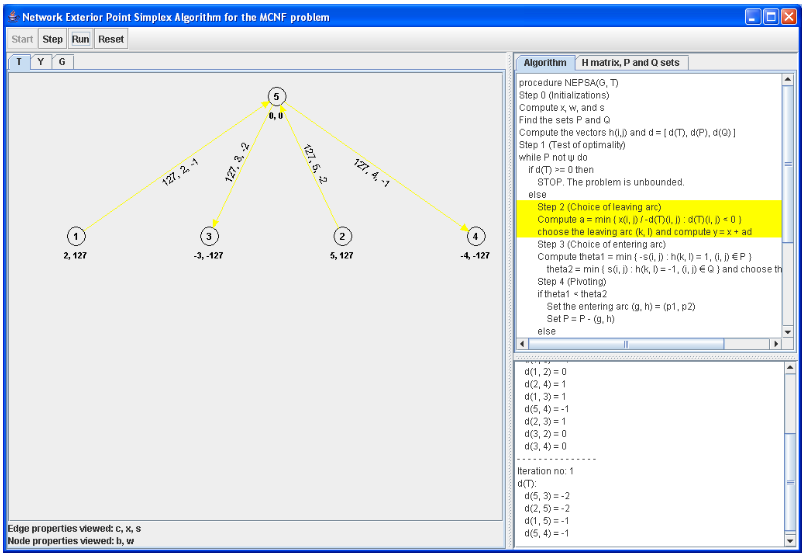
```
Big M = 127
Created artificial node
Created artificial edges
```

Edge properties viewed: c, x, s
Node properties viewed: b, w

Figure 3. The augmented network

At *Step 0*, NEPSA algorithm starts with the initial feasible tree, which is depicted in blue colour in the above Figure 3. The basic flow vector is $x^{(1)} = (x_{15}^{(1)}, x_{25}^{(1)}, x_{53}^{(1)}, x_{54}^{(1)}) \Rightarrow x^{(1)} = (2,5,3,4)$, the dual variables vector is $w^{(1)} = (w_1^{(1)}, w_2^{(1)}, w_3^{(1)}, w_4^{(1)}, w_5^{(1)}) \Rightarrow w^{(1)} = (127,127,-127,-127,0)$ and the reduced costs vector is $s^{(1)} = (s_{13}^{(1)}, s_{23}^{(1)}, s_{24}^{(1)}, s_{12}^{(1)}, s_{32}^{(1)}, s_{34}^{(1)}) \Rightarrow s^{(1)} = (-258,-249,-248,3,252,2)$. Therefore, $P^{(1)} = \{(1,3),(2,3),(2,4)\}$ and $Q^{(1)} = \{(1,2),(3,2),(3,4)\}$.

At *Step 1*, it holds that $P^{(1)} \neq \varnothing$, therefore the algorithm doesn't terminate. Since, $d^{(1)}(T) = \{d_{15}^{(1)}, d_{25}^{(1)}, d_{53}^{(1)}, d_{54}^{(1)}\} \Rightarrow d^{(1)} = \{-1,-2,-2,-1\}$, NEPSA proceeds to *Step 2*. At each iteration, the candidate leaving arcs are yellow colored as it is shown in the following Figure 4. It should be also pointed out that all the computations and the updated values are presented at the lower sub – frame of the right frame.

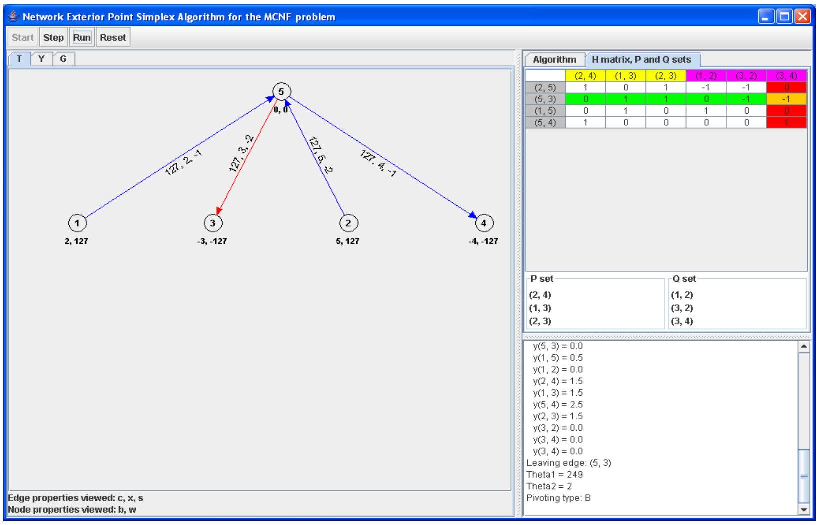


```
procedure NEPSA(G, T)
Step 0 (Initializations)
Compute x, w, and s
Find the sets P and Q
Compute the vectors h(i,j) and d = [ d(T), d(P), d(Q) ]
Step 1 (Test of optimality)
while P not ψ do
   if d(T) >= 0 then
      STOP. The problem is unbounded.
   else
      Step 2 (Choice of leaving arc)
      Compute a = min { x(i, j) / -d(T)(i, j) : d(T)(i, j) < 0 }
      choose the leaving arc (k, l) and compute y = x + ad
      Step 3 (Choice of entering arc)
      Compute theta1 = min { -s(i, j) : h(k, l) = 1, (i, j) ∈ P }
         theta2 = min { s(i, j) : h(k, l) = -1, (i, j) ∈ Q } and choose th
      Step 4 (Pivoting)
      if theta1 < theta2
         Set the entering arc (g, h) = (p1, p2)
         Set P = P - (g, h)
      else
```

```
  d(1, 2) = 0
  d(2, 4) = 1
  d(1, 3) = 1
  d(5, 4) = -1
  d(2, 3) = 1
  d(3, 2) = 0
  d(3, 4) = 0
- - - - - - - - - - - - - - - -
Iteration no: 1
d(T):
  d(5, 3) = -2
  d(2, 5) = -2
  d(1, 5) = -1
  d(5, 4) = -1
```

Figure 4. The candidate leaving arcs

The minimum ratio test is $a^{(1)} = 1{,}5$ and arc *(5,3)* leaves $T^{(1)}$. The leaving arc is highlighted below in Figure 5 using red colored line. Furthermore, the user may see the classical network simplex tableau in the tab entitled "H Matrix, P and Q sets", at the upper sub – frame, as it is also depicted in Figure 5.



	(2, 4)	(1, 3)	(2, 3)	(1, 2)	(3, 2)	(3, 4)
(2, 5)	1	0	1	-1	-1	0
(5, 3)	0	1	1	0	-1	-1
(1, 5)	0	1	0	1	0	0
(5, 4)	1	0	0	0	0	1

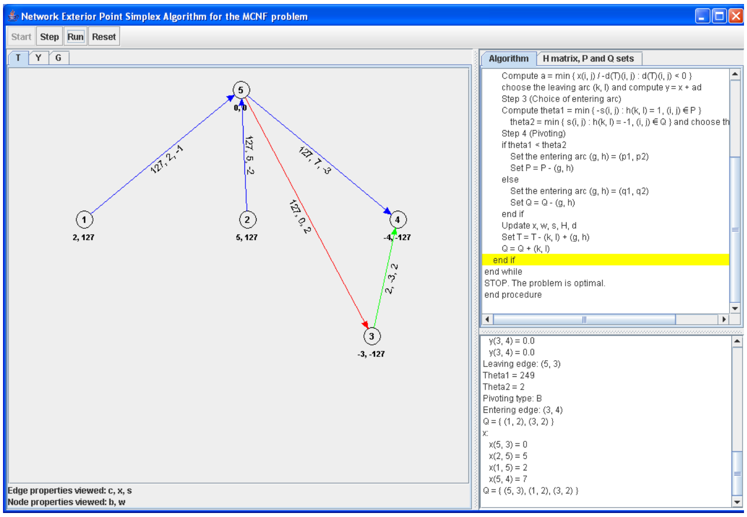
P set	Q set
(2, 4)	(1, 2)
(1, 3)	(3, 2)
(2, 3)	(3, 4)

```
  y(5, 3) = 0.0
  y(1, 5) = 0.5
  y(1, 2) = 0.0
  y(2, 4) = 1.5
  y(1, 3) = 1.5
  y(5, 4) = 2.5
  y(2, 3) = 1.5
  y(3, 2) = 0.0
  y(3, 4) = 0.0
  y(3, 4) = 0.0
Leaving edge: (5, 3)
Theta1 = 249
Theta2 = 2
Pivoting type: B
```

Edge properties viewed: c, x, s
Node properties viewed: b, w

Figure 5. Selection of the leaving arc

At *Step 2*, the ratios $\theta_1^{(1)}$ and $\theta_2^{(1)}$ are 249 and 2 respectively, so it holds that $\theta_1^{(1)} > \theta_2^{(1)}$ and arc *(3,4)* will enter the tree $T^{(1)}$. This is a *type B iteration*. At the 2nd iteration, the basic flow vector shall not be feasible. This fact in terms of Linear Programming is equivalent to following an exterior path; outside the feasible region. The entering arc is highlighted using green color as shown in Figure 6.



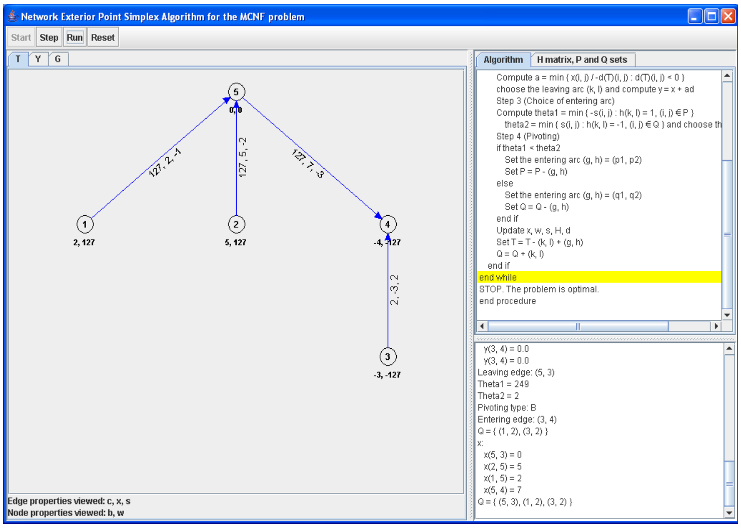
```
      Compute a = min { x(i, j) / -d(T)(i, j) : d(T)(i, j) < 0 }
      choose the leaving arc (k, l) and compute y = x + ad
      Step 3 (Choice of entering arc)
      Compute theta1 = min { -s(i, j) : h(k, l) = 1, (i, j) ∈ P }
        theta2 = min { s(i, j) : h(k, l) = -1, (i, j) ∈ Q } and choose th
      Step 4 (Pivoting)
      if theta1 < theta2
        Set the entering arc (g, h) = (p1, p2)
        Set P = P - (g, h)
      else
        Set the entering arc (g, h) = (q1, q2)
        Set Q = Q - (g, h)
      end if
      Update x, w, s, H, d
      Set T = T - (k, l) + (g, h)
      Q = Q + (k, l)
   end if
end while
STOP. The problem is optimal.
end procedure
```

```
  y(3, 4) = 0.0
  y(3, 4) = 0.0
Leaving edge: (5, 3)
Theta1 = 249
Theta2 = 2
Pivoting type: B
Entering edge: (3, 4)
Q = { (1, 2), (3, 2) }
x:
  x(5, 3) = 0
  x(2, 5) = 5
  x(1, 5) = 2
  x(5, 4) = 7
Q = { (5, 3), (1, 2), (3, 2) }
```

Edge properties viewed: c, x, s
Node properties viewed: b, w

Figure 6. Selection of the entering arc

At Step 4, the new basis tree $T^{(2)}$ is animatedly balanced and presented in Figure 7.

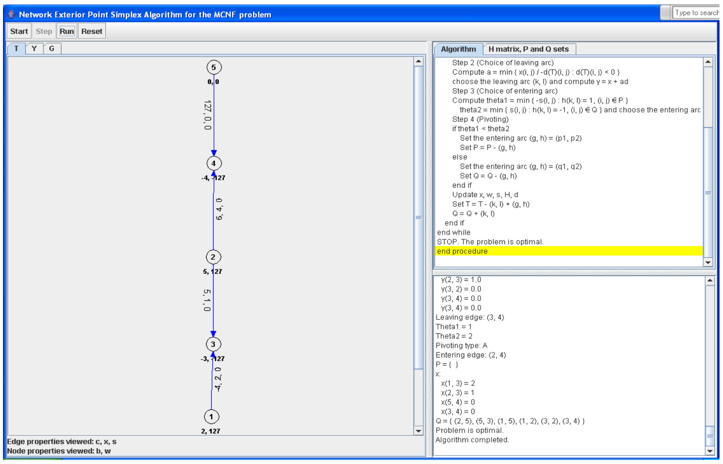


```
        Compute a = min { x(i, j) / -d(T)(i, j) : d(T)(i, j) < 0 }
        choose the leaving arc (k, l) and compute y = x + ad
        Step 3 (Choice of entering arc)
        Compute theta1 = min { -s(i, j) : h(k, l) = 1, (i, j) ∈ P }
          theta2 = min { s(i, j) : h(k, l) = -1, (i, j) ∈ Q } and choose th
        Step 4 (Pivoting)
        if theta1 < theta2
          Set the entering arc (g, h) = (p1, p2)
          Set P = P - (g, h)
        else
          Set the entering arc (g, h) = (q1, q2)
          Set Q = Q - (g, h)
        end if
        Update x, w, s, H, d
        Set T = T - (k, l) + (g, h)
        Q = Q + (k, l)
    end if
end while
STOP. The problem is optimal.
end procedure
```

```
  y(3, 4) = 0.0
  y(3, 4) = 0.0
Leaving edge: (5, 3)
Theta1 = 249
Theta2 = 2
Pivoting type: B
Entering edge: (3, 4)
Q = { (1, 2), (3, 2) }
x:
  x(5, 3) = 0
  x(2, 5) = 5
  x(1, 5) = 2
  x(5, 4) = 7
Q = { (5, 3), (1, 2), (3, 2) }
```

Figure 7. The updated tree solution

The example will not be further solved step by step, but it will be assumed that the user wants to run continuously the NEPSA till the termination of the algorithm. After the user presses the "Run" button, then all the iterations animate one after the other with a short delay. At this point the final tree solution is visualized as in the following Figure 8. Finally, it should be also mentioned, that the user has the possibility to take a look at computations of the previous iterations, using the vertical scroll bar at the lower sub – frame.



```
        Step 2 (Choice of leaving arc)
        Compute a = min { x(i, j) / -d(T)(i, j) : d(T)(i, j) < 0 }
        choose the leaving arc (k, l) and compute y = x + ad
        Step 3 (Choice of entering arc)
        Compute theta1 = min { -s(i, j) : h(k, l) = 1, (i, j) ∈ P }
          theta2 = min { s(i, j) : h(k, l) = -1, (i, j) ∈ Q } and choose the entering arc
        Step 4 (Pivoting)
        if theta1 < theta2
          Set the entering arc (g, h) = (p1, p2)
          Set P = P - (g, h)
        else
          Set the entering arc (g, h) = (q1, q2)
          Set Q = Q - (g, h)
        end if
        Update x, w, s, H, d
        Set T = T - (k, l) + (g, h)
        Q = Q + (k, l)
    end if
end while
STOP. The problem is optimal.
end procedure
```

```
  y(2, 3) = 1.0
  y(3, 2) = 0.0
  y(3, 4) = 0.0
  y(3, 4) = 0.0
Leaving edge: (3, 4)
Theta1 = 1
Theta2 = 2
Pivoting type: A
Entering edge: (2, 4)
P = {  }
x:
  x(1, 3) = 2
  x(2, 3) = 1
  x(5, 4) = 0
  x(3, 4) = 0
Q = { (2, 5), (5, 3), (1, 5), (1, 2), (3, 2), (3, 4) }
Problem is optimal.
Algorithm completed.
```

Figure 8. Algorithm completion

6. Conclusion - Future Work

In this paper we present an application which constitutes an 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 the Network Exterior Point Simplex Algorithm. The design of this tool is completely customized and can be utilized in various scientific areas. Furthermore, we plan to implement educational animation software also for other new algorithms, like the Dual Network Exterior Point Simplex Algorithm (DNEPSA), which is described in [Geranis et. al. (2008)].

To continue with, such applications can prove to be very helpful to all the students, 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 application from any place. It doesn't matter what his/her system is. It only requires that the Java Virtual Machine (JVM), is previously installed into his/her system. However, 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 a graph algorithm and a supplementary tool to verify the understanding of the algorithm. Students, who use algorithm animation software and also study algorithms using text books, are expected to understand better and easier how the algorithm works, or any other difficult intermediate step / computation. But, we should not forget that it is easier to find books explaining an algorithm than to have a free, easy to use, web based supplementary tool.

In order to estimate the educational value of this tool, on - line questionnaires could be used. Following this direction, the proposed didactic tool is schedule to be thoroughly evaluated inside the classrooms during the lectures. At the end of the laboratory course "Network Optimization", students of our department will fill it, in order to get feedback. This is very important, to improve and evaluate the teaching effectiveness of our software in real instruction environment.

To summarize with, it is more interesting to find out how a user - student makes use of this proposed educational application, rather than just explaining to him/her what might be the potential benefits, see [Lawrence et. al. (1994)]. Moreover, the proper use of algorithm animations in learning environments is already shown by other researchers; see [Kehoe et. al. (2001)]. Finally, 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 [Byrne et. al. (1999)].

References

Ahuja, R. K., Magnanti, T. L., Orlin, J. B. and Reddy, M. R. (1995), "Applications of Network Optimization", in: M.O. Ball, T.L. Magnanti, C.L. Monma and G.L.

Nemhauser (Eds), *Handbooks of Operations Research and Management Science, Network Models*, Elsevier Publications, 7, 1-83.

Almstrum, V.L., Hazzan, O., Guzdial, M. and Petre, M. (2005), "Challenges to computer science education research", *in Proc. of the 36th SIGCSE Technical Symposium on Computer Science Education*, St. Louis, Missouri, USA.

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*, Science Publications, New York, USA, 1(1), 19-23.

Bazaraa, M.S., Jarvis, J.J. and Sherali, H.D. (2005), *Linear Programming and Network Flows*, 3rd ed., Wiley Publications.

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.

Cordova, J.L. (2003), "The use of web-based visualization techniques and its effect on student comprehension", *Journal of Computing Sciences in Colleges*, 18(6), 67-71.

Dosios, K., Paparrizos, K., Samaras, N. and Sifaleras, A. (2003), "NetPro, an Educational Platform for Network Optimization", *in Proc. of the 16th National Conference of Hellenic Operational Research Society*, Larisa, Greece, 1, 287-295.

Geranis, G., Paparrizos, K. and Sifaleras, A. (2008), "A dual exterior point simplex type algorithm for the minimum cost network flow problem", accepted for publication in the *Yugoslav Journal of Operations Research*.

Glover, F., Klingman, D. and Phillips, N. (1992), *Network Models in Optimization and Their Applications in Practice*, 1st ed., Wiley Publications.

Hendrix, T.D., Cross, J.H. and Barowski, L.A. (2004), "An extensible framework for providing dynamic data structure visualizations in a lightweight IDE", *in Proc. of the 35th SIGCSE Technical Symposium on Computer Science Education*, ACM Press, Norfolk, Virginia, USA.

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.

Karagiannis, P., Markelis, I., Paparrizos, K., Samaras, N. and Sifaleras, A. (2006), "E - learning technologies: employing matlab web server to facilitate the education of mathematical programming", *International Journal of Mathematical Education in Science and Technology*, Taylor & Francis Publications, 37(7), 765-782.

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.

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.

Lazaridis, V., Paparrizos, K., Samaras, N. and Sifaleras, A. (2007), "Visual LinProg: A Web-based Educational Software for Linear Programming", *Computer Applications in Engineering Education*, Wiley Periodicals Inc., 15(1), 1-14.

Lazaridis, V., Samaras, N., Zissopoulos, D. (2003), "Visualization and teaching simplex algorithm, *in Proc. of the 3rd IEEE International Conference on Advanced Learning Technologies*, 9-11 July, Athens, Greece, 270-271.

Luengo, V. (2005), "Some didactical and Epistemological Considerations in the Design of Educational Software: The Cabri-euclide Example", *International Journal of Computers for Mathematical Learning*, 10(1), 1-29.

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

Naps, T. (1997), "Algorithm visualization on the World Wide Web – the difference Java makes!, *in Proc. of the 2nd Conference on Integrating technology Into Computer Science Education*, Uppsala, Sweden.

Papamanthou, C., Paparrizos, K., and Samaras, N. (2005), "A Parametric Visualization Software for the Assignment Problem", *Yugoslav Journal of Operations Research,* 15(1), 1-12.

Paparrizos, K., Samaras, N. and Sifaleras, A. (2004), "A learning tool for the visualization of general, directed or undirected rooted trees", in: K. Morgan & J.M. Spector (eds), *WIT Transactions on Information and Communication Technologies*, 30, WIT Press, 205-213, *presented at the 1st International Conference on New Learning Paradigms and New Learning Tools*, 10-12 May, Skiathos, Greece.

Paparrizos, K., Samaras, N. and Sifaleras, A. (2008), "An exterior Simplex type algorithm for the minimum cost network flow problem", *accepted for publication in Computers & Operations Research,* Elsevier Ltd.

Saraiya, P., Shaffer, C.A., McCrickard, D.S. and North, C. (2004), "Effective features of algorithm visualizations", *in Proc. of the 35th SIGCSE Technical Symposium on Computer Science Education*, ACM Press, Norfolk, Virginia, USA.

Sinclair, N., Zazkis, R. and Liljedahl, P. (2004), "Number Worlds: Visual and Experimental Access to Elementary Number Theory Concepts", *International Journal of Computers for Mathematical Learning*, 8(3), 235-263.

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.

Varkas, G., Kostoglou, V. and Paparrizos, K. (2005), "Broadband Networks and Services: Innovative Means of Human Communication", *in Proc. of the 4th International Conference on New Horizons in Industry and Education,* Corfu, Greece, 52-57.