The 7th Balkan Conference on Operational Research
"BACOR 05"
Constanta, May 2005, Romania

A NEW SIMPLEX TYPE ALGORITHM FOR THE MINIMUM COST NETWORK FLOW PROBLEM

KARAGIANNIS PANAGIOTIS
PAPARRIZOS KONSTANTINOS
SAMARAS NIKOLAOS
SIFALERAS ANGELO *

Department of Applied Informatics, University of Macedonia, Greece, Thessaloniki

---

Abstract
A new primal simplex type algorithm for the Minimum Cost Network Flow Problem (MCNFP) is presented. The proposed algorithm belongs to a special "exterior simplex type" category. Contrary to the network primal simplex algorithm, the new algorithm computes two flows. One flow is basic but not always feasible and the other is feasible but not basic. The pseudo code and analytical description of the pivot procedure is given. Furthermore, the necessary steps are analytically shown through an illustrative example.

Keywords*: Minimum Cost Network Flow Problem, Network Optimization, Operation Research.*

1. INTRODUCTION

Network Optimization is a large part of Combinatorial Optimization. The Minimum Cost Network Flow Problem, (henceforth the abbreviation MCNFP will be used) constitutes a wide category of problems; perhaps the most important of the research area of Network Optimization. There exists a plethora of every-day applications, in

---

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

Informatics, Constructions, Telecommunications sector etc, which can be mathematically formulated using the MCNFP. Many of them can be found in [1] and [2]. Furthermore, other well known problems like for example the shortest path problem, the assignment problem and others, are special cases of the MCNFP.

The proposed new algorithm belongs to a special "*exterior simplex type*" category and it is of the same type as the one described in [3]. Instead of the long expression "*Network Exterior Point Simplex Algorithm for the MCNFP*", the discussed algorithm will be referred with the acronym "*NEPSA*". Contrary to the Network Primal Simplex algorithm, NEPSA computes two flows. One flow is basic but not always feasible and the other is feasible but not always basic. Previous computational results on the general linear problem have shown, through a computational study, that this type of algorithms is more efficient than the classical Revised Primal Simplex algorithm in CPU time. This computational improvement is probably due to the essential reduction on the number of iterations. It is believed, that these encouraging results could be also applied to the MCNFP.

To this date, there are no specializations of exterior type simplex algorithms for the minimum cost flow problem. There have been implementations of exterior type simplex algorithms exists only for the assignment problem, see [4]. In this paper, we present the first exterior type network simplex algorithm for the minimum cost flow problem. The proofs of lemmas and theorems were omitted, for the sole reason that this paper doesn't grow too much in size.

In Section 2, some notations and definitions are given. In Section 3 we describe NEPSA in a general form. A demonstration of NEPSA is given in Section 4 with an illustrative example. Finally, in Section 5, conclusions are made and we also discuss some possible future work.

2. NOTATIONS AND DEFINITIONS

Let $G = (N, A)$ be a directed network with n nodes and m arcs. 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 in the un-capacitated MCNFP. This way each arc has a lower bound on the flow capacity $l_{ij} = 0$, and upper bound $u_{ij} = +\infty$. Associated with each arc (i,j) is also an amount x_{ij} of flow on the arc (contrary to the network primal 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 transshipment node, depending upon whether $(b_i \otimes 0)$, where $\otimes$ denotes the type of constraint >, < or = respectively. We make the assumption that $\sum_{i=1}^{n+1} b(i) = 0,$ (total supply equals total demand) and G will be called as a balanced network. Now, the minimum cost flow problem can be stated as follows:

$$\begin{aligned} &\min \sum_{\{(i,j)\in A\}} c_{ij} x_{ij} \\ &\text{s.t} \sum_{\{k:(i,k)\in A\}} x_{ik} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b_i, \quad i \in N \\ &\quad x_{ij} \geq 0, \quad (i,j) \in A \end{aligned} \tag{2.1}$$

A vector of dual variables for the network G is a vector *w* with *n* components. For each vector w, the reduced costs s_{ij} of each non basic arc, are defined as $s_{ij} = c_{ij} - w_i + w_j$. Network Simplex type algorithms compute basic solutions *x* and *(w,s)* which are complemental.

Furthermore, in this paper, notations such as paths, directed paths, cycles, directed cycles and trees, are defined as in [5]. The basic solution for the MCNFP can be represented using a tree. Each tree of a MCNFP consists of a flow vector *x*, which is the unique solution of the above equations constraints. The basic flow of the tree T, will be denoted as *x(T)*. We say that the flow of the tree is feasible, if $x_{ij} \geq 0, \ \forall \ \text{arc}(i, j) \in T$. Each tree, which has a nonnegative flow, will be called a feasible tree. Equivalent, each tree which has a negative flow, will be called a non feasible tree,

Contrary to the Network Primal Simplex algorithm, NEPSA first selects a leaving arc and afterwards selects 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 $C^{(k)}$ the cycle that would have been created at the *k-th* iteration, if prior to the leaving arc, the entering arc have joined the basic tree *T*. More generally, for any variable ψ, $\psi^{(k)}$ will denote the value of the ψ variable at the *k-th* iteration. If two arcs, for example *(i,j)* and *(g,h)* have the same orientation in $C^{(k)}$, then this fact will be symbolized as $(i, j) \uparrow\uparrow (g, h)$. If, on the other hand, arcs *(i,j)* and *(g,h)* have the opposite orientation in $C^{(k)}$, then this fact will be symbolized as $(i, j) \uparrow\downarrow (g, h)$.

At *Step 0*, an initial feasible tree is constructed using the well known big M method. The set of arcs is partitioned in two sets. The first set corresponds to the basis tree and will be denoted henceforth as *T*. The second set corresponding to the non basic arcs will be further partitioned in the following two sets *P* and Q:

$$P = \{(i, j)\}, (i, j) \notin T : s_{ij} < 0$$

$$Q = \{(i, j)\}, (i, j) \notin T : s_{ij} \geq 0$$

This way, the initial set of arcs A is finally partitioned as $A = [T \ P \ \text{Q}]$.

Vector h_{gh} denotes the representation of each non basic arc, for example *(g,h)*, in terms of the basic arcs *(i,j)*. Each non basic arc *(g,h)* corresponds to a column h_{gh}. Each element of this column corresponds to a basic arc. The elements of h_{gh} are calculated using the following rules. For detailed description of these calculations see [6].

$$h_{gh}(i, j) = -1, \text{ if } (i, j) \in C^k \text{ and } (i, j) \uparrow\uparrow (g, h)$$

$$h_{gh}(i, j) = \ \ 1, \text{ if } (i, j) \in C^k \text{ and } (i, j) \uparrow\downarrow (g, h)$$

$$h_{gh}(i, j) = \ \ 0, \text{ if } (i, j) \notin C^k$$

Vector *d* denotes a "complemental" flow which assists the algorithm to compute the second feasible but not always basic flow. Accordingly, vector *d* is partitioned as $d = \{d(T), d(P), d(Q)\}$. The elements of d are computed using the following relations.

$$d(T) = -\sum_{(i,j)\in P} h_{ij}, d(P) = (\lambda_1, \lambda_2, ..., \lambda_{|P|}), d(Q) = (0, 0, ..., 0) \qquad (2.2)$$

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, vector λ is initialized using $\lambda = (1,1,...,1)$.

If $P = \varnothing$, then the problem is optimal and the algorithm terminates. At *Step 1* the optimality condition, whether it holds $P = \varnothing$ or not, is examined. If the previous relation holds, then the algorithm stops; an optimal solution has been found. Otherwise, the algorithm continues. In this case, if $\exists (i, j) \in T : d_{ij} < 0$, then the algorithm proceeds to the next step, otherwise the problem is unbounded and therefore negative cost cycles have been identified. At *Step 2*, the selection of the leaving arc is made using the following minimum ratio test:

$$a = \frac{x_{kl}}{-d_{kl}} = \min\left\{\frac{x_{ij}}{-d_{ij}} : (i, j) \in T, d_{ij} < 0\right\} \tag{2.3}$$

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 2*, the selection of entering arc is being done using the following minimum ratio tests:

$$\theta_1 = -s_{p_1 p_2} = \min\left\{-s_{ij} : h_{ij}(k,l) = 1, (i, j) \in P\right\} \tag{2.4}$$

$$\theta_2 = s_{q_1 q_2} \min\left\{s_{ij} : h_{ij}(k,l) = -1, (i, j) \in Q\right\} \tag{2.5}$$

If $\theta_1 \le \theta_2$, then the pivot will be called "*Type A iteration*". In this case the entering arc will be *(g,h) = (p_1,p_2)* and $P = P \sim (p_1, p_2)$; otherwise, the pivot will be called "*Type B iteration*". In the latter case the entering arc will be *(g,h) = (q_1,q_2)* and $Q = Q \sim (q_1, q_2)$.Using the new partition *(T P Q)*, where $T = T \sim (k,l) \cup (g,h)$ the vectors x_{ij}, s_{ij}, h_{ij} and d_{ij} are updated.

There are two choices for the implementation of the *d* vector. The first choice is to compute the feasible flow *y* at each iteration and to update *d* using $d^{(k+1)} = y^{(k)} - x^{(k+1)}$, $\forall$ *k-th* iteration; if $(g,h) \in P$, $d_{gh}^{(k+1)} = d_{gh}^{(k+1)} + 1$. The implementation based on the second choice, doesn't calculate the feasible flow *y* at each iteration. Contrary to the previous update method, only the *d(T)* subset is computed. At each iteration, the *d(T)* is updated using the same method as the columns h_{ij} (from the network simplex tableau). It can be proved that the elements of the *d(T)* vector using the first method, are equal to the elements of the *d(T)* vector, using the latter method, multiplied by a constant number $\varphi = a^{(1)} * a^{(2)} * .... * a^{(i)}$. This difference in the values of $d_{ij}(T)$ between the two methods doesn't affect the comparison of the ratios. Therefore, it is much more preferable to implement the NEPSA algorithm in H/Y using the latter method, since it demands less data to be kept and accessed at each iteration. This method also will be used in the example of Section 3.

3. ALGORITHM DESCRIPTION

The presentation of NEPSA in this paper relies on the two following assumptions.

Assumption 1, (Non - degeneracy assumption).

It is assumed that every basic feasible solution is non-degenerate; that is firstly $x_{ij} \neq 0$ $\forall$ *(i,j)* $\in T$ and secondly $s_{ij} \neq 0$, $\forall$ *(i,j)* $\notin T$.

Assumption 2, (Optimality assumption)

Moreover, we will assume that there exists an optimal solution for all the problems, which NEPSA algorithm is applied to.

The formal description of NEPSA is as follows:

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 (2.2)
6  Step 1 (Test of Optimality)
7  do while ( P ≠ ∅ )
8      if ( d(T) ≥ 0 ) then
9          EXIT (The problem is unbounded)
10     else
11         Step 2 (Selection of leaving arc)
12         Compute a, using relation (2.3)
13         Choose the leaving arc (k,l)
14         Compute feasible flow y
15         Step 3 (Selection of entering arc)
16         Compute θ1, θ2, using relations (2.4) and (2.5)
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)
```

Table 2.1. Algorithm's pseudo – code

The correctness of the proposed algorithm relies on the following theorems. However, the proofs of correctness of these theorems are omitted in this paper.

Theorem 1 If the problem is not degenerate, then the value of the objective function is decreased, from iteration to iteration.

Theorem 2 If the problem is not degenerate, then the algorithm will perform a finite number of iterations.

Lemma 1 If it holds that *d(T)* = 0, then the arcs belonging to the P set form a directed cycle.

Theorem 3 If it holds that $d(T) \geq 0$ and $P \neq \varnothing$, then the problem is unbounded.

Theorem 4 At each iteration (either pivot *type A* or *B*), it holds that $\forall (i, j) \in P : s_{ij} < 0$ and $\forall (i, j) \in Q : s_{ij} \geq 0$.

Theorem 5 If $\exists (i, j) : x_{ij} < 0 \Rightarrow \exists d_{ij} > 0 : \beta = \max\left\{\frac{x_{ij}}{-d_{ij}}\right\} < a$.

Theorem 6 The basic flow *y* remains always feasible.

Theorem 7 If $P = \varnothing$, then the basis tree *T* is optimal.

4. AN ILLUSTRATIVE EXAMPLE

In this section a demonstration of the two different pivot types will be presented, using two iterations of the following minimum cost network flow problem. The network which represents the discussed MCNFP is shown in figure 2.1a. At *step 0*, the big M method is applied to the initial network. After the insertion of the artificial node and the corresponding arcs, the modified network is shown in Figure 2.1b.

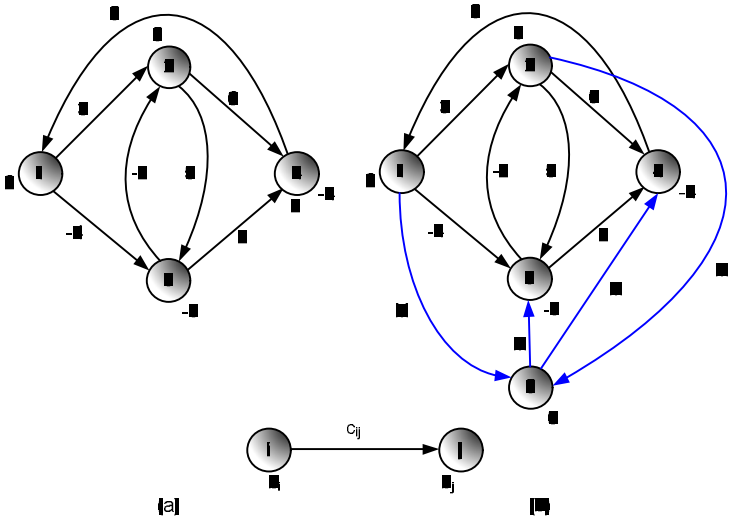

Figure 2.1: The initial network (a) and the augmented network (b)

At *step 0*, NEPSA algorithm starts with the initial feasible tree, which is depicted in blue color in Figure 2.1b. 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:

$$s^{(1)} = (s_{13}^{(1)}, s_{23}^{(1)}, s_{24}^{(1)}, s_{12}^{(1)}, s_{32}^{(1)}, s_{34}^{(1)} s_{41}^{(1)}) \Rightarrow s^{(1)} = (-258,-249,-248,3,252,2,259).$$

Therefore, $P^{(1)} = \{(1,3),(2,3),(2,4)\}$ and $Q^{(1)} = \{(1,2),(3,2),(3,4),(4,1)\}$. The total cost corresponding to the basic flow vector $x^{(1)}$, is $z^{(1)} = \sum_{(i,j)\in T^{(1)}} c_{ij} x_{ij}^{(1)} \Rightarrow z^{(1)} = 1778$.

1st Iteration

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*. The minimum ratio test is $a^{(1)} = 1{,}5$ and arc *(5,3)* exits $T^{(1)}$. The leaving arc is highlighted in Figure 2.2a using red intersecting lines.

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 $T^{(1)}$. Accordingly, 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 in Figure 2.2a using green color. At *step 4*, the new basis tree $T^{(2)}$ is presented in Figure 2.2b.

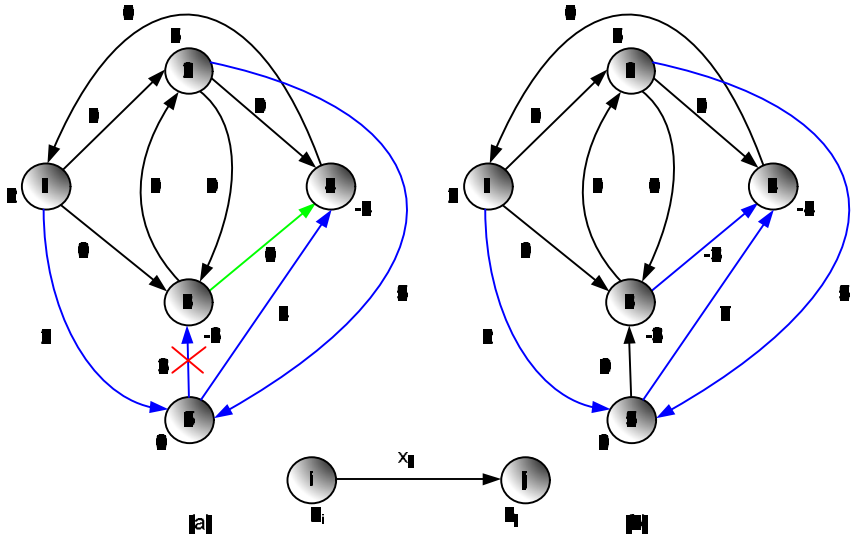

Figure 2.2: The leaving & entering arc (a) and the updated basis tree $t^{(2)}$ (b).

The updated basic flow vector is now:

$x^{(2)} = (x_{15}^{(2)}, x_{25}^{(2)}, x_{34}^{(2)}, x_{54}^{(2)}) \Rightarrow x^{(2)} = (2, 5, -3, 7)$

and the updated reduced costs vector is:

$s^{(2)} = (s_{13}^{(2)}, s_{23}^{(2)}, s_{24}^{(2)}, s_{12}^{(2)}, s_{32}^{(2)}, s_{41}^{(2)}) \Rightarrow s^{(2)} = (-256, -247, -248, 3, 250, 259)$.

Therefore, $P^{(2)} = \{(1,3), (2,3), (2,4)\}$ and $Q^{(2)} = \{(1,2), (3,2), (4,1)\}$, (arc *(5,3)* was artificial, so it is discarded from any future computations). The total cost corresponding to the basic flow vector $x^{(2)}$, is $z^{(2)} = \sum_{(i,j)\in T} c_{ij} x_{ij}^{(2)} \Rightarrow z^{(2)} = 1772$.

2nd Iteration

At *step 1*, it holds that $P^{(2)} \neq \varnothing$, therefore the algorithm doesn't terminate. Since, $d^{(2)}(T) = \{d_{15}^{(2)}, d_{25}^{(2)}, d_{34}^{(2)}, d_{54}^{(2)}\} \Rightarrow d^{(2)} = \{-1, -2, 2, -3\}$, NEPSA proceeds to *step 2*. The minimum ratio test is $a^{(2)}$ = 2 and arc *(1,5)* exits $T^{(2)}$. The leaving arc is highlighted in Figure 2.3a using red intersecting lines.

At *step 2*, the ratios $\theta_1^{(2)}$ and $\theta_2^{(2)}$ are 256 and 259 respectively, so it holds that $\theta_1^{(2)} < \theta_2^{(2)}$ and arc *(1,3)* will join the basis tree $T^{(2)}$. Accordingly, this is a *type A iteration*. The entering arc is highlighted in Figure 2.3a using green color. At *step 4*, the new basis tree $T^{(3)}$ is presented in Figure 2.3b.

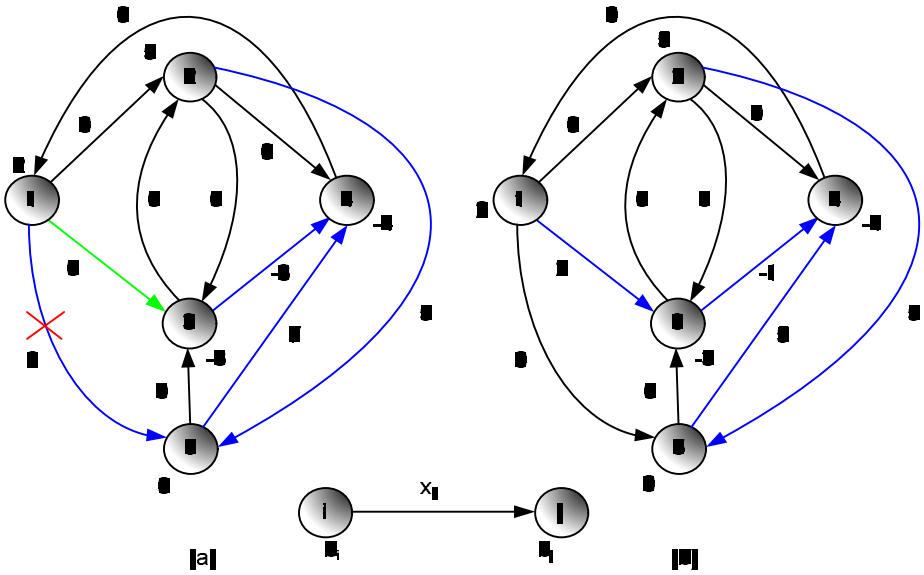

Figure 2.3: The leaving & entering arc (a) and the updated basis tree $t^{(3)}$ (b)

This was a graphical depiction of the two characteristic pivot types of NEPSA. The example will be stopped here and not be further solved, because the rest computations are easy to be carried out.

One of the main features of the proposed new algorithm, which diversifies it from the classical Network Primal Simplex is the selection of the leaving and entering arcs. The latter algorithm would always choose as entering and leaving arcs, two arcs which have the opposite orientation inside the $C^{(k)}$. On the contrary, NEPSA algorithm in *type B* pivots, choose as leaving and entering arcs, a couple of arcs having the same orientation inside the $C^{(1)}$, (see Figure 2.2a at the 1st iteration).

5. CONCLUSIONS AND FUTURE WORK

There have been developed many efficient data structures. Some of them are the dynamic trees as described in [7], Fibonacci heaps as described in [8] or other class of data structures using depth, parent, and preorder lists as described in [9]. Therefore, it is much interesting to use such data structures, (in Fortran Programming Language) in the NEPSA algorithm.

If certain improvements will be made, regarding the data structures used and better programming techniques, then it is possible to test NEPSA algorithm against some of the state-of-the-art implementations, like for example RELAX-IV [10], NETFLO [11] or RNET [12]. However, it would require first to develop appropriate anti-cycling rules, see [13] and examine the behavior of NEPSA to certain well known pathological instances (at least for the classical Network Primal Simplex algorithm), see [14] and [15]. Moreover, NEPSA algorithm will be incorporated in the Network Optimization suite *NetPro*, see [16]. This way, *NetPro* will expand its capabilities with new algorithms for the MCNFP.

BIBLIOGRAPHY

[1] Ahuja, R. K., Magnanti, T. L., Orlin, J. B. and Reddy, M. R. (1995) "Applications of Network Optimization", in Handbooks of Operations Research and Management Science, Network Models, Editors Ball, M. O., Magnanti, T. L., Monma, C. L. and Nemhauser, G. L., 7, Elsevier Publications, pp. 1-83;

[2] Glover F., Klingman, D. and Phillips N. (1992) "Network Models in Optimization and Their Applications in Practice", 1st Ed., Wiley Publications;

[3] Paparrizos, K., Samaras, N. and Stephanides, G. (2003) "An efficient simplex type algorithm for sparse and dense linear programs", European Journal of Operational Research, 148, pp. 323-334;

[4] Paparrizos, K. (1991) "An infeasible (exterior point) simplex algorithm for assignment problems", Mathematical Programming, 51, pp. 45-54;

[5] Paparrizos, K., Samaras, N. and Sifaleras, A. (2005) "Network Optimization", University of Macedonia Publications;

[6] Bazaraa, M. S., Jarvis, J. J. and Sherali H. D. (2005) "Linear Programming and Network Flows", Wiley Publications;

[7] Goldberg, A. V., Grigoriadis, M. D. and Tarjan, R. E. (1991) "Use of dynamic trees in a network simplex algorithm for the maximum flow problem", Mathematical Programming, 50, pp. 277-290;

[8] Fredman, M. C. and Tarjan, R. E. (1987) “Fibonacci heaps and their uses in improved network optimization algorithms”, Journal of the ACM, 34(3), pp. 596-615;
[9] Ali, A. I., Helgason, R. V., Kennington, J. L. and Lall H. S. (1978) “Primal simplex network codes: state-of-the-art implementation technology”, Networks, 8, pp. 315-339;
[10] Bertsekas, D.P. and Tseng, P.(1994) “RELAX-IV: a faster version of the RELAX code for solving minimum cost flow problems”, Technical report, Massachusetts Institute of Technology, Laboratory for Information and Decision Systems;
[11] Kennington, J. L. and Helgason, R. V. (1980) Algorithms for Network Programming, Wiley Publications.
[12] Grigoriadis, M. (1984) “An Efficient Implementation of the Network Simplex Method”, Mathematical Programming Study, 26, pp. 83-111;
[13] Cunningham, W. H. (1976) “A network simplex method”, Mathematical Programming, 11, pp. 105-116;
[14] Zadeh, N. (1973) “More Pathological Examples for Network Flow Problems”, Mathematical Programming, 5, pp. 217-224;
[15] Zadeh, N. (1973) “A bad network problem for the simplex method and other minimum cost flow algorithms”, Mathematical Programming, 5, pp. 255-266;
[16] Dosios K., Paparrizos, K., Samaras, N. and Sifaleras, A. (2003) “NetPro, an Educational Platform for Network Optimization”, In Proc. of 16th National Conference of HELORS, Larissa, pp. 287-295.