Yugoslav Journal of Operations Research
Vol 19 (2009), Number 1, 157-170
DOI:10.2298/YUJOR0901157G

A DUAL EXTERIOR POINT SIMPLEX TYPE ALGORITHM FOR THE MINIMUM COST NETWORK FLOW PROBLEM

George GERANIS
Konstantinos PAPARIZZOS
Angelo SIFALERAS
Department of Applied Informatics, University of Macedonia,
geranis,paparriz,sifalera@uom.gr

Received: December 2007 / Accepted: May 2009

Abstract: A new dual simplex type algorithm for the Minimum Cost Network Flow Problem (MCNFP) is presented. The proposed algorithm belongs to a special "exterior-point simplex type" category. Similarly to the classical network dual simplex algorithm (NDSA), this algorithm starts with a dual feasible tree-solution and reduces the primal infeasibility, iteration by iteration. However, contrary to the NDSA, the new algorithm does not always maintain a dual feasible solution. Instead, the new algorithm might reach a basic point (tree-solution) outside the dual feasible area (exterior point - dual infeasible tree).

Keywords: Operations research, combinatorial optimization, minimum cost network flow problem.

1. INTRODUCTION

The Minimum Cost Network Flow Problem (MCNFP) is the problem of finding a minimum cost flow of product units, through a number of source nodes, sinks and transshipment nodes. Other common problems, such as the shortest path problem, the transportation problem, the transshipment problem, the assignment problem etc., are the special cases of the MCNFP. Such problems appear very frequently in different technology sectors, like the Information Technology, the Telecommunications, the Transportation, the Resource Management, etc. Algorithms developed for the MCNFP can offer good solutions for such problems. A number of different problems that can be solved by the following MCNFP methods are described in [1] and [9].

The MCNFP can be easily transformed into a Linear Programming Problem and well known general linear programming techniques could be applied in order to find an optimal solution. Such techniques do not take advantage of some special features met in the MCNFP. So, other special Simplex-type algorithms have been developed, such as the Primal Network Simplex Method and the Dual Network Simplex Method. There are also other non Simplex-type algorithms that can be used for the solving of the same problem, as presented in [7], [2] and [15]. An exterior-point primal simplex-type algorithm for solving the MCNFP has also been presented in [13].

This paper comes to present for the first time an exterior point dual simplex-type algorithm for the MCNFP. The algorithm is named Dual Network Exterior Point Simplex Algorithm (DNEPSA) for the MCNFP. It starts with a dual feasible tree-solution and, iteration by iteration, it produces new tree-solutions closer to an optimal solution, reducing the problem's infeasibility. Contrary to the Network Dual Simplex Algorithm, the tree-solution at every iteration is not necessarily dual feasible but, after a number of iterations, the algorithm reaches a tree-solution that is both primal feasible and dual feasible and therefore it is optimal.

Section 2 gives the notation that will be used in this paper and a short description of the MCNFP. In Section 3, the Dual Network Exterior Point Simplex Algorithm (DNEPSA) is described and the steps that have to be followed are described in detail. An illustrative example presenting the algorithm step by step is given in Section 4. Finally, Section 5 gives some conclusions and plans for future work.

2. NOTATIONS AND PROBLEM STATEMENT

In this Section we shall give a short description of the MCNFP. Let G=(N,A) be a directed network that consists of a finite set of nodes N and a finite set of directed arcs A, that link together pairs of nodes. Let n and m be the number of nodes and arcs respectively. For each node $i \in N$, there is an associated variable b_i representing the available supply or demand at that node. A node i is a supply node (source), if it is $b_i > 0$. On the other hand, it is a demand node (sink), if it is $b_i < 0$. Finally, the node i is a transshipment node in case it is $b_i = 0$. The total supply has to be equal to the total demand, i.e. it has to be $\sum_{i \in N} b_i = 0$ (balanced network).

For every arc (i,j) we have an associated flow x_{ij} that shows the amount of product units transferred from node i to node j and an associated cost per unit value x_{ij}. Therefore, the total cost is equal to $\sum_{(i,j) \in A} c_{ij} x_{ij}$ and the MCNFP is the problem of finding a flow that minimizes that total cost.

We can have, for the flow x_{ij} on arc (i,j), a lower and an upper bound, l_{ij} and u_{ij} respectively. This gives an additional constraint $l_{ij} \le x_{ij} \le u_{ij}$ for every arc (i,j). In our case we consider it is $l_{ij} = 0$ and $u_{ij} = +\infty$. In other words, our algorithm is applied to the uncapacitated MCNFP. For every node i it has to be

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = b_i$$

because the outgoing flow must be equal to the incoming flow plus the node's supply. Therefore, the MCNFP can be formulated as follows:

$$\text{minimize} \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{2.1}$$

subject to

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = b_i \forall i \in N \tag{2.2}$$

$$x_{ij} \geq 0, \forall (ij) \in A \tag{2.3}$$

Since it is $\sum_{i\in N} b_i = 0$, by using formula (2.2), it comes out that

$$\sum_{i\in N} \left(\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} \right) = 0$$

That means that constraints (2.2) are linearly dependent and we could arbitrarily drop out one of them. Of course, a problem like this could be solved using standard well-known Linear Programming algorithms, but these algorithms do not take advantage of some special features met in the MCNFP.

There is a set of dual variables w_i, one for every node, and a number of reduced cost variables w_{ij}, one for every directed arc. These are the variables used for the formulation of the dual problem. Network simplex-type algorithms start from a basic tree-solution and compute vectors x, w, s consisting of variables x_{ij}, w and w_{ij} respectively. If for a tree-solution T, it is $x_{ij} \geq 0$ for every arc $(i,j) \in T$, then that solution is said to be primal feasible. If for a tree-solution T, it is $s_{ij} \geq 0$ for every arc $(i,j) \notin T$ then it is said to be dual feasible. A solution being both primal feasible and dual feasible represents an optimal solution. Primal network simplex-type algorithms start from a primal feasible solution, while dual network simplex-type algorithms, like the algorithm described here, start from a dual feasible solution.

3. ALGORITHM DESCRIPTION

The Dual Network Exterior Point Simplex Algorithm (DNEPSA for short), starts from a dual feasible basic tree-solution T and, after a number of iterations, it comes to a tree-solution that is both primal feasible and dual feasible and therefore, it is optimal. In contrary to the existing dual network simplex-type algorithms, the tree-solutions formed during the iterations are not necessarily always dual feasible but they can be both primal infeasible and dual infeasible. So, DNEPSA is a simplex-type algorithm starting from a dual solution that reaches an optimal solution by following a route consisting of solutions that do not belong to the primal feasible area.

There are different techniques that can be used in order to find a starting dual feasible tree-solution. An algorithm that can construct a dual feasible tree-solution for the generalized network problem (and also for pure networks) is described in [8] and an improved version of the algorithm is presented in [11], which gives a dual feasible solution that is closer to an optimal solution.

Given the starting dual feasible tree-solution, it is easy, starting from the leaf nodes, to compute the flows x_{ij} for all the basic arcs (i,j) in that tree (step 1). It is also very easy to compute values for the dual variables w_i from the equations:

$$w_i - w_i = c_{ij} \text{ , for every arc } (i,j) \in T \tag{3.1}$$

In equations (3.1) we have n-1 equations and n variables, so, we can choose one of the dual variables (e.g. w_i) and set it equal to an arbitrary value (e.g. 0). Then, it is easy to compute the values of the rest of the dual variables.

In order to calculate the reduced costs s_{ij} for the non-basic arcs (i,j), we can use the following equations:

$$s_{ij} = c_{ij} + w_i - w_j \text{ , for every arc } (i,j) \notin T \tag{3.2}$$

while it is $s_{ij} = 0$ for all the basic arcs.

Next, the algorithm creates a set, named I_, that contains the basic arcs (i,j) having negative flow, i.e. it is $x_{ij} < 0$. It also creates set I_+ containing the rest of the arcs. If it is I_=∅, this means that the tree-solution is feasible and therefore it is optimal (step 2).

After this, the algorithm considers the non-basic arcs $(i,j) \notin T$. When such an arc (i,j) is added to the basic tree-solution T, then a cycle C is created. That cycle may contain arcs of I_ having the same orientation as (i,j) and others having the opposite orientation. For every non-basic arc, let d_{ij} be the difference between the number of arcs in I_ having the same orientation in cycle as (i,j) minus the number of them having the opposite orientation. J_ consists of those non-basic arcs (i,j) having $d_{ij} > 0$ (step 3).

After creating set J_, we have to choose amongst its arcs, the one that will be the entering arc (g,h). This is the arc of J_ that gives the minimum ratio $-s_{ij} / d_{ij}$ (step 4).

Next, the algorithm has to find the leaving arc (k,l). This is done by checking the cycle C formed after adding the entering arc (g,h) to the tree T. For the arcs of I_ having the same orientation in C as (g,h), we choose arc (k_1,l_1) that corresponds to the minimum absolute flow. Similarly, for the arcs of I_+ having orientation opposite to the entering arc (g,h), we choose arc (k_2,l_2) that corresponds to the minimum absolute flow. If we denote θ_1 and θ_2 these two minimum values, we decide the leaving arc by comparing θ_1 against θ_2. If it is $\theta_1 \le \theta_2$, then the leaving arc is (k,l) = (k_1,l_1), otherwise it is (k,l) = (k_2,l_2).

At this point, the algorithm has come to a new basic tree-solution T. The same process has to be repeated until the algorithm reaches to an optimal solution.

The algorithm's steps are described below. The symbols $\uparrow\uparrow$ and $\uparrow\downarrow$ used here, denote arcs that have the same orientation or opposite orientation, respectively.

Algorithm DNEPSA

1. Start with a dual feasible tree T. Compute the flow variables x_{ij} for all the basic arcs, the dual variables w_i and the reduced cost variables s_{ij}, by applying formulas (3.1) and (3.2).
2. Set $I\_ = \{(i,j) \in T : x_{ij} < 0\}$ and $I_+ = \{(i,j) \in T : x_{ij} \geq 0\}$ If it is $I\_ = \varnothing$, then the current tree-solution T is an optimal solution.
3. Set
   $J\_ = \{(i,j) \notin T : if\ \ added\ \ to\ \ T, in\ \ the\ \ cycle\ \ created, it\ is\ d_{ij} = p_{ij} - n_{ij} > 0\}$
   where p_{ij} is the number of arcs in $I\_$ having the same orientation as (i,j) and n_{ij} is the number of arcs in $I\_$ having the opposite orientation.
4. Choose arc $(g,h) \in J\_$, where it is

$$-\frac{s_{gh}}{d_{gh}} = \min\{-\frac{s_{ij}}{d_{ij}} : (i,j) \in \mathrm{J\_}\}$$

Arc (g,h) is the entering arc.

5. Compute values θ_1 and θ_2 where:

$$\theta_1 = x_{k_1 l_1} = \min\left\{-x_{ij} \,\middle|\, (i,j) \in I\_ \ and (i,j) \uparrow\uparrow (g,h)\right\}$$

$$\theta_2 = x_{k_2 l_2} = \min\left\{-x_{ij} \,\middle|\, (i,j) \in I_+ and (i,j) \uparrow\downarrow (g,h)\right\}$$

If $\theta_1 \leq \theta_2$ then the leaving arc is $(k,1) = (k1)$, otherwise the leaving arc is $(k,1) = (k_2, 1_2)$.

6. For the new tree-solution T, compute the flows x_{ij} and the dual problem variables w_i and s_{ij}. Repeat the process from step 2.

The algorithm is presented in more detail in the next Section, where an illustrative example is given.

4. AN ILLUSTRATIVE EXAMPLE

We'll now give an illustrative, step by step, example where the algorithm presented will be applied to a MCNFP. Figure 1 shows a network G = (N, A), consisting of 6 nodes and 12 arcs. Next to each node there is a value showing the node's supply (negative values mean demands). For every arc in A, the cost per product unit flow is also shown.

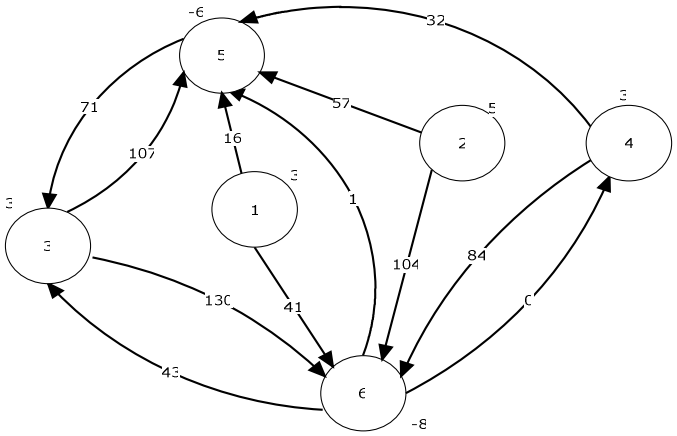

Figure 4.1: The graph G = (N,A) where DNEPSA is applied

We will apply below the algorithm's steps to find a solution for the MCNFP as applied to graph G. The algorithm finds an optimal solution after 3 iterations.

Iteration 1

Step-1. In order to start, the algorithm needs an initial dual feasible basic tree-solution. Figure 2 below shows such a dual feasible tree. Such an initial solution can be obtained by using existing techniques, as it was said in Section 2.

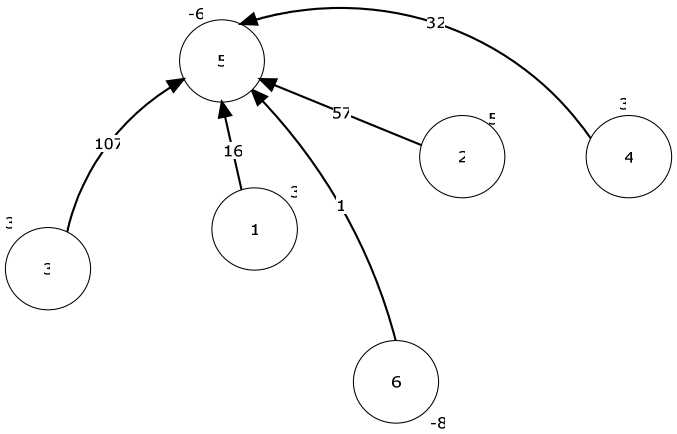

Figure 4.2: The initial dual feasible tree

The tree shown is a dual feasible tree, since it is $s_{ij} \geq 0$ for every arc (i,j). This can be easily verified by solving equations (3.1), which take the following form:

$$w_5 - w_1 = 16$$

$$w_5 - w_2 = 57$$

$$w_5 - w_3 = 107$$

$$w_5 - w_4 = 32$$

$$w_5 - w_6 = 1$$

By setting w_1 equal to 0, we have $w_5 = 16$, $w_2 = -41$, $w_3 = -91$, $w_4 = -16$ and $w_6 = 15$. By applying equations (3.2) we have:

$$s_{16} = c_{16} + w_1 - w_6 = 41 + 0 - 15 = 26$$

$$s_{26} = c_{26} + w_2 - w_6 = 104 + (-41) - 15 = 48$$

$$s_{36} = c_{36} + w_3 - w_6 = 130 + (-91) - 15 = 24$$

$$s_{46} = c_{46} + w_4 - w_6 = 84 + (-16) - 15 = 53$$

$$s_{53} = c_{53} + w_5 - w_3 = 71 + 16 - (-91) = 178$$

$$s_{64} = c_{64} + w_6 - w_4 = 0 + 15 - (-16) = 31$$

$$s_{63} = c_{63} + w_6 - w_3 = 43 + 15 - (-91) = 149$$

while, for all the basic arcs it is $s_{ij} = 0$.

For every node i in the graph, it is

$$\sum_{(i,k)\in N} x_{ik} - \sum_{(j,i)\in N} x_{ji} = b_i$$

This happens because for every node the outgoing flow has to be equal to the incoming flow plus the node's supply. That is, the following equations have to be satisfied:

$$node\,1 : x_{15} = 3$$

$$node\,2 : x_{25} = 5$$

$$node\,3 : x_{35} = 3$$

$$node\,4 : x_{45} = 3$$

$$node\,4 : x_{45} = 3$$

$$node\,5 : -x_{15} - x_{25} - x_{35} - x_{45} - x_{65} = -6$$

$$node\,6 : x_{65} = -8$$

By solving the above equations we have $x_{15} = 3$, $x_{25} = 5$ $x_{45} = 3$ $x_{65} = -8$. This is not a feasible tree since we found some negative flows (it is $x_{65} < 0$)

Step-2 It is I_ = {(6,5)}, since it is $x_{65} = -8 < 0$ *and the remaining arcs form* I_+.

Step-3 If we add the non-basic arc (1,6) to the basic tree, a cycle C is created. In this cycle, arc (1,6) has the same orientation as (6,5) which belongs to I_. So, (1,6) $\in J\_$. *By checking in a similar way all the non-basic arcs, it comes out that* $J\_ = \{(1,6),(2,6),(3,6),(4,6)\}$ *with* $d_{16} = d_{26} = d_{36} = d_{46} = 1$.

Step-4 For the arcs in J_ it is $s_{16} = 26$, $s_{26} = 48$, $s_{36} = 48$, *and* $s_{46} = 53$. *It is*

$$\frac{s_{36}}{d_{36}} = \frac{24}{1} = \min\{\frac{26}{1}, \frac{48}{1}, \frac{24}{1}, \frac{53}{1}\}$$

So arc (g,h)=(3,6) is the entering arc.

Step-5 After adding the entering arc (3,6) to the basic tree, a cycle C is created, as shown in Figure 3.

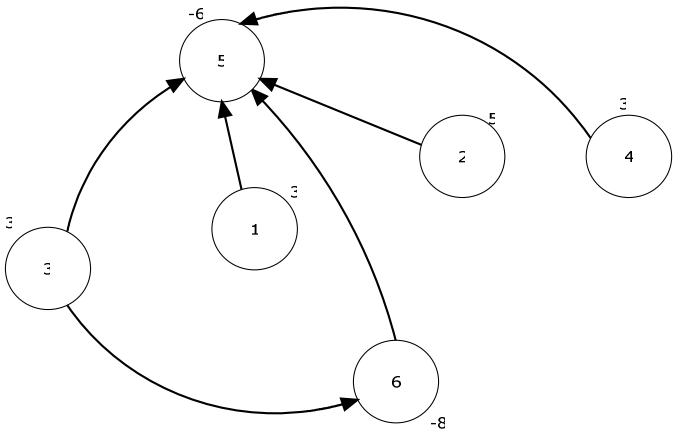

Figure 4.3: The cycle created after adding the entering arc

Arc (6,5) belongs to I_ and has the same orientation as the entering arc (3,6). So, it is $\theta_1 = -x_{65} = 8$. Arc (3,5), on the other hand, belongs to I_+ and does not have the same orientation as the entering arc. So, it is $\theta_2 = x_{35} = 3$. We have $\theta_1 > \theta_2$ which means that arc (3,5) is the leaving arc.

Step-6 The tree shown in Figure 4 is now the new basic tree.

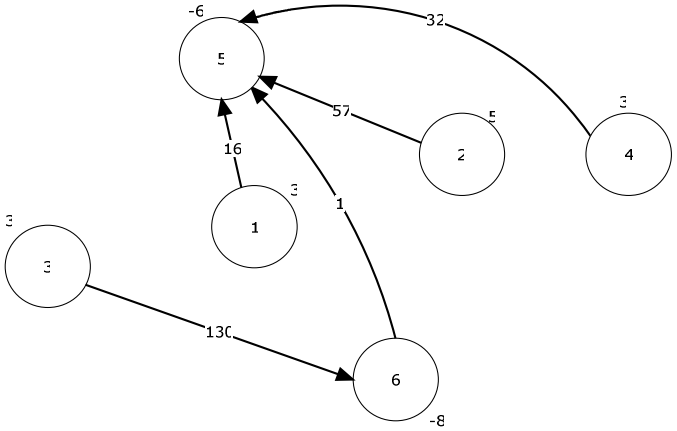

Figure 4.4: The new basic tree-solution

For the new tree it is $x_{15}=3$, $x_{25}=5$, $x_{36}=3$, $x_{36}=3$, $x_{45}=3$ and $x_{65}=-5$. This is not a feasible tree, so the process has to be continued. By using formulas (3.1) and (3.2), the same way as for the starting basic tree, we find that $s_{16}=26$, $s_{26}=48$, $s_{35}=-24$, $s_{46}=53$, $s_{53}=202$, $s_{64}=31$ and $s_{63}=173$. This new basic tree is not a dual feasible tree-solution.

Iteration 2

Step-2 It is I_ = {(6,5)}, since it is x_{65} = -5 < 0 and the remaining arcs form I_+.
Step-3 By checking, as in the first iteration, what happens when the non-basic arcs are added to the basic tree and their orientation, we find that it is J_={(1,6),(2,6),(4,6),(5,3)} with $d_{16}=d_{26}=d_{46}=d_{53}=1$.
Step-4 For the arcs in J_ it is $s_{16}=26, s_{26}=48, s_{46}=53$ and $s_{53}=202$. It is

$$\frac{s_{16}}{d_{16}}=\frac{26}{1}=\min\{\frac{26}{1},\frac{48}{1},\frac{53}{1},\frac{202}{1}\}$$

So arc (g,h)=(1,6) is the entering arc.
Step-5 After adding the entering arc (1,6) to the tree, a cycle C is created as shown in Figure 5.

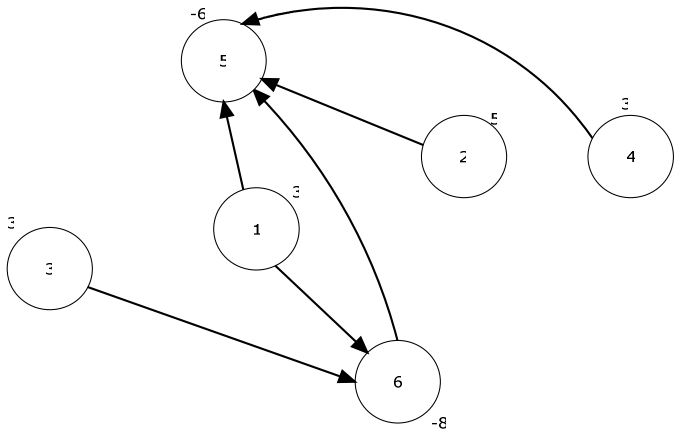

Figure 4.5: The basic tree after adding the entering arc

Arc (6,5) belongs to I_ and has the same orientation as the entering arc (1,6). So, it is $\theta_1=-x_{65}=5$. Arc (1,5), on the other hand, belongs to I_+ and does not have the same orientation as the entering arc. So, it is $\theta_2=x_{15}=3$. We have $\theta_1>\theta_2$ so, arc (1,5) is the leaving arc.
Step-6 The tree, shown in Figure 6, forms now the new basic tree-solution.

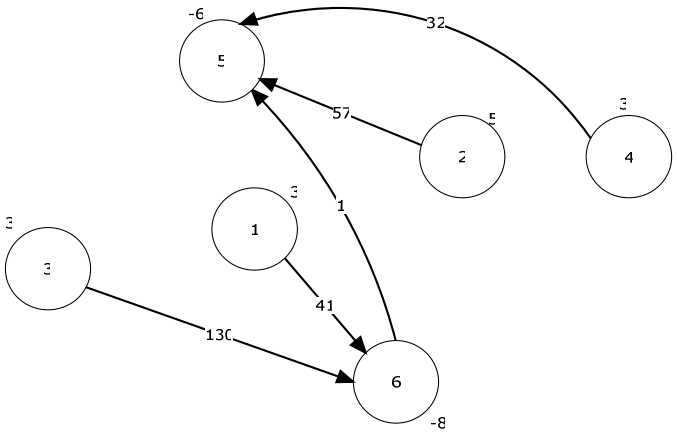

Figure 4.6: The new basic tree-solution

For the new tree, it is $x_{16}=3, x_{25}=5, x_{36}=3, x_{45}=3$ *and* $x_{65}=-2$ i.e. it is not a feasible tree. It is also:

$$s_{15}=-26, s_{26}=48, s_{35}=-24, s_{46}=53, s_{53}=202, s_{64}=31 \text{ and } s_{63}=173 .$$

Iteration 3

Step-2 It is $I_{-}=\{(6,5)\}$*, since it is* $x_{65}=-2<0$ *and* I_{+} *contains the remaining arcs.*
Step-3 Similarly, as in the previous iterations, we find that $J_{-}=\{(2,6),(4,6),(5,3)\}$ *and* $d_{26}=d_{46}=d_{53}=1$.
Step-4 For the arcs in J_{-} *it is* $s_{26}=48, s_{46}=53$ *and* $s_{53}=202$ *. It is*

$$\frac{s_{26}}{d_{26}}=\frac{48}{1}=\min\{\frac{48}{1},\frac{53}{1},\frac{202}{1}\}$$

So, arc (g,h)=(2,6) is the entering arc.
Step-5 If we add arc (2,6) to the tree, a cycle C is created, as shown in Figure 7.

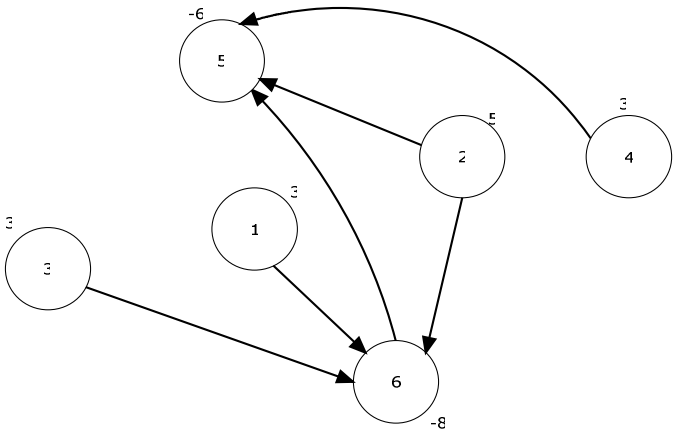

Figure 4.7: The basic tree after adding the entering arc

Arc (6,5) belongs to I_ and has the same orientation as the entering arc (2,6). So, it is $\theta_1 = -x_{65} = 2$. Arc (2,5), on the other hand, belongs to I_+ and does not have the same orientation as the entering arc. So it is $\theta_2 = x_{25} = 5$. It is $\theta_1 \leq \theta_2$ so arc (6,5) is the leaving arc.

Step-6 Therefore, the new tree shown in Figure 8 is now the basic tree.

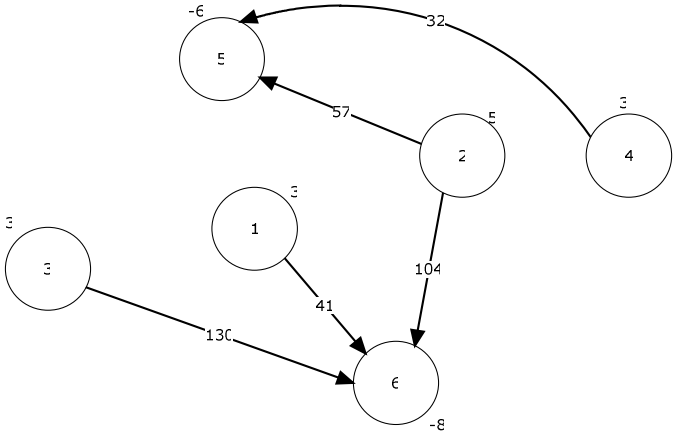

Figure 4.8: The new basic tree-solution

For this tree it is $x_{16} = 3, x_{36} = 3, x_{45} = 3, x_{26} = 2$ *and* $x_{25} = 3$, i.e. it is a primal feasible tree-solution and also a dual feasible tree since it is $s_{15} = 22, s_{35} = 24, s_{46} = 5, s_{53} = 154, s_{64} = 79, s_{65} = 48$ *and* $s_{63} = 173$. Therefore, we have

found an optimal solution. This is detected immediately by our algorithm since it is $I\_ = \varnothing$ and the algorithm stops.

5. CONCLUSIONS AND FUTURE WORK

First of all, we plan to present all the necessary mathematical proof of correctness of the proposed algorithm in a future work. DNEPSA is based on a set of lemmas and theorems, which were omitted in this paper, due to paper length. Furthermore, there are special improved data structures that could be used in order to improve the performance of DNEPSA. Such data structures include dynamic trees and Fibonacci heaps, as described in [10], [16] and [6]. There are also other useful data structures and techniques, as presented in [3] and [5], which obtain very good performance, by applying various operations on graphs and trees.

There are several state-of-the-art algorithm implementations that can be applied to the MCNFP. For example, implementations like RELAX IV, NETFLO and MOSEK demonstrate very good performance results. Some of these implementations are described in [4] and [14]. At this point, DNEPSA has already been developed (in C programming language) and tested thoroughly against a big variety of problems. Therefore, it will be very interesting to compare DNEPSA against some of the well known MCNF algorithms

Finally, DNEPSA will be incorporated into the Network Optimization suite WebNetPro which is described in [13]. This way, WebNetPro's capabilities will be extended by adding new algorithms into this web suite.

REFERENCES

[1] Ahuja, R., Magnanti, T., Orlin, J., and Reddy, M., "Applications of network optimization", *Handbooks of Operations Research and Management Science,* (1995) 1-83.

[2] Ahuja, R., and Orlin, J., "Improved primal simplex algorithms for shortest path, assignment and minimum cost flow problems", Massachusetts Institute of Technology, Operations Research Center, Massachusetts Institute of Technology, Operations Research Center, Working Paper OR 189-88, (1988).

[3] Ali, A.I., Helgason, R.V., Kennington, J.L., and Lall, H.S., "Primal simplex network codes: State-of-the-art implementation technology", *Networks,* 8 (4) (1978) 315-339.

[4] Bertsekas, D. P., and Tseng, P., "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, 1994.

[5] Eppstein, D., "Clustering for faster network simplex pivots", *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms,* 1994.

[6] Fredman, M., and Tarjan, R., "Fibonacci heaps and their uses in improved network optimization algorithms", *Journal of the ACM,* 34 (3) (1987) 596-615.

[7] Glover, F., Karney, D., and Klingman, D., "Implementation and computational comparisons of primal, dual and primal-dual computer codes for minimum cost network flow problems", *Networks,* 4 (3) (1974) 191-212.

[8] Glover, F., Klingman, D., and Napier, A., "Basic dual feasible solutions for a class of generalized networks", *Operations Research,* 20 (1) (1972) 126-136.

[9] Glover, F., Klingman, D., and Phillips, N., *Network Models in Optimization and Their Applications in Practice*, Wiley Publications, 1992.

[10] Goldberg, A., Grigoriadis, M., and Tarjan, R., "Use of dynamic trees in a network simplex algorithm for the maximum flow problem", *Mathematical Programming,* 50 (3) (1991) 277-290.

[11] Hultz, J., and Klingman, D., "An advanced dual basic feasible solution for a class of capacitated generalized networks", *Operations Research*, 24 (2) (1976).

[12] Karagiannis, P., Markelis, I., Paparrizos, K., Samaras, N., and Sifaleras, A., "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) (2006) 765-782.

[13] Karagiannis, P., Paparrizos, K., Samaras, N., and Sifaleras, A., "A new simplex type algorithm for the minimum cost network flow problem", *Proceedings of the 7th Balkan Conference on Operational Research,* 2005, 133-139.

[14] Kennington, J.L., and Helgason, R.V., *Algorithms for Network Programming*, Wiley Publications, 1980.

[15] Orlin, J., "Genuinely polynomial simplex and non-simplex algorithms for the minimum cost flow problem", Sloan School of Management, M.I.T., Cambridge, MA, Technical Report No. 1615-84, 1984.

[16] Tarjan, R. E., "Dynamic trees as search trees via Euler tours, applied to the network simplex algorithm", *Mathematical Programming*, 78(2) (1997) 169-177.