On the Computational Behavior of a Dual Network Exterior Point Simplex Algorithm for the Minimum Cost Network Flow Problem

George Geranis*, Konstantinos Paparrizos*, Angelo Sifaleras*

**Department of Applied Informatics, University of Macedonia*
Greece, Thessaloniki, 156 Egnatia Str., Postal Code 54006

Abstract

The Minimum Cost Network Flow Problem (MCNFP) constitutes a wide category of network flow problems. A Dual Network Exterior Point Simplex Algorithm (DNEPSA) for the MCNFP is presented here, together with some computational results. Similarly to the classical Dual Network Simplex Algorithm (DNSA), the new algorithm starts with a dual feasible tree-solution and, after a number of iterations, it produces an optimal solution. However, contrary to DNSA, our algorithm does not always maintain a dual feasible solution. Instead, it produces tree-solutions that can be infeasible for both, the dual and the primal problem. This family of algorithms is believed to be more efficient than the classical Simplex-type algorithms. They can cross over the infeasible region of the dual problem and find an optimal solution reducing the number of iterations needed. In this paper, we describe the algorithm and, for the first time, we present its computational behavior compared to the Dual Network Simplex Algorithm.

Keywords*: Operational Research, Minimum Cost Network Flow Problem, Dual Network Exterior Point Simplex Algorithm*

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 supply nodes (sources), demand nodes (sinks) and transshipment nodes. There have been developed various Simplex-type algorithms for the MCNFP. Such Simplex-type algorithms for the MCNFP include the well-known Primal and the Dual Network Simplex Methods. There exist also other non Simplex-type algorithms that can be used for the MCNFP, as presented in [2], [7] and [11]. An exterior-point primal simplex-type algorithm for the MCNFP has also been presented in [12].

This paper presents, for the first time, some preliminary computational results of a new exterior point dual simplex-type algorithm for the MCNFP. The algorithm is named Dual Network Exterior Point Simplex Algorithm (DNEPSA) for the MCNFP and it is described in [5]. DNEPSA starts from a dual feasible tree-solution and, iteration by iteration, it produces new tree-solutions closer to an optimal solution. Contrary to the Network Dual Simplex Algorithm, the tree-solution at every iteration is not necessarily dual feasible. The algorithm can move outside the feasible region of the dual problem and return back to it in order to find an optimal solution.

2 Problem Statement and Notation

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 value b_i representing the available supply or demand at that node. The total supply must be equal to the total demand, i.e. it is $\sum_{i \in N} b_i = 0$. For each arc $(i,j) \in A$ there is an associated flow x_{ij} of product units from node i to node j and an associated cost per product unit c_{ij}. The total flow cost is equal to $\sum_{(i,j) \in A} c_{ij} x_{ij}$. We may have a lower and an upper bound for every flow x_{ij} on arc (i,j), denoted l_{ij} and u_{ij} respectively. DNEPSA can be applied to *uncapacitated* MCNFPs, that is we have $l_{ij} = 0$ and $u_{ij} = +\infty, \forall (i,j) \in A$. For every node $i \in N$, the outgoing flow must be equal to the incoming flow plus the node's supply. The MCNFP is formulated is formulated as follows:

$$\begin{aligned} &\min \sum_{\mathrm{(i,j)\in A}} \mathrm{c_{ij}} x_{ij} \\ &s.t. \sum_{\mathrm{i:(k,i)\in A}} x_{ki} - \sum_{\mathrm{j:(j,k)\in A}} x_{jk} = b_k, k \in N \\ &0 \leq x_{ij} \leq +\infty, \forall (i,j) \in A \end{aligned} \tag{1}$$

There is a set of dual variables w_i, one for each node, and a number of reduced cost variables s_{ij}, one for every directed arc, used for the formulation of the dual problem. If for a tree-solution T, it is $x \geq 0$, then it is a *primal feasible* solution, while if $s \geq 0$, it is a *dual feasible* solution. Primal network simplex-type algorithms start from a primal feasible tree-solution and they move, at every iteration, to a new primal feasible solution, until they find an optimal solution. On the other hand, dual network simplex-type algorithms start from a dual feasible solution and they reach an optimal solution, by following a route of dual feasible solutions. DNEPSA, although it starts from a dual feasible solution, moves through solutions that are not necessarily dual feasible but, after a number of iterations, it comes to a tree-solution that is both primal and dual feasible, i.e. it is optimal.

3 Algorithm Description

DNEPSA needs a starting dual feasible tree solution T. There are different techniques that can be used in order to find such a solution. Such an algorithm, for the generalized network problem, is described in [9]. The arcs of T and the corresponding flows are called *basic arcs* and *basic variables* respectively. For the non basic arcs $(i,j) \notin T$ it is $x_{ij} = 0$ and $s_{ij} \geq 0$, while for the basic arcs $(i,j) \in T$ it is $s_{ij} = 0$. For the dual variables w_i, $1 \leq i \leq n$ it is:

$$w_i - w_j = c_{ij}, \ \forall (i,j) \in T \tag{2}$$

In (2) we have n-1 equations and n variables. By giving any value to an arbitrarily chosen variable, we can compute the values for the rest of the variables. The reduced costs s_{ij} for the non-basic arcs (i,j) can be calculating as follows:

$$s_{ij} = c_{ij} - w_i + w_j, \ \forall (i,j) \notin T \tag{3}$$

The algorithm uses a set, named I_, that contains the basic arcs (i,j) having negative flow and a set I_+ containing the rest of the basic arcs:

$$I_- = \{(i,j) \in T : x_{ij} < 0\} \ , \ I_+ = \{(i,j) \in T : x_{ij} \geq 0\} \tag{4}$$

If it is $I_- = \emptyset$, then the current tree-solution is optimal. If a non-basic arc (i,j) is added into the basic tree T, then a cycle C is created. In that case, let h be the vector of orientations of all basic arcs relative to the entering arc (i,j). If an arc (u,v) in C has the same orientation as (i,j), then it is $h_{uv} = -1$, otherwise it is $h_{uv} = +1$. For an arc (u,v) not belonging to C, it is $h_{uv} = 0$. DNEPSA keeps in touch with the dual feasible region by maintaining a vector d that gives a direction to the dual feasible region:

$$\begin{array}{l} d_{ij} = 1,\ \ if\,(i,j) \in I_- \\ d_{ij} = 0,\ \ if\,(i,j) \in I_+ \\ d_{ij} = \sum\limits_{(u,v)\in I_-} h_{uv},\ \ if\,(i,j) \notin T \end{array} \tag{5}$$

DNEPSA also maintains a set J₋, that is:

$$J_- = \{(i,j) \notin T : s_{ij} > 0 \ and \ d_{ij} < 0\} \tag{6}$$

If it is $J_- = \emptyset$, while it is $I_- \neq \emptyset$, then the problem is infeasible. By using set J₋, the algorithm calculates the following minimal ratio:

$$\alpha = \frac{s_{gh}}{-d_{gh}} = \min\{\frac{s_{ij}}{-d_{ij}} : (i,j) \in J_-\} \tag{7}$$

Ratio α is used in order to choose the *entering arc* (g,h). In order to choose the *leaving arc (k,l)*, DNEPSA finds in I₋ arc (k_1, l_1) of minimum absolute flow, having the same orientation as the entering arc (g,h). Similarly, DNEPSA finds in I_+ arc (k_2, l_2) of minimum flow, having orientation opposite to (g,h). It is:

$$\begin{array}{l} \theta_1 = -x_{k_1l_1} = \min\{-x_{ij} : (i,j) \in I_- \ \ and \ \ (i,j) \uparrow\uparrow (g,h)\} \\ \theta_2 = \ \ x_{k_2l_2} = \min\{\ x_{ij} : (i,j) \in I_+ \ \ and \ \ (i,j) \uparrow\downarrow (g,h)\} \end{array} \tag{8}$$

where notation $\uparrow\uparrow$ is used for arcs that have the same orientation, while notation $\uparrow\downarrow$ stands for arcs having the opposite orientation to each other. If it is $\theta_1 \leq \theta_2$, then arc (k_1, l_1) is the leaving arc. In that case, we say we have a *type A iteration.* An arc of negative flow $x_{kl} = -\theta_1$ is leaving and an arc with flow $x_{gh} = \theta_1$ is entering the basic solution. If it is, on the other hand, $\theta_1 > \theta_2$, then arc (k_2, l_2) is the leaving arc. In that case, we say we have a *type B iteration.* An arc of positive flow $x_{kl} = \theta_2$ is leaving and an arc with flow $x_{gh} = \theta_2$ is entering the basic solution. Figure 1 gives a diagram describing type A and type B iterations. The subtree containing node k of the leaving arc (k,l) is symbolised as T^+, while the other subtree is symbolised as T^-.

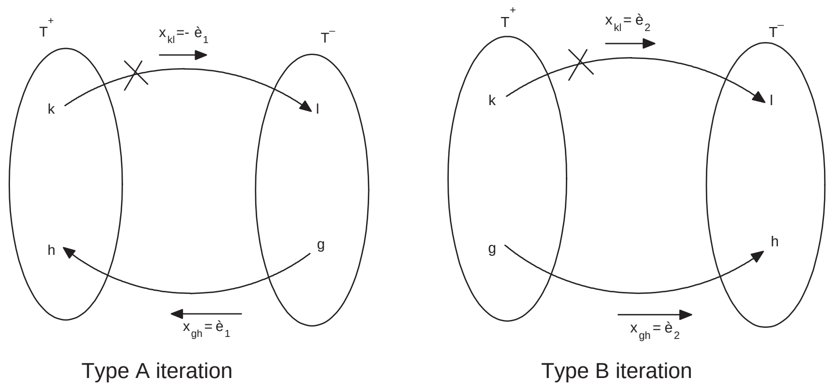

Figure 1: *Type A and B iterations*

The same procedure continues until the algorithm finds an optimal solution or it discovers infeasibility. The algorithm steps are shown in Algorithm 1.

It is not necessary for the algorithm to compute the values of the variables x_{ij}, s_{ij} and d_{ij} or to create sets I_- and I_+ from scratch. They can be efficiently updated by using their values in the tree-solution of the previous iteration.

Algorithm 1 Algorithm DNEPSA

```
Require: $G = (N, A), b, c, T$
 1: procedure DNEPSA($G, T$)
      Step 0 (Initializations)
 2:     Compute $x$, $w$ and $s$ using relations (1), (2) and (3) respectively
 3:     Find sets $I_-$ and $I_+$ from (4) and vector d from (5).
      Step 1 (Test of optimality)
 4:     while $I_- \neq \emptyset$ do
 5:         Find set $J_-$ from relation (6)
 6:         if $J_- = \emptyset$ then
 7:             STOP, the problem is infeasible
 8:         else
      Step 2 (Choice of the entering arc)
 9:             Compute $\alpha$, using Relation (7) and choose the entering arc $(g, h)$
      Step 3 (Choice of the leaving arc)
10:             Compute $\theta_1$, $\theta_2$ using Relations (8)
11:             Choose the leaving arc $(k, l)$ given by the minimum of $\theta_1$ and $\theta_2$
      Step 4 (Pivoting)
12:             Set $T = T \setminus (k, l) \cup (g, h)$ and update $x$, $s$ and $d$
13:             if $\theta_1 \leq \theta_2$ then
14:                 Set $I_- = I_- \setminus (k, l)$ and $I_+ = I_+ \cup (g, h)$
15:             else
16:                 Set $I_+ = I_+ \cup (g, h) \setminus (k, l)$
17:             end if
18:         end if
19:     end while
20: end procedure
```

4 Implementation and Computational Results

In order to evaluate the performance of DNEPSA, we performed a computational study comparing DNEPSA against the classic Dual Network Simplex Algorithm (DNSA). All the MCNF problem instances were created using the well-known NETGEN random graph generator [10]. The experiments were performed on a PC with an Intel P4 3.6 GHz processor and 1 GB RAM DDR 2, running the Windows XP Pro SP2 operating system. The competitive algorithms was programmed in C and compiled using the gcc compiler. The functions used for the implementation of the algorithms have been written following the same programming techniques adjusted to the special characteristics of each algorithm. Both algorithms were implemented by using the Augmented Thread Index method [6]. This method was chosen because it allows the fast update of the basic tree, as also it can easily identify the cycle that would have been created with the addition of the entering arc. The two algorithms need an initial starting dual feasible solution. The same starting point was used for both algorithms. The time needed to find the starting solution was included into the measurements.

DNEPSA proved to be superior to Dual Network Simplex algorithm for networks of density 20% or more. This superiority concerns both the number of the iterations and the time needed in order to find an optimal solution. For networks of smaller density, DNEPSA appears to be superior in time (due to faster updating techniques) but it seems to be almost equivalent to DNSA when concerning the number of iterations needed. Table 1 shows the results of running DNEPSA and Dual Network Simplex algorithm on a number of various problems produced randomly by NETGEN. Different network sizes were used, starting from networks of size 50x588 up to networks of size 500x59700. For every network size a number of 20 different problems were solved and the mean values of the number of iterations and the time needed are shown into the table. DNEPSA's superiority can be seen in the last two columns of the table which

Table 1: Comparison of DNEPSA and Dual Network Simplex Algorithm (DNSA)

Network	DNEPSA niters	DNEPSA time	DNSA niters	DNSA time	% niters	% time
50x588	57.55	0.02	65.65	0.03	87.66%	78.20%
100x2376	167.50	0.29	190.30	0.36	88.02 %	80,35%
150x5364	348.34	1.61	389.85	2.03	89.35%	79.48%
200x9552	597.55	5.70	645.95	6.91	92.51%	82.46%
250x14940	960.12	14.95	1027.07	18.60	93.48%	80.37%
300x21528	1124.23	29.95	1252.67	37.31	89.75%	80.27%
350x29316	1754.34	65.70	1872.20	78.39	93.70%	83.81%
400x38404	2442.21	126.50	2736.63	157.73	89.24%	80.20%
450x48492	3277.34	215.50	3641.67	260.38	90.00%	82.76%
500x59700	4199.50	379.90	4870.80	483.91	86.22%	78.51%

contain the percentage of the number of iterations and time compared that DNEPSA needs compared to DNSA.

An explanation of DNEPSA's superiority is the fact that DNEPSA can cross over the infeasible region of the dual problem and return back by finding an optimal solution. This leads to an essential reduction on the number of iterations.

5 Future Work

A subject for future work is the improvement of the performance of the algorithm by using special data structures for storing and updating the necessary variables. Such data structures include dynamic trees and Fibonacci heaps, as described in [14] and [4]. It would be very interesting to use such data structures and compare DNEPSA's performance against some state-of-the-art algorithms. Such state-of-the-art algorithms include RELAX IV, combinatorial code CS2, interior-point code DLNET and NETFLO, as described in [3], [8] and [13]. The algorithm's behavior has also to be examined in some well-known pathological instances, as described in [15]. Furthermore, it would be interesting to develop a capacitated version of DNEPSA, although it is possible for any capacitated network to be transformed into an uncapacitated equivalent one by removing arc capacities. This technique is analytically described in [1].

References

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

[2] R.K. Ahuja and J. Orlin, *Improved Primal Simplex Algorithms for Shortest Path, Assignment and Minimum Cost Flow Problems*, Massachusetts Institute of Technology, Operations Research Center, Working Paper, OR 1988, pp. 189–188.

[3] D.P. Bertsekas and P. Tseng, *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.

[4] M. Fredman and R.E. Tarjan, *Fibonacci heaps and their uses in improved network optimization algorithms*, J. ACM, **34**(3) (1987) pp. 596–615.

[5] G. Geranis, K. Paparrizos and A. Sifaleras, *A dual exterior point simplex type algorithm for the minimum cost network flow problem*, in 8^{th} *Balkan Conference on Operational Research*, 14-17 September, Belgrade-Zlatibor, Serbia, 2007, pp.139–149.

[6] F. Glover, D. Karney and D. Klingman, *The augmented predecessor index method for locating stepping stone paths and assigning dual prices in distribution problems*, Transport Sci. **6**(2) (1972) pp. 171–180.

[7] F. Glover, D. Karney and D. Klingman, *Implementation and Computational Comparisons of Primal, Dual and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems*, Networks, **4**(3) (1974) pp. 191–212.

[8] A.V. Goldberg, *An Efficient Implementation of a Scaling Minimum-Cost Flow Algorithm*, J. Algorithms, **22**(1) (1997) pp. 1–29.

[9] J. Hultz, D. Klingman and R. Russell, *An Advanced Dual Basic Feasible Solution for a Class of Capacitated Generalized Networks*, Oper. Res. **24**(2) (1976) pp. 301–313.

[10] D. Klingman, A. Napier and J. Stutz, *NETGEN: A program for generating large scale capacitated assignment, transportation and minimum cost flow networks*, Manag. Sci. **20**(5) (1974) pp. 814–821.

[11] J. Orlin, *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).

[12] K. Paparrizos, N. Samaras and A. Sifaleras, *An exterior Simplex type algorithm for the minimum cost network flow problem*, Comput. Oper. Res. **36**(4) (2009) pp. 1176-1190.

[13] M. Resende and G. Veiga, *An efficient implementation of a network interior point method*, in: *Network flows and matching*, D.S. Johnson and C.C. McGeoch, eds., First DIMACS implementation challenge, American Mathematical Society, Providence, Rhode Island, DIMACS series in discrete mathematics and theoretical computer science **12** 1993 pp. 299–348.

[14] R.E. Tarjan, *Dynamic Trees as Search Trees via Euler Tours, Applied to the Network Simplex Algorithm*, Math. Program., **78**(2) (1997) pp. 169–177.

[15] N. Zadeh, *More Pathological Examples for Network Flow Problems*, Math. Program. **5**(1) (1973) pp. 217–224.