A NETWORK SIMPLEX ALGORITHM FOR THE MINIMUM COST-BENEFIT NETWORK FLOW PROBLEM

Cenk Çalışkan, Utah Valley University, 800 W. University Parkway, Orem, UT 84058, 801-863-6487, cenk.caliskan@uvu.edu

ABSTRACT

The minimum cost-benefit network flow problem is to find a minimum cost flow over a capacitated, directed network for which the total benefit of flow must be at least a specified minimum value, K; where each unit of flow on a given arc incurs a constant cost and provides a constant benefit. This problem is an extension of the classical minimum cost network flow problem in which a single side constraint is added. In this research, we describe a very efficient specialized simplex algorithm for this problem that is implemented directly on the network.

Keywords: minimum cost network flow, network simplex, partitioning, factorization

INTRODUCTION

Let *G=(V, E)* be a directed network consisting of a set V of vertices (nodes) and a set E of edges (arcs). In this network, the flow on each arc *(i,j)* is represented by the nonnegative variable x_{ij} with a cost c_{ij}, benefit k_{ij} and capacity u_{ij}. Let b_i be the supply/demand of each node i. The minimum cost-benefit network flow problem is to minimize the total cost of flow over network *G* that sends a total of b_i units of flow from each supply node *i* and receives a total of $-b_i$ units of flow at each demand node *i*, where the total benefit of flow is at least *K*. The problem then can be formulated as the following linear program:

$$\min \sum_{(i,j)\in E} c_{ij} x_{ij} \tag{1}$$

s.t.

$$\sum_{(i,j)\in E} x_{ij} - \sum_{(j,i)\in E} x_{ji} = b_i \quad \forall i \in V \tag{2}$$

$$\sum_{(i,j)\in E} k_{ij} x_{ij} \geq K \tag{3}$$

$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i,j) \in E \tag{4}$$

The minimum cost-benefit network flow problem is a special case of the minimum cost network flow problem with side constraints. Minimum cost network flow problem with side constraints may be solved very efficiently in practice using specialized simplex algorithms that exploit the network substructure present in the coefficient matrix of the LP. However, these algorithms work with multiple side constraints, so they perform matrix operations as well as network simplex operations and since they are more general, they usually do not fully exploit the additional special properties of the problem when there is only a single constraint. Klingman and Russell [7] study the transportation problem with a single side constraint and propose a specialized simplex algorithm. However, their algorithm is not extendable to the more general minimum cost-benefit network flow problem. Graves and McBride [6], Chen and Saigal [4], and McBride [9] present specialized simplex algorithms for the minimum cost network flow problem with (multiple) side constraints, but they are not directly applicable to the minimum cost-benefit network flow problem. Çalışkan [3] studies the constrained maximum flow problem, which is related to the minimum cost-benefit network flow problem, but their algorithm isn't directly applicable to the present problem. We believe that it is important to study the singly-constrained minimum cost network flow problem separately as it is an important problem that is encountered frequently in practice, and it lends itself nicely to a network simplex algorithm that does not require any matrix operations, re-factorizations, or any other complicated non-network operations.

There is a related problem that can be transformed into minimum cost-benefit network flow problem. In this problem, instead of minimizing the total cost, the objective is maximizing the total benefit, subject to a budget limit of D on the total cost of flow. We will call this problem *maximum benefit-cost network flow problem*. It is easy to see that maximum benefit-cost network flow problem can also be solved as a minimization problem, and the side constraint can be converted to a ≥ constraint. These problems appear very frequently in practice. For example, in government projects, the objective is usually maximizing benefit to the citizens subject to a budget. Another example concerns the environmental issues. There may be environmental costs associated with each unit of flow on each arc, and the objective could be to minimize the total financial cost of flow subject to an upper limit on the environmental cost. Alternatively, the objective can be to maximize the total benefit to the citizens, subject to an upper limit on the environmental cost or environmental risk. We can think of many examples like these that appear in practice and they all can be solved using the approach presented in this paper; regardless of whether the objective is to maximize or minimize, and regardless of whether the side constraint is ≤, =, or ≥. Without loss of generality, we will consider the minimum cost-benefit network flow problem in this paper.

THE SPECIALIZED NETWORK SIMPLEX ALGORITHM

In this section, we specialize the simplex algorithm by partitioning the basis into a tree and a non-tree part. Our approach is similar to those of Graves and McBride [6], Chen and Saigal [4], and Mamer and McBride [9]. They all consider the minimum cost network flow problem with multiple side constraints. Our approach is specialized for the minimum cost-benefit network flow problem, so it exploits the additional nice properties that are associated with a single side constraint.

Transformation of the Basis

In this section, we describe how a basis (matrix) of the minimum cost-benefit network flow problem can be transformed into an upper triangular matrix so that all simplex operations can be carried out directly on the network.

The excess variable for the minimum benefit constraint (Eq. 3) can be thought of an arc that forms a loop from a node to itself and that has a benefit of -1 and a cost of 0. Since it does not appear in any of the balance equations (Eq. 2), we can associate it with any of the nodes. Without loss of generality, we will assume that it is a loop on node 1. To be consistent with the rest of the notation, we will denote the excess variable x_{11}. With the introduction of x_{11}, Eq. (3) becomes an equality constraint.

A basis of Eqs. (1)-(4) consists of arcs that form a spanning tree, and an additional non-tree arc. We denote the non-tree arc (α, β). This non-tree arc forms a unique cycle on the tree, which is called a fundamental cycle. We will denote this fundamental cycle $\psi_{\alpha\beta}$. Note that if x_{11} is basic, it is always the non-tree arc; which means that it is the fundamental cycle itself. Let B be a basis (matrix). Then, we can express B as the following partition:

$$B = \begin{bmatrix} A & a_{\alpha\beta} \\ k_A & k_{\alpha\beta} \end{bmatrix}, \tag{5}$$

where A is the node-arc incidence matrix of the spanning tree; $a_{\alpha\beta}$ is the node-arc incidence vector of *(α, β)*; k_A is the benefit vector of the spanning tree arcs; and $k_{\alpha\beta}$ is the benefit of *(α, β)*. We denote the benefit vector of all basic variables in *B*, k_B. Let the node-arc incidence matrix of nonbasic arcs at their upper bounds be *N*. Then, we denote the benefit vector for arcs in *N*, k_N. The flow vector *x* and the cost vector c can also be similarly partitioned: x_A and c_A are the flow and cost vectors for the spanning tree arcs, x_N and c_N are the flow and cost vectors for the non-basic arcs at their upper bounds, x_B and c_B are the flow and cost vectors for all basic variables in *B*, and $x_{\alpha\beta}$ and $c_{\alpha\beta}$ are the flow and cost of the non-tree basic arc *(α, β)*.

Proposition 1 *By a series of column operations, B can be turned into an upper-triangular matrix as follows:*

$$\overline{B} = \begin{bmatrix} A & 0 \\ k_A & \overline{k}_{\alpha\beta} \end{bmatrix}, \tag{6}$$

where $\overline{k}_{\alpha\beta}$ is the net benefit around $\psi_{\alpha\beta}$ when traversed in the direction of *(α, β)*. Furthermore, these column operations correspond to post-multiplying *B* by:

$$B' = \begin{bmatrix} I & -A^{-1}a_{\alpha\beta} \\ 0 & 1 \end{bmatrix} \tag{7}$$

Proof: Consider the unique chain *P* from *β* to *α* in the spanning tree. We can convert this chain to a (directed) path from *β* to *α* by reversing each reverse arc. This corresponds to multiplying the columns of such arcs in the basis by -1. Let the resulting path be *P'*. If we sum the columns corresponding to *P'*, the resulting column will have zeros on all rows but the rows corresponding to *β* and *α* and the last row; which will be 1, -1, and $\sum_{(i,j)\in P'} k_{ij}$, respectively. If we add this sum to the column of *(α, β)*, the resulting column will have zeros in all rows but the last one, which will be $k_{\alpha\beta} + \sum_{(i,j)\in P'} k_{ij} = \sum_{(i,j)\in\psi_{\alpha\beta}} \pm k_{ij}$. This is the net benefit around the fundamental cycle of *(α, β)*. Now, assume that *B'* is a matrix that can be partitioned as follows:

$$B' = \begin{bmatrix} M & n \\ o & p \end{bmatrix}, \tag{8}$$

where *M* is an *(n-1)×(n-1)* matrix, *n* is an *(n-1)×1* column vector, *o* is a *1×(n-1)* row vector, and *p* is a real number. We claim that $\overline{B} = BB'$. Then, it follows that:

$$A = AM + a_{\alpha\beta}o \tag{9}$$

$$0 = An + a_{\alpha\beta}p \tag{10}$$

$$k_A = k_A M + k_{\alpha\beta}o \tag{11}$$

$$\overline{k}_{\alpha\beta} = k_A n + k_{\alpha\beta}p. \tag{12}$$

It can be shown that the solution to Eqs. (9)-(12) is as follows: $M = I$, $n = -A^{-1}a_{\alpha\beta}$, $o = 0$ and $p = 1$. Thus, $\overline{k}_{\alpha\beta} = k_{\alpha\beta} - k_A A^{-1}a_{\alpha\beta}$. But $A^{-1}a_{\alpha\beta}$ is the solution to a system $Az = a_{\alpha\beta}$. Since A is a tree, the solution to this system represents the unique chain from *β* to *α* on the tree. The solution is as follows: $z_{ij} = 1$ if *(i, j)* is oriented opposite to *(α, β)* on the fundamental cycle, $z_{ij} = -1$ if *(i, j)* is oriented in the same direction as *(α, β)*, and $z_{ij} = 0$ otherwise. Thus, $\overline{k}_{\alpha\beta}$ is the net benefit around the fundamental cycle of *(α, β)*, or $\psi_{\alpha\beta}$, when traversed in the direction of *(α, β)*. □

Computing Arc Flows

In this section, we describe how arc flows can be determined directly on the network given a basis *B*, just like in the network simplex algorithm for the minimum cost network flow problem. Upper triangularity of the transformed basis permits us to carry out these computations with a small additional effort compared to the pure network computations.

Proposition 2 *Given a basis matrix B and a matrix of nonbasic arcs N (at their upper bounds), the flow vector x can be determined by solving a simple flow computation problem on the network as follows:*

$$\bar{x}_A = A^{-1}(b - Nx_N) \tag{13}$$

$$x_{\alpha\beta} = \frac{K - k_A\bar{x}_A - k_N x_N}{\bar{k}_{\alpha\beta}} \tag{14}$$

$$x_A = \bar{x}_A - x_{\alpha\beta}A^{-1}a_{\alpha\beta}. \tag{15}$$

Proof: The flow vector x_B is the solution to a system: $Bx_B + Nx_N = b$. Defining $\bar{x}$ such that $x_B = B'\bar{x}$, and using the fact that $\bar{B} = BB'$, we can instead solve the following equivalent system: $\bar{B}\bar{x} + Nx_N = b$. Exploiting the upper triangularity of $\bar{B}$ results in Eqs. (13)-(15). □

Eq. (13) is the solution to the system $A\bar{x}_A = b - Nx_N$. This is an easy flow computation problem encountered in the network simplex algorithm for the minimum cost network flow problem. The right hand side represents the remaining supply/demands at all nodes after the nonbasic arc flows are fixed at their upper bounds. When the flow of an arc *(i,j)* in *N* is set to or u_{ij}, we update $b(i):=b(i)-u_{ij}$, and $b(j):=b(j)+u_{ij}$, and Eq. (13) reflects this. The rest of the arc flows, i.e. the flows of the arcs on the spanning tree ($\bar{x}_A$) can be easily calculated by picking a leaf node at a time, determining the flow on the incident arc as the supply/demand of the leaf node (or the negative of it, depending on the orientation of the arc), and updating the supply/demand of the incident node by the amount of flow assigned to the arc, and then removing the leaf node from the set of unexamined nodes, until all nodes are examined. It is possible to determine $\bar{x}_A$ in *O(m)* time this way. We will not explain this procedure in detail, as it is well known in the literature (See, for example, Ahuja et al. [1], page 414).

Equation (14) determines the flow on arc *(α, β)* given the flows on arcs at their upper bounds and the flows on the spanning tree arcs. The total benefit is always *K* for any basis. The numerator determines the remaining benefit after the spanning tree arcs and nonbasic arcs at their upper bounds are assigned and use up a portion of the total benefit. As determined before, the denominator, $\bar{k}_{\alpha\beta}$ is the net benefit of flow on $\psi_{\alpha\beta}$, when traversed in the direction of *(α, β)*. Thus, equation (14) determines $x_{\alpha\beta}$ by allocating all of the remaining benefit to it. However, this will change the flows on other arcs of $\psi_{\alpha\beta}$, due to flow conservation. Equation (15) updates those arc flows on $\psi_{\alpha\beta}$ after $x_{\alpha\beta}$ is allocated. As before, $A^{-1}a_{\alpha\beta}$ represents the unique chain between *α* and *β* on the spanning tree, and its value is 1 for an arc on $\psi_{\alpha\beta}$ that is oriented opposite to *(α, β)*, -1 for an arc that is oriented in the same direction as *(α, β)*, and 0 for arcs that are not on $\psi_{\alpha\beta}$. Thus, equation (15) increments the flows on arcs that are oriented in the same direction as *(α, β)* by $x_{\alpha\beta}$, and decrements the flows on all other arcs on $\psi_{\alpha\beta}$ by the same amount. All of these computations can be carried out directly on the network, without any matrix operations. As we will see later in the paper, we will not need to compute flows from scratch, we will only need to update them as we pivot. We include this section in the paper for a complete discussion of the algorithm.

Computing Dual Variables

In this section, we describe how we can compute dual variables directly on the network, just as in the network simplex algorithm for the minimum cost network flow problem. We again exploit the upper triangularity of the transformed basis to efficiently carry our the computations.

Proposition 3 *Let the dual variable that corresponds to the minimum benefit constraint (Eq.3) be λ, and the dual variables that correspond to the mass balance equations (Eqs.2) be π. Then, λ can be calculated as the ratio of the net cost to net benefit around $\psi_{\alpha\beta}$, when traversed in the direction of (α, β). Furthermore, the dual variables π can be determined by solving the following system of equations*

$$c_{ij} - \pi_i + \pi_j - \lambda k_{ij} = 0 \quad \forall (i, j) \in A \tag{16}$$

Proof: We can solve the system $[\pi\ \lambda]\ B = c$ to obtain the values of π and λ. Post-multiplying both sides of this equation by B', we obtain: $[\pi\ \lambda]\overline{B} = [c_A\ c_{\alpha\beta}]B'$. Exploiting the upper triangularity of $\overline{B}$, we obtain:

$$\lambda = \frac{c_{\alpha\beta} - c_A A^{-1} a_{\alpha\beta}}{\bar{k}_{\alpha\beta}} \tag{17}$$

$$c_A - \pi A - \lambda k_A = 0 \tag{18}$$

Using the same arguments in the proof of Proposition 1, it can be shown that the numerator of Eq. (17) is the net cost around the fundamental cycle $\psi_{\alpha\beta}$, when traversed in the direction of (α, β). We will call it $\bar{c}_{\alpha\beta}$ for the rest of the paper. $\bar{k}_{\alpha\beta}$ is the net benefit around $\psi_{\alpha\beta}$ when traversed in the direction of (α, β), as defined in Proposition 1. Equation (18) reduces to $c_{ij} - \pi_i + \pi_j - \lambda k_{ij} = 0$ for all arcs (i,j) in the spanning tree. □

In Eq. (16), λ is determined independent of π. We can modify the arc costs c_{ij} as $c_{ij} = c_{ij} - \lambda\ k_{ij}$, and then apply the well-known procedure to compute dual variables for all nodes using Eq. (16) (See, for example, Ahuja et al. [1], page 411). This procedure starts by arbitrarily assigning zero as dual variable to one of the nodes (root node), say, node 1, and then determining the dual variables for the incident nodes using Eq. (16), eliminating the node from the set of unexamined nodes and moving on to one of the incident nodes and repeating the same calculation, and to the next one, and so on, in a depth-first fashion, until there are no unexamined nodes in the network. This way, the dual variables can be determined in *O(n)* time.

We can also determine the dual variables individually, without having to calculate all others. This will be helpful for updating the dual variables after a pivot. In a spanning tree, there is a single chain that links every pair of nodes, and all nodes are linked since the network is connected. So, there exists a single chain between node 1 and every other node in the spanning tree. Consider node $v \in V$ and let the unique chain between node 1 and node v be P. Because all arcs on P are basic arcs, $c_{ij}^{\pi\lambda} = 0$ for all $(i, j) \in P$. In order to convert P to a path from node v to node 1, reverse each reverse arc on P, while traversing P from node v to 1. Let the resulting path be P'. For the resulting path P', $\sum_{(i,j)\in P'} c_{ij} - \pi_i + \pi_j - \lambda k_{ij} = 0$, or $\pi_v = \sum_{(i,j)\in P'} (c_{ij} - \lambda k_{ij}) + \pi_1$. Since π_1 is always zero, we obtain the following:

$$\pi_v = \sum_{(i,j)\in P} \pm c_{ij}^{'} \quad \forall v \in V, \tag{19}$$

where the sign of c'_{ij} is 1 for all $(i,j) \in P$ that are oriented towards node 1, and -1 for all $(i,j) \in P$ that are oriented away from node 1.

Optimality Testing And the Entering Arc

We define the reduced cost for an arc (i,j) as follows:

$$c_{ij}^{\pi\lambda} = c_{ij} - \pi_i + \pi_j - \lambda k_{ij} = c'_{ij} - \pi_i + \pi_j \tag{20}$$

Similar to the minimum cost network flow problem, the optimality conditions for the minimum cost-benefit network flow problem can be stated as follows:

$$c_{ij}^{\pi\lambda} \geq 0 \ \textit{for every nonbasic arc } (i,j) \textit{ for which } x_{ij} = 0 \tag{21}$$

$$c_{ij}^{\pi\lambda} \leq 0 \ \textit{for every nonbasic arc } (i,j) \textit{ for which } x_{ij} = u_{ij} \tag{22}$$

If the current basis and the corresponding primal and dual solutions satisfy the optimality conditions, the algorithm will terminate. Otherwise, the algorithm selects a nonbasic arc that violates the optimality conditions and that arc enters the basis. For any eligible arc (i,j), we refer to $|c_{ij}^{\pi\lambda}|$ as its *violation*. Even though we can use any of the various pivot rules, we assume *Dantzig's pivot rule* in this paper, which is selecting an arc with the maximum violation. It is known that Dantzig's pivot rule results in the least number of pivots.

Updating The Flows And The Tree

Let the entering arc be *(α', β')*, which will form a second cycle when it is added to the spanning tree. Let this cycle be $\psi_{\alpha'\beta'}$. In order to enter *(α', β')* into the basis, its flow needs to be increased if it is currently at its lower bound, and decreased otherwise. Let the amount of flow change around *(α', β')* be θ. We will explain how to determine θ later in this section. Due to flow conservation, if we change $x_{\alpha'\beta'}$ by θ units, we have to change the flows of all arcs in $\psi_{\alpha'\beta'}$ by the same amount. Furthermore, the benefit constraint will be violated by $\bar{k}_{\alpha'\beta'}\theta$ units. This violation will have to be eliminated by changing flows on other arcs in the network. The following proposition determines all flow changes needed to enter *(α', β')* into the basis.

Proposition 4 *Given the current spanning tree A, non-tree basic arc(α, β), and the entering arc (α', β'), the flows will change as follows: (i) if $x_{\alpha'\beta'} = 0$, the flow around $\psi_{\alpha'\beta'}$ will increase by θ units, if $x_{\alpha'\beta'} = u_{\alpha'\beta'}$, decrease by θ units; (ii) the flow around $\psi_{\alpha\beta}$ will change by $(\bar{k}_{\alpha'\beta'}/\bar{k}_{\alpha\beta})\theta$ units; (iii) θ is the maximum possible value that keeps all arcs in $\psi_{\alpha\beta}$ and $\psi_{\alpha'\beta'}$ within their upper and lower bounds.*

Proof: Let y be the column corresponding to $x_{\alpha\beta}$ in the simplex tableau. Note that y represents changes in basic arc flows per unit change in $x_{\alpha'\beta'}$. The vector y can be obtained by solving the system: $By = a_{\alpha'\beta'}$. We can partition y as y_A and $y_{\alpha\beta}$, corresponding to the tree arcs and the non-tree basic arc. Defining $\bar{y}$ such that $y = B'\bar{y}$, and using the fact that $\bar{B} = BB'$, we can instead solve the following equivalent system: $\bar{B}\bar{y} = a_{\alpha'\beta'}$. Exploiting the upper triangularity of $\bar{B}$ results in:

$$y_{\alpha\beta} = \frac{k_{\alpha'\beta'} - k_A A^{-1} a_{\alpha'\beta'}}{\bar{k}_{\alpha\beta}} = \frac{\bar{k}_{\alpha'\beta'}}{\bar{k}_{\alpha\beta}} \tag{23}$$

$$y_A = A^{-1}a_{\alpha'\beta'} - A^{-1}a_{\alpha\beta}y_{\alpha\beta} \tag{24}$$

As before, $A^{-1}a_{\alpha'\beta'}$ represents the unique chain in the spanning tree from β' to α'; and $A^{-1}a_{\alpha\beta}$ represents the unique chain from β to α. $y_{\alpha\beta}$ is the change in $x_{\alpha\beta}$ per unit of change in $x_{\alpha'\beta'}$. In Eq. (24), only the arcs that are either on $\psi_{\alpha\beta}$ or $\psi_{\alpha'\beta'}$ have nonzero values, so no other arc flows will change. As it can be seen from the first term of Eq. (24), the rate of change in the flows of arcs in $\psi_{\alpha'\beta'}$ are either 1 or -1, depending on the orientation of the arc with respect to *(α', β')*; and from the second term, the rate of change in the flows of arcs in $\psi_{\alpha\beta}$ are either $\bar{k}_{\alpha'\beta'}/\bar{k}_{\alpha\beta}$ or $-\bar{k}_{\alpha'\beta'}/\bar{k}_{\alpha\beta}$, depending on the orientation of the arcs with respect to *(α, β)*. Thus, if the flow around $\psi_{\alpha'\beta'}$ changes by θ units, the flow around $\psi_{\alpha\beta}$ will change by $(\bar{k}_{\alpha'\beta'}/\bar{k}_{\alpha\beta})\theta$ units. Clearly, since increasing the flow on *(α', β')* (or decreasing if it is coming from the upper bound) improves the objective function, θ should be chosen as large as possible, constrained by the upper and lower bounds of arcs in $\psi_{\alpha\beta}$ and $\psi_{\alpha'\beta'}$. □

Proposition 4 establishes that the flows on both cycles need to be changed, but the direction and the amount of changes require careful analysis. The direction of change around $\psi_{\alpha'\beta'}$ depends on whether $x_{\alpha'\beta'}$ is at its lower or upper bound. However, the direction of change around $\psi_{\alpha\beta}$ depends on whether the flow around $\psi_{\alpha'\beta'}$ increases or decreases, and the signs of $\bar{k}_{\alpha\beta}$ and $\bar{k}_{\alpha'\beta'}$. A flow decrease on an arc can be considered as a flow increase in the opposite direction. For simplicity of exposition, we convert flow decreases around cycles to flow increases in the opposite direction. For this reason, whenever we have to decrease flow around $\psi_{\alpha\beta}$, we increase the flow on the reverse of *(α, β)* and on the other arcs in the cycle in the same direction. Similarly, we use the reverse of *(α', β')* if we have to decrease the flow around $\psi_{\alpha'\beta'}$. The cost and benefit of a reverse arc are simply the negatives of the cost and benefit of the original arc, and the capacity of the reverse arc is the flow on the original arc. In the actual implementation of the algorithm, a flow increase on a reverse arc reflects on the original arc as a flow decrease. We denote the arc used in place of *(α, β)* (either itself or its reverse), $\sigma_{\alpha\beta}$. Similarly, we denote the arc that is used in place of *(α', β')*, $\sigma_{\alpha'\beta'}$. We can determine $\sigma_{\alpha\beta}$ and $\sigma_{\alpha'\beta'}$ as follows:

$$\sigma_{\alpha'\beta'} := \begin{cases} (\alpha',\beta'), & if\ x_{\alpha'\beta'} = 0 \\ (\beta',\alpha'), & if\ x_{\alpha'\beta'} = u_{\alpha'\beta'} \end{cases} \tag{25}$$

$$\sigma_{\alpha\beta} := \begin{cases} (\alpha,\beta), & if\ x_{\alpha'\beta'} = 0\ and\ \bar{k}_{\alpha'\beta'}\bar{k}_{\alpha\beta} \le 0\ or \\ & if\ x_{\alpha'\beta'} = u_{\alpha'\beta'}\ and\ \bar{k}_{\alpha'\beta'}\bar{k}_{\alpha\beta} \ge 0 \\ (\beta,\alpha), & if\ x_{\alpha'\beta'} = 0\ and\ \bar{k}_{\alpha'\beta'}\bar{k}_{\alpha\beta} \ge 0\ or \\ & if\ x_{\alpha'\beta'} = u_{\alpha'\beta'}\ and\ \bar{k}_{\alpha'\beta'}\bar{k}_{\alpha\beta} \le 0. \end{cases} \tag{26}$$

By Proposition 4, the flows of arcs on $\psi_{\alpha\beta}$ and $\psi_{\alpha'\beta'}$ will change as in the following equations when *(α, β)* enters the basis, and all other arc flows will remain the same. In the equations that follow and in the rest of the paper, let $\rho = \bar{k}_{\alpha'\beta'}/\bar{k}_{\alpha\beta}$.

1. For all arcs *(i,j)* that are on $\psi_{\alpha'\beta'}$ but not on $\psi_{\alpha\beta}$:

$$x_{ij} := \begin{cases} x_{ij} + \theta, & if\ (i,j) \text{ is in the same direction as } \sigma_{\alpha'\beta'} \\ x_{ij} - \theta, & if\ (i,j) \text{ is opposite to } \sigma_{\alpha'\beta'} \end{cases} \tag{27}$$

2. For all arcs *(i,j)* that are on $\psi_{\alpha\beta}$ but not on $\psi_{\alpha'\beta'}$:

$$x_{ij} := \begin{cases} x_{ij} - \rho\theta, & if\ (i,j) \text{ is in the same direction as } \sigma_{\alpha\,\beta} \\ x_{ij} + \rho\theta, & if\ (i,j) \text{ is opposite to } \sigma_{\alpha\,\beta} \end{cases} \tag{28}$$

3. For all arcs *(i,j)* that are on both $\psi_{\alpha'\beta'}$ and $\psi_{\alpha\beta}$:

$$x_{ij} := x_{ij} + \mu_{ij}\theta \tag{29}$$

where

$$\mu_{ij} := \begin{cases} 1+\rho, & \text{if (i, j) is in the same direction as } \sigma_{\alpha\,\beta} \textit{ and } \sigma_{\alpha'\,\beta'} \\ -1-\rho, & \text{if (i, j) is opposite to } \sigma_{\alpha\,\beta} \textit{ and } \sigma_{\alpha'\,\beta'} \\ 1-\rho, & \text{if (i, j) is in the same direction as } \sigma_{\alpha'\,\beta'} \text{ but not as } \sigma_{\alpha\,\beta} \\ -1+\rho, & \text{if (i, j) is in the same direction as } \sigma_{\alpha\,\beta} \text{ but not as } \sigma_{\alpha'\,\beta'} \end{cases} \tag{30}$$

The amount of change, θ, will be determined as the maximum value that keeps the arcs on both cycles within their upper and lower bounds. Thus, we need to determine the maximum remaining capacity of each arc that is in the same direction as the direction of the flow increase in each cycle, and the amount of flow on each opposite arc. Then, we need to take the minimum of them as the maximum possible change. We need to take into account the fact that for every unit of change around $\psi_{\alpha'\beta'}$, the amount of change around $\psi_{\alpha\beta}$ is ρ. We also need to take into account the fact that if an arc is on both cycles, its flow will be changed by both cycle flow augmentations, so the combined effect on those arcs needs to be considered. In the following equations, θ_1 represents the maximum amount of flow that can be increased around $\psi_{\alpha'\beta'}$, in the direction of $\sigma_{\alpha'\beta'}$, without violating the flow bounds of arcs that are exclusively in $\psi_{\alpha'\beta'}$. Similarly, θ_2 represents the maximum amount of flow that can be increased around $\psi_{\alpha'\beta'}$, in the direction of $\sigma_{\alpha'\beta'}$, without violating the flow bounds of arcs that are exclusively in $\psi_{\alpha\beta}$. Finally, θ_3 represents the maximum flow increase around $\psi_{\alpha'\beta'}$, in the direction of $\sigma_{\alpha'\beta'}$, without violating the flow bounds of arcs that are both in $\psi_{\alpha\beta}$ and $\psi_{\alpha'\beta'}$. The minimum of the three gives us the maximum amount of increase around $\psi_{\alpha'\beta'}$.

$$\theta_1 = \min_{\substack{(i,j)\in\psi_{\alpha'\beta'} \\ \notin\psi_{\alpha\beta}}} \begin{cases} u_{ij} - x_{ij}, & if\ (i,j) \textit{ is in the same direction as } \sigma_{\alpha'\,\beta'} \\ x_{ij}, & if\ (i,j) \textit{ is opposite to } \sigma_{\alpha'\,\beta'} \end{cases} \tag{31}$$

$$\theta_2 = \min_{\substack{(i,j)\in\psi_{\alpha\beta} \\ \notin\psi_{\alpha'\beta'}}} \begin{cases} \dfrac{u_{ij} - x_{ij}}{\rho}, & if\ (i,j) \textit{ is in the same direction as } \sigma_{\alpha\,\beta} \\ \dfrac{x_{ij}}{\rho}, & if\ (i,j) \textit{ is opposite to } \sigma_{\alpha\,\beta} \end{cases} \tag{32}$$

$$\theta_3 = \min_{\substack{(i,j)\in\psi_{\alpha\beta} \\ \in\psi_{\alpha'\beta'}}} \begin{cases} \dfrac{u_{ij} - x_{ij}}{\mu_{ij}}, & if\ \mu_{ij} > 0 \\ \dfrac{x_{ij}}{\mu_{ij}}, & if\ \mu_{ij} < 0 \\ \infty, & if\ \mu_{ij} = 0 \end{cases} \tag{33}$$

$$\theta = \min\{\theta_1, \theta_2, \theta_3\}. \tag{34}$$

Once the flows are updated, one of the basic variables that reaches its lower or upper bound leaves the basis. If the entering arc is the same as the leaving arc (which happens if the arc moves from its upper bound to its lower bound or vice versa), the basis does not change. This means that the spanning tree does not need to be updated, either. If the leaving arc is the current non-tree basic arc, then we can designate the entering arc as the new non-tree basic arc. In this case again, the spanning tree will not be updated. Otherwise, we will need to update the spanning tree, but we will keep the non-tree basic arc the same, to avoid the change in the dual variable of the benefit constraint, λ. We explain the dual variable updates in the next section.

Updating The Dual Variables

As explained before, dual variables are determined from Eq. (16). However, we do not have to calculate the dual variables from scratch using Eq. (16) after every simplex pivot. Depending on the entering and the leaving arcs, the dual variables will be updated as follows. Let $(\hat{\alpha}, \hat{\beta})$ be the leaving arc.

1. *The leaving arc is (α’, β’):*

In this case, *(α’, β’)* simply moves from one of its bounds to the other. The basis remains the same, so the dual variables stay the same as well. No update is needed.

2. *The leaving arc is (α, β):*

In this case, *(α’, β’)* replaces *(α, β)* as the non-tree basic arc. Thus, the spanning tree remains the same, but λ, the dual variable of the benefit constraint, changes. Let the change in λ be $\Delta\lambda$. By Eq. (19), for each node $v \in V$, the change in π_v will be $\Delta\lambda \sum_{(i,j)\in P} \pm k_{ij}$, where P is the unique chain between node v and node 1. By maintaining the unique path P and the sum $\sum_{(i,j)\in P} \pm k_{ij}$ for all nodes in the network throughout the algorithm, we can easily update the dual variables when this case happens during a pivot.

3. *The leaving arc is neither (α, β) nor (α’, β’), but another tree arc:*

In this case, the spanning tree changes, but the non-tree basic arc remains the same. Thus, for all arcs $(i, j) \in E$, $c_{ij}^{'}$ remain the same. Removal of $(\hat{\alpha}, \hat{\beta})$ from the spanning tree partitions it into two parts: A_1, and A_2. One of them contains the root node, or node 1. Let A_1 contain the root node. The entering arc, *(α’, β’)* has one end-point in A_1 and another in A_2 since it brings the two sub-trees back together. It can be shown that the dual variable π_i for every node in A_1 does not change after the pivoting, and the ones in A_2 change by a constant amount: they increase by $c_{\alpha'\beta'}^{\pi\lambda}$ if α' is in A_1, and decrease by $c_{\alpha'\beta'}^{\pi\lambda}$ if α' is in A_2.

Degeneracy

A pivot does not necessarily decrease the objective function value. If this happens, such a pivot is called a *degenerate* pivot. If we encounter a degenerate pivot, then the algorithm may indefinitely cycle through a set of basic feasible solutions that represent the same extreme point, which is called *cycling*. In order to prevent cycling, we use the notion of *strongly feasible spanning trees*, developed by Cunningham [5] and Barr er al. [2].

Definition *A spanning tree is called a strongly feasible spanning tree if and only if:*

1. every tree arc with zero flow is oriented toward node 1, and every tree arc whose flow equals its capacity is oriented away from node 1
2. we can send a positive amount of flow from any node to node 1 along the tree path without violating any flow bound.

Suppose that when we are making a pivot, there is only one candidate for the leaving arc. In this case, the next spanning tree will also be strongly feasible. However, if there are multiple candidates, we need to choose the leaving arc carefully in order to maintain the strongly feasible spanning tree. We call the candidate arcs *blocking arcs*. Consider the cycle $\psi_{\alpha'\beta'}$ formed by the entering arc *(α', β')* in the spanning tree. There is a unique chain between node *α'* and node 1 in the spanning tree, same is true for node *β'*. These unique paths must join at some node, and we call this node the *apex* of $\psi_{\alpha'\beta'}$. The apex of $\psi_{\alpha\beta}$ is defined similarly. We can now describe our exit rule that maintains the strong feasibility of the spanning tree, which is a modified version of the exit rule in Cunningham[5].

Leaving Arc Rule

Choose the leaving arc as follows:

1. All blocking arcs are on $\psi_{\alpha'\beta'}$: Choose the leaving arc as the last blocking arc encountered in traversing $\psi_{\alpha'\beta'}$ along its orientation starting at the apex of $\psi_{\alpha'\beta'}$.
2. All blocking arcs are on $\psi_{\alpha\beta}$: Choose the leaving arc as the last blocking arc on $\psi_{\alpha\beta}$. Since *(α, β)* is no longer the non-tree basic arc, make *(α', β')* the non-tree basic arc. Finally, add *(α, β)* to the spanning tree if it is not the leaving arc.
3. There are blocking arcs on both cycles: Choose the leaving arc as the last blocking arc on $\psi_{\alpha'\beta'}$. Then, remove the last blocking arc on $\psi_{\alpha\beta}$ from the spanning tree and make it the non-tree basic arc, while adding *(α, β)* to the spanning tree if it is not the leaving arc.

Lemma 1 *If the leaving arc rule described above is used, the next spanning tree will also be strongly feasible.*

Proof: Our exit rule concerns two pivot cycles as opposed to Cunningham's exit rule that involves one pivot cycle, but the basic idea is the same. Same arguments can be used that are used in the single pivot cycle case. See, for instance, Martin [10], page 500, or Ahuja et al. [1], page 424.

Finding an Initial Basis

The algorithm needs to start with a feasible basis. The obvious choice is a solution with all arc costs are zero. However, in this case the excess of the benefit constraint will be nonzero, but the network simplex algorithm we propose assumes that the benefit constraint is tight. In order to force the benefit constraint to be tight, we introduce an artificial arc from *s* to *t*. We assign a reasonably high cost to the artificial arc *(s,t)*, say *M*. Then, we assign a flow of *K/M* to the artificial arc. Artificial arc *(s,t)* becomes the additional arc of the basis. All other arcs in the basis will have zero flow. We can then arbitrarily create the spanning tree of the basis. The proposed network simplex algorithm will drive the artificial arc out of the basis due to its high cost, and never re-introduce it to the basis after it becomes nonbasic. If the flow on the artificial arc is still nonzero at optimality, this indicates that the benefit constraint is redundant.

CONCLUSIONS AND FURTHER RESEARCH

We proposed a solution algorithm for the minimum cost-benefit network flow problem, which is a specialization of the network simplex algorithm. The algorithm is carried out directly on the network. In this research, we developed the theoretical framework of the algorithm, in future research, we intend to test the empirical performance of the algorithm via a set of network problems created using network creators such as NETGEN (Klingman et al. [8]) or MAFRANGEN (Çalışkan [3]) and compare it against existing general LP and network solvers.

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. S. Barr, F. Glover, and D. Klingman. The alternating path basis algorithm for the assignment problem. *Mathematical Programming Study*, 13:1–13, 1977.

[3] C. Çalışkan. A specialized network simplex algorithm for the constrained maximum flow problem. *European Journal of Operational Research*, 210:137–147, 2011.

[4] S. Chen and R. Saigal. A primal algorithm for solving a capacitated network flow problem with additional constraints. *Networks*, 7:59–79, 1977.

[5] W. H. Cunningham. A network simplex method. *Mathematical Programming*, 11:105–116, 1976.

[6] G.W. Graves and R.D. McBride. The factorization approach to large-scale linear programming. *Mathematical Programming*, 10:91–110, 1976.

[7] D. Klingman and R. Russell. Solving constrained transportation problems. *Operations Research*, 23(1):91–106, 1975.

[8] D. Klingman, A. Napier, and J. Stutz. NETGEN: A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. *Management Science*, 20(5):814–821, 1974.

[9] J.W. Mamer and R.D. McBride. A decomposition-based pricing procedure for large-scale linear programs: an application to the linear multicommodity flow problem. *Management Science*, 46(5):693–709, 2000.

[10] R. K. Martin. *Large Scale Linear and Integer Optimization: A Unified Approach*. Kluwer Academic Publishers, Norwell, MA, 1999.