An Algorithm to Solve the Proportional Network Flow Problem

David R. Morrison, Jason J. Sauppe, Sheldon H. Jacobson

April 1, 2013

Abstract

The proportional network flow problem is a generalization of the equal flow problem on a generalized network in which the flow on arcs in given sets must all be proportional. This problem appears in several natural contexts, including processing networks and manufacturing networks. This paper describes a transformation on the underlying network that reduces the problem to the equal flow problem; this transformation is used to show that algorithms that solve the equal flow problem can be directly applied to the proportional network flow problem as well, with no increase in asymptotic running time. Additionally, computational results are presented for the proportional network flow problem demonstrating equivalent performance to the same algorithm for the equal flow problem.

1 Introduction

The equal flow problem, which was first introduced by Ali et al. (1988), is a widely-studied variant of the standard minimum-cost network flow problem. In this problem, side constraints are added to the minimum-cost flow formulation that stipulate that the flow on given sets of arcs (called **equal flow sets**) must all be equal. Ahuja et al. (1999) and Calvete (2003) derive network-simplex-based algorithms that are able to solve the equal flow problem on pure networks with good worst-case time guarantees. Morrison et al. (2013) provide an extension of this algorithm called `GenNetEq` that is able to solve the equal flow problem on a generalized network–that is, a network with multiplicative arc factors that increase or decrease the amount of flow traveling over the arcs (Ahuja et al., 1993). Though `GenNetEq` has slightly worse complexity bounds than the algorithm of Calvete (2003), it improves upon the per-iteration complexity of the standard simplex method.

The **proportional network flow problem** (referred to herein as the proportional flow problem) is a further generalization of the equal flow problem on a generalized network that relaxes the equal flow constraints. Instead of requiring the flow on all arcs in an equal flow set to be equal, the proportional flow problem requires the flow on arcs in the set (now called a **proportional flow set**) to be proportional to each other (with the proportions specified in the problem input).

The proportional flow problem has a number of applications: processing networks, which were first introduced by Koene (1982), and further studied by Chang et al. (1989), are a specific type of proportional flow problem in which all arcs in a proportional flow set originate from a single node. In particular, processing networks are useful in the study of energy and pollution models (Chinneck, 1995) and forestry management (Chinneck and Moll, 1995). Additionally, Bahçeci and Feyzioğlu (2012) describe a network simplex algorithm for the proportional flow problem on a network that is comprised of a number of disconnected subnetworks that arises in a supply and manufacturing network.

The main contribution of this paper is to show that the proportional flow and equal flow problems are in fact equivalent problems from both a theoretical and computational perspective. A modification of the graph structure for the proportional flow problem is shown to transform it into an instance of the equal flow problem; this transformation is then used to show how `GenNetEq` can be used to directly solve the proportional flow problem with no increase in running time. Computational results are presented to show this equivalence from a practical standpoint, as well.

In the following section, the proportional flow problem on a generalized network is formally defined; Section 3 gives a brief overview of the `GenNetEq` algorithm, and Section 4 describes a graph modification that allows `GenNetEq` to solve the problem. Based on this modification, Section 5 shows how to apply `GenNetEq` directly to an instance of the proportional flow problem without modifying the graph. Section 6 presents computational results for an implementation of `GenNetEq` on the proportional flow problem. Finally, Section 7 gives some conclusions and future research directions.

2 The Proportional Flow Problem

The **proportional flow problem** is defined on a directed graph $G = (N, A)$ where N is the set of nodes and A is the set of (directed) arcs in the graph. A supply vector b is given with entries for each node $i \in N$. If $b_i > 0$, node i is a source; if $b_i < 0$, node i is a sink, and if $b_i = 0$, node i is a transshipment node.

Each arc $(i, j) \in A$, has an associated cost c_{ij}, an associated capacity $u_{ij} > 0$, and an associated multiplicative factor $\mu_{ij} \geq 0$. The multiplicative factor is such that if x_{ij} units of flow are sent from node i to node j along arc (i, j), then $\mu_{ij}x_{ij}$ units of flow arrive at node j. If $\mu_{ij} > 1$, then arc (i, j) is **gainy**, and if $\mu_{ij} < 1$, then arc (i, j) is **lossy**. Otherwise, $\mu_{ij} = 1$ and arc (i, j) is **breakeven**.

Furthermore, a collection of tuples $\mathcal{P} = \{P_1, P_2, ..., P_k\}$ is given, where $P_\ell = (X_\ell, a_\ell, q_\ell)$. Each X_ℓ is a (pairwise-disjoint) subset of network arcs, with $a_\ell \in X_\ell$, and $q_\ell : X_\ell \to \mathbb{R}^+$ with $q_\ell(a_\ell) = 1$, $\ell \in \{1, 2, ..., k\}$. The arc a_ℓ is referred to as the **canonical arc** for the ℓ^{th} proportional flow set, and the function q_ℓ is the ℓ^{th} **proportion function**, which specifies the proportionality constants for the other arcs in the proportional flow set. Each proportional flow set has a cost c_ℓ representing the additional cost incurred by sending a unit of flow over a_ℓ.

The objective of the proportional flow problem on a generalized network is to find a feasible flow function $f : A \to \mathbb{R}$ of minimum cost that satisfies the demand at every node, as well as the additional proportionality property: for each $P_\ell \in \mathcal{P}$ and each arc $(i, j) \in X_\ell$, $f_{ij} = q_\ell(i, j) f_{a_\ell}$. As in the equal flow problem, the proportional flow problem on a generalized network can be modeled as a linear program. To see this, for each $i \in N$, define

$$d_\ell(i) = \sum_{(i,j)\in X_\ell} q_\ell(i, j) - \sum_{(j,i)\in X_\ell} q_\ell(j, i)\mu_{ji}$$

This quantity represents the net change in flow at node i if a single additional unit of flow travels along the canonical arc for proportional flow set P_ℓ. Also define u_ℓ and c_ℓ as the capacity function and cost function for P_ℓ, respectively. Namely $u_\ell = \min_{(i,j)\in X_\ell} u_{ij}$, and c_ℓ is the sum of the costs of the individual arcs in P_ℓ. Lastly, define A' to be the set of arcs not contained in any proportional flow set, x_{ij} to be the flow over an arc $(i, j) \in A'$, and x_ℓ to be the flow over the canonical arc for equal flow set P_ℓ. Then, the linear programming formulation for the proportional flow problem is given by

$$\begin{aligned} \text{minimize} \quad & \sum_{(i,j)\in A'} c_{ij}x_{ij} + \sum_{\ell=1}^{k} c_\ell x_\ell \\ \text{subject to} \quad & \sum_{j:(i,j)\in A'} x_{ij} - \sum_{j:(j,i)\in A'} \mu_{ji}x_{ji} + \sum_{\ell=1}^{k} d_\ell(i)x_\ell = b_i, \ \forall\ i \in N \\ & 0 \le x_{ij} \le u_{ij}, \ \forall\ (i, j) \in A' \\ & 0 \le x_\ell \le u_\ell, \ \forall\ \ell = 1, 2, ..., k \end{aligned} \tag{1}$$

Note that the equal flow problem arises as a special case of the proportional flow problem by setting $q_\ell(i,j) = 1$ for all ℓ and all $(i,j) \in X_\ell$. In this case, the collection of equal flow sets is denoted by $\mathcal{R} = \{R_1, R_2, ..., R_k\}$, where $R_\ell \subseteq A$ for each ℓ. Additionally, note that (1) is polynomially-sized in the input, and hence, can be solved efficiently using standard linear programming techniques. The primary contribution of this paper is to introduce a network-simplex-based algorithm that is able to operate directly on the underlying graph structure, and thus perform better in the worst case than the standard simplex algorithm. In particular, since the structure of the LP for the proportional flow problem and the equal flow problem are so similar, it is desirable to be able to apply `GenNetEq` to the proportional flow problem.

3 The `GenNetEq` Algorithm

The `GenNetEq` algorithm from Morrison et al. (2013) is a network-simplex-based algorithm that maintains a basis structure called a **$\mathcal{S}$-augmenting forest**, where $\mathcal{S} \subseteq \mathcal{R}$. This basis structure consists of $t \geq 0$ **type I trees** (i.e., a tree with an associated **extra arc** that forms a cycle in the tree) and $|\mathcal{S}|$ **type II trees** (i.e., trees with no extra arc), whose union collectively spans G.

Given an $\mathcal{S}$-augmenting forest F for G, define C to be the submatrix of the constraint matrix of (1) consisting of rows and columns corresponding to nodes and arcs in the type II trees of F, together with columns corresponding to the equal flow sets (since `GenNetEq` operates on instances of the equal flow problem, all proportionality constants are 1). Then F is **good** if C has full rank.

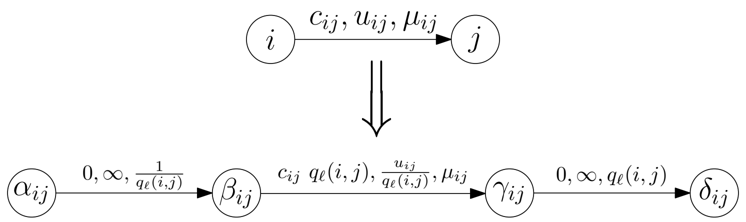
The `GenNetEq` algorithm moves from one good $\mathcal{S}$-augmenting forest to another via **pivots**; a pivot finds one arc or equal flow set that violates the optimality conditions given by the reduced arc costs (and thus whose inclusion may improve the objective function value). To maintain the basis structure, the pivot then removes an arc or equal flow set from the basis that is saturated first when pushing flow over the entering variable.

At each iteration of the algorithm, `GenNetEq` calls a subroutine called `ComputeFlows`, which updates the flow values on the arcs and equal flow sets in the graph. To do this, the flows on the type II trees are computed in terms of the (unknown) equal flow set variables. Then, since C has full rank, the flows on the equal flow sets can be computed using the system of equations at the root of each type II tree. Finally, the flows on the type I trees are computed using the standard algorithm for generalized networks (Ahuja et al., 1993). The `ComputeFlows` procedure runs in $O(m' + np + p^3)$ time, and is the bottleneck procedure for `GenNetEq`.

4 A Graph Transformation

The presence of the multiplier function μ allows for a simple graph transformation to be performed to encode the proportionality function q_ℓ for each proportional flow set in the actual graph structure itself; once this reduction is performed, `GenNetEq` can be run on the resulting network to solve the problem.

The transformation is as follows: for each proportional flow set $P_\ell \in \mathcal{P}$, replace each non-canonical arc $(i, j) \in X_\ell$ with a directed path containing nodes $\alpha_{ij}, \beta_{ij}, \gamma_{ij}$, and δ_{ij}, and three arcs $(\alpha_{ij}, \beta_{ij}), (\beta_{ij}, \gamma_{ij}), (\gamma_{ij}, \delta_{ij})$ (see Figure 1; when the meaning is clear from context, the subscripts on these nodes is dropped). Set $\mu_{\alpha\beta} = 1/q_\ell(i, j)$, $\mu_{\beta\gamma} = \mu_{ij}$, and $\mu_{\gamma\delta} = q_\ell(i, j)$. Furthermore, set $c_{\alpha\beta} = c_{\gamma\delta} = 0$, $c_{\beta\gamma} = q_\ell(i, j) \cdot c_{ij}$, $u_{\alpha\beta} = u_{\gamma\delta} = \infty$ and $u_{\beta\gamma} = u_{ij}/q_\ell(i, j)$. Finally, set $b_\beta = b_\gamma = 0$, and call the resulting network G'. Construct equal flow sets $\mathcal{R} = \{R_1, R_2, ..., R_k\}$ such that $R_\ell = \{(\beta_{ij}, \gamma_{ij}) \mid (i, j) \in X_\ell, (i, j) \neq a_\ell\} \cup \{a_\ell\}$.



c_{ij}, u_{ij}, μ_{ij}
i
j
α_{ij}
$0, \infty, \frac{1}{q_\ell(i,j)}$
β_{ij}
$c_{ij}\ q_\ell(i,j), \frac{u_{ij}}{q_\ell(i,j)}, \mu_{ij}$
γ_{ij}
$0, \infty, q_\ell(i,j)$
δ_{ij}

Figure 1: A proportional flow arc $ij \in G$ gets replaced by a path $\alpha_{ij}\beta_{ij}\gamma_{ij}\delta_{ij}$ in G, where $\beta_{ij}\gamma_{ij}$ is an equal flow arc. The values shown above each arc are the cost, capacity, and multiplicative factor, respectively.

The network G' together with the collection of equal flow sets $\mathcal{R}$ is a valid instance of the equal flow problem, and thus algorithm `GenNetEq` can be applied to compute an optimal solution or determine that the problem is infeasible. Once such a solution f' has been determined, it can be transformed to the original problem by letting $f_{ij} = f'_{ij}$ for all arcs (i, j) not in a proportional flow set and for all canonical arcs belonging to proportional flow sets. Additionally, for any other arc $(i, j) \in X_\ell$, let the flow f_{ij} equal $q_\ell(i, j) \cdot f'_{\beta\gamma}$. The following theorem establishes the correctness of this transformation:

Proposition 1. *A solution to the proportional flow problem $(G, \mathcal{P})$ is feasible if and only if the corresponding solution to the equal flow problem $(G', \mathcal{R})$ is feasible. Furthermore, the cost of the two solutions are equal.*

Proof. Let f' be a feasible solution to the equal flow problem on $(G', \mathcal{R})$. Let $j \in N$ be a node such that j is the head of some path $\alpha\beta\gamma\delta$ corresponding to a proportional flow arc (i, j). By flow balance at node γ, the flow into j along this path must be

$$\mu_{\beta\gamma} \cdot f'_{\beta\gamma} \cdot q_\ell(i,j) = \mu_{ij} \cdot f'_{a_\ell} \cdot q_\ell(i,j) = \mu_{ij} \cdot f_{a_\ell} \cdot q_\ell(i,j) = \mu_{ij} f_{ij}$$

Thus, replacing this path with a single arc (i, j) carrying flow $q_\ell(i,j) \cdot f_{a_\ell}$ maintains flow balance at node j, and since the flow along $\beta\gamma$ is at most $u_{ij}/q_\ell(i,j)$, the flow along arc (i, j) is at most u_{ij}, and no capacities are violated.

Similarly, if i is the tail of some path $\alpha\beta\gamma\delta$ corresponding to a proportional flow arc (i, j), by flow balance at node β, then the flow leaving i along this path must be $f'_{\beta\gamma} \cdot q_\ell(i,j)$, which is equal to f_{ij}. Thus replacing this path with the proportional flow arc does not destroy flow balance at node i. Finally, note again that the capacities along arc (i, j) are not violated.

An identical proof holds for the reverse direction, starting with a feasible solution to the proportional flow problem on $(G, \mathcal{P})$. Thus, the only remaining point to address is that the cost to send a unit of flow over any path $\alpha\beta\gamma\delta$ is equal to $c_{\beta\gamma} = q_\ell(i,j)c_{ij}$; consequently, the cost of a solution to the equal flow problem $(G', \mathcal{R})$ is equal to the cost of the corresponding proportional flow problem $(G, \mathcal{P})$. ■

The reduction from $(G, \mathcal{P})$ to $(G', \mathcal{R})$ can be done in time that is linear in the number of arcs in equal flow sets. Note that performing the above transformation does increase the number of nodes and arcs in the network that is solved by `GenNetEq`. While this blow-up is only a constant factor of the number of arcs in the proportional flow sets, the number of nodes in G' may increase substantially, depending on the number of arcs contained in proportional flow sets in G. This gives a worst-case per-iteration time of $O(m' + mp + p^3)$, as compared to the $O(m' + np + p^3)$ running time for solving the equal flow problem. However, it turns out that `GenNetEq` can be applied directly to the proportional flow instance, as discussed in the following section.

5 Solving the Proportional Flow Problem

While `GenNetEq` can solve the modified network as shown in Section 4, it can be shown that the proportional flow problem can be solved directly, with no loss in worst-case running time from the equal flow problem. To see this, note that the amount of flow on arcs $\alpha\beta$ and $\gamma\delta$, as well as the node potentials at β and γ are irrelevant to the solution. Furthermore, since β and γ each have exactly two adjacent arcs, the flow on both $\alpha\beta$ and $\gamma\delta$ can be written in terms of the flow on $\beta\gamma$.

In particular, note that in the LP formulation of the equal flow problem on G', all occurrences of $x_{\alpha\beta}$ can be replaced with $q_\ell(i,j)x_{\beta\gamma}$, where (i,j) is the corresponding proportional flow arc for a path $\alpha\beta\gamma\delta$. Then, deleting the flow balance equation corresponding to node β yields a new linear program where all instances of the variable $x_{\alpha\beta}$ have been eliminated. Similarly, for some proportional flow arc (i,j), all occurrences of $x_{\gamma\delta}$ can be replaced with $\mu_{ij}x_{\beta\gamma}$, and the flow balance constraint corresponding to node γ can be dropped from the LP formulation (note that this process deletes exactly two rows and two columns from the LP constraint matrix).

Proposition 2. *An $\mathcal{S}$-augmenting forest F for a proportional flow network G is good if and only if the $\mathcal{S}$-augmenting forest F' for the derived equal flow network G' is good.*

Proof. Let D be the submatrix for F' containing rows and columns corresponding to nodes and arcs in the type II trees of F', as well as the columns corresponding to the equal flow sets. Additionally, let $\hat{A}$ be the set of arcs in G' that correspond to $\alpha\beta$ or $\gamma\delta$ arcs in the transformation from G to G'.

Consider node $i \in G \cap G'$; the flow balance constraint at i for the derived equal flow problem is

$$b_i = \sum_{j:(i,j)\in A'-\hat{A}} x_{ij} + \sum_{j:(i,j)\in\hat{A}} x_{ij} - \sum_{j:(j,i)\in A'-\hat{A}} \mu_{ji}x_{ji} - \sum_{j:(j,i)\in\hat{A}} \mu_{ji}x_{ji}$$

Note, however, that when substituting for the flows on arcs in $\hat{A}$ in terms of the flow on equal flow arcs, and observing that for $(j,i) \in \hat{A}$, we have $\mu_{ji} = p_\ell(a_i)$ for some $\ell \in 1, 2, ..., k$, the resulting constraint is precisely the flow balance constraint for node i in (1) on G. Deleting the appropriate rows and columns of D therefore yields the matrix C, and since we have deleted exactly the same number of rows and columns, C has full rank if and only if D does. ■

Proposition 2 implies that the flows on the proportional flow sets of G can be computed by the

`ComputeFlows` subroutine of `GenNetEq` if and only if the flows on the equal flow sets of G' can be computed. In other words, to compute the flow on an instance of the proportional flow problem, first the flow is solved on the type II trees in terms of the (unknown) proportional flow sets. Then, since the C matrix has full rank (by Proposition 2), the system of equations induced at the roots of the type II trees yields the flow values on the proportional flow sets, and the type I trees can be computed as before. This establishes the correctness of `GenNetEq` for proportional flow problems; it should be noted that the theorems presented in Morrison et al. (2013) also establish this correctness, but do not provide any additional insight into the problem. Finally, since `GenNetEq` can be directly applied to the network G, it has a running time of $O(m' + np + p^3)$, as shown in Morrison et al. (2013).

6 Computational Results

An implementation of `GenNetEq` was modified to solve the proportional flow problem. Computational experiments were run to determine its efficacy (the `GenNetEq` code received minor updates from Morrison et al. (2013) that result in slightly faster running times, both for the equal flow problem and the proportional flow problem). This implementation was written in C++, and was tested on a variety of problem instances generated by a modification of the EFNETGEN algorithm of Morrison et al. (2013). The experiments were run on a single core of an Intel Core i7-930 2.8 GHz quad-core processor, with 12 GB of available memory.

The EFNETGEN generator (itself based on the GNETGEN algorithm of Klingman et al. (1974) and Clark et al. (1992)) allows for placement of proportional flow arcs within the network while maintaining problem feasibility. The proportionality constants are randomly generated for these arcs in the following manner: the first arc placed for any proportional flow set P_ℓ is taken to be the canonical arc a_ℓ. The proportionality constants for the remaining arcs are chosen from the set $\{1, 1 + p_{step}, 1 + 2p_{step}, ..., p_{max}\}$, where p_{step} and p_{max} are experimental parameters.

For the experiments, networks were generated with 1200 nodes, and sizes ranging from 71 940 arcs to 647 460 arcs. Problems were generated with 10, 50, 100, 150, and 200 proportional flow sets, with varying values for p_{step} and p_{max}, as given in Table 1; for comparison with the equal flow problem, one set of instances has $p_{max} = 1$. The total number of arcs in proportional flow sets was

p_{max}	p_{step}
1	-
2	1
2	0.1
5	0.5

Table 1: Parameter values used for the proportionality constants. The first entry is the equal flow problem.

30% of the total number of arcs, and these arcs were distributed evenly among all sets. A total of 30 experiments were performed for all parameter combinations, and results were averaged across all experiments (shown in Table 2).

	10 prop. flow sets		**50 prop. flow sets**		**100 prop. flow sets**	
Arcs	**Pivots**	**Time**	**Pivots**	**Time**	**Pivots**	**Time**
71 940	9220	3.88	9230	5.09	9270	5.33
143 880	13 800	6.88	13 700	9.25	13 800	10.0
215 820	18 500	10.7	18 500	14.1	18 500	15.2
287 760	23 400	13.8	23 300	17.8	23 300	20.2
359 700	28 200	17.5	28 300	22.2	28 400	25.4
431 640	33 200	22.1	33 100	27	33 100	30.7
503 579	37 900	25.4	38 000	30.7	38 000	35.0
575 520	43 000	29.6	43 100	35.1	42 800	40.0
647 460	48 200	33.9	48 200	39.6	48 100	44.6
	150 prop. flow sets		**200 prop. flow sets**			
71 940	9230	5.55	9260	5.74		
143 880	13 700	10.5	13 800	10.9		
215 820	18 500	16.2	18 500	16.9		
287 760	23 300	21.9	23 300	22.8		
359 700	28 300	27.8	28 200	29.1		
431 640	33 100	33.6	33 000	35.9		
503 579	38 000	38.9	37 900	41.4		
575 520	43 000	44.3	43 100	47.5		
647 460	48 000	49.8	48 100	53.6		

Table 2: Computational results for `GenNetEq` on a variety of different problem instances. Times are given in total CPU seconds and averaged across all instances.

It was observed that the values of the proportionality constants have no impact at all on the running time of `GenNetEq`; thus, the values shown in Table 2 are averaged over all possible combinations of p_{step} and p_{max} parameters. In particular, these results show that the proportional flow problem and the equal flow problem can be solved for practical applications in the same amount of computation time, and thus are equivalent from this perspective as well.

In addition, computational experiments for these instances were also performed using the primal simplex solver included with CPLEX 12.3. Likewise, the values of the proportionality constants had no impact on the solution times for CPLEX; the results from CPLEX are consistent with the times reported for the equal flow problem in Morrison et al. (2013). It should be noted that, while CPLEX outperforms `GenNetEq` for these problems, CPLEX is a highly-optimized solver for linear programming problems, and thus will perform quite well on problems of this nature. In all cases, `GenNetEq` performs within an order of magnitude of CPLEX's performance; on average, CPLEX performs about 6 times faster than `GenNetEq` across all instances. However, this performance is weighted towards the smaller instances, which CPLEX is able to solve around 9 times faster than `GenNetEq`. For problems with 287 760 arcs or more, CPLEX is only about 5 times as fast as `GenNetEq`.

7 Conclusion

This paper describes a reduction for the proportional flow problem to the equal flow problem on a generalized network. By performing this reduction, algorithms such as `GenNetEq` can be used for the former problem without modification. Furthermore, this reduction is used to show that `GenNetEq` can be applied to the proportional flow problem directly, with no loss in the worst-case asymptotic running time of the algorithm. Finally, computational results are presented for an implementation of `GenNetEq` showing that it can be solved in the same amount of computation time as the equal flow problem. These experimental results were compared against the standard linear programming solver in CPLEX and found to be competitive with it.

Further research into this problem should consider the effects of specialized network structures on the algorithm; both the processing networks of Koene (1982) and the transportation supply networks of Bahçeci and Feyzioğlu (2012) have a significantly restricted network structure with regards to the topology of the proportional flow arcs. These restrictions may allow for significant improvements to be made in the solution algorithm.

Acknowledgments

The authors would like to thank one anonymous referee for comments which resulted in a significantly-improved version of this paper. All computational results were obtained with the Simulation and Optimization Laboratory at the University of Illinois, Urbana-Champaign. This research has been supported in part by the Air Force Office of Scientific Research (FA9550-10-1-0387), the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program. The third author was supported in part by (while serving at) the National Science Foundation. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, the National Science Foundation, or the United States Government.

References

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows.* Prentice Hall, New Jersey, 1993.

R. K. Ahuja, J. B. Orlin, G. M. Sechi, and P. Zuddas. Algorithms for the simple equal flow problem. *Management Sci.*, 45(10):1440–1455, Oct. 1999.

A. I. Ali, J. Kennington, and B. Shetty. The equal flow problem. *Eur. J. Oper. Res.*, 36(1):107–115, Jul. 1988.

U. Bahçeci and O. Feyzioğlu. A network simplex based algorithm for the minimum cost proportional flow problem with disconnected subnetworks. *Opt. Lett.*, 6(6):1173–1184, Aug. 2012.

H. I. Calvete. Network simplex algorithm for the general equal flow problem. *Eur. J. Oper. Res.*, 150(3):585–600, Nov. 2003.

M. D. Chang, C-H. J. Chen, and M. Engquist. An improved primal simplex variant for pure processing networks. *ACM Trans. Math. Softw.*, 15(1):64–78, Mar. 1989.

J. W. Chinneck. Processing network models of energy/environment systems. *Comput. & Indust. Engrg.*, 28(1):179–189, Jan. 1995.

J. W. Chinneck and R. H. H. Moll. Processing network models for forest management. *Omega*, 23 (5):499–510, Oct. 1995.

R. Clark, L. Kennington, R. R. Meyer, and M. Ramamurti. Generalized networks: Parallel algorithms and an empirical analysis. *ORSA J. on Comput.*, 4(2):132–145, 1992.

D. Klingman, A. Napier, and J. Stutz. NETGEN: A program for generating large scale capacitated assignment, transportation and minimum cost flow networks. *Management Sci.*, 20:814–820, 1974.

J. Koene. *Minimal cost flow in processing networks: a primal approach.* PhD thesis, Technische Universiteit Eindhoven, 1982. URL `http://alexandria.tue.nl/extra1/PRF4A/8203150.pdf`.

D. R. Morrison, J. J. Sauppe, and S. H. Jacobson. A network simplex algorithm for the equal flow problem on a generalized network. *INFORMS J. on Comput.*, 25:2–12, 2013.