Network Flow Optimization with Minimum Quantities

Hans Georg Seedig
seedigh@in.tum.de

December 1st, 2010

Economies of scale often appear as decreasing marginal costs, but may also occur as a constraint that at least a minimum quantity should be produced or nothing at all. Taking the latter into account can support decisions about keeping a production site, setting up a new one, establishing new routes in logistics network, and many more. In our work, we define the corresponding problem MINIMUM COST NETWORK FLOW WITH MINIMUM QUANTITIES and prove its computational hardness. Following that, we devise a tailored Branch-&-Bound algorithm with an efficient update step that usually affects only small parts of the network. Experiments are conducted on real problems and artificial networks. In our results, the update step made the inevitable but well-studied subproblem of the initial computation of a minimum cost network flow problem taking most of the actual total running time. As our method does also compare favorably with a heuristic algorithm that has been implemented before, we recommend it for practical use.

1 Introduction

The problem we study is derived form the classical MINIMUM COST NETWORK FLOW Problem (MCNF). Because of their immanent applicability to cost-intensive tasks, MCNF and many of its variants have been studies extensively in the past. Among these are the problems where the costs per unit amount are increasing or decreasing (convex or concave costs) and where the amount routed through a network changes on the way (generalized flow problem). Treating other than fixed costs per unit amount often reflects the real cost structure of a given real-world problem. Costs amortize with bigger amounts, for example at production sites. Having a factory building a single car makes this car a lot more expensive compared to a factory where hundreds of cars are built each day. On the other side, the marginal costs might rise for for larger numbers. If the facilities are used to capacity, an increase in output demands for the construction of new facilities, additional work force and so on. These effects are often referred to as *economy of scale*.

A new question was raised recently where closing some production facilities was considered favorably for lowering costs. While the fixed setup costs that occur as soon as anything is produced at all could not be quantified, the customer named a threshold amount of output from which on a location was considered worth being kept. This amount is what we call *Minimum Quantity* (or *MQ*).

So, the problem to solve deals only with constant marginal costs but restricts the solution space as suggestions where a location had an output level above zero but below the MQ would be considered ineligible.

2 Preliminaries

In this work, a *network* $N = (V, E, u, c)$ is a digraph (V, E) equipped with a *capacity* function $u : E \to \mathbb{R}^+ \cup \{\infty\}$ and a *cost* function $c : E \to \mathbb{R}$.[1] At the nodes, a *demand* function $b : V \to \mathbb{R} \cup \{-\infty\}$ is given. We call each node s with $b(s) > 0$ a *source*, having *supply*, and every node t with $b(t) < 0$ a *sink*. As every network with more than one source and/or more than one sink is easily transformed to an equivalent one with a single source and a single sink, we assume the latter to be the case. Let $|V| =: n$ and $|E| =: m$.

A *feasible flow* in a network N is a function $f : E \to \mathbb{R}^+$ satisfying the *capacity constraints* for each e in E: $f(e) \leq u(e)$ and flow conservation constraints

$$\sum_{e_1=(v',v)} f(e_1) - \sum_{e_2=(v,v'')} f(e_2) = b(v) \quad \forall\, v \in V. \tag{1}$$

Given a network N, we define the *flow network* $N_F(f) = (V, E^f, u^f, c^f)$ where E^f is the set of all edges from N having a positive flow value in f, the capacity u^f equals f for all these edges and the costs on these edges are the same as in N.

Similarly, the *residual network* $N_R(f) = (V, E^f_R, u^f_R)$ with costs c^f_R is the network with capacities indicating how much flow can still be sent at which price along an edge without violating the capacity bound and how much can be sent back to a node without the flow being negative while reducing the total costs.

3 Minimum Cost Network Flow

The central problem regarding network flows is that of finding a feasible flow with minimum cost that satisfies all demands. For a network with one source

[1]We will use vector notation u_e and c_e to denote the capacity and cost of edge e.

s,one sink t, and demand d, the problem is defined as follows.

Minimum Cost Network Flow (MCNF)

Input: Network $N = (V, E)$, $s, t \in V$, capacities u_e and costs c_e for all $e \in E$ and demands per node $b_t = -b_s = d, b_v = 0 \, \forall v \in V \setminus \{s, t\}$

Task: Find a feasible flow $f = (f_1, \ldots, f_m)$ that minimizes

$$\sum_{e \in E} c_e \cdot f_e.$$

Different types of algorithms have been devised for MCNF with more than one allowing a strongly polynomial running time. Today, the most popular methods (see, e.g., [7, 4]) use a scaling approach of one kind or another. An interesting optimality criterion for an optimal flow f^* is that the residual network $N_R(f^*)$ contains no cycles of total negative cost. This implies an algorithm that successively identifies and eliminates such cycles by increasing the flow along the cycle's edges. When in each step the cycle with minimum mean costs is canceled, a strongly polynomial bound can be achieved [5].

4 Minimum cost Network Flow with Minimum Quantities

We define the problem of a minimum cost network flow with minimum quantities on the first layer or MCNFMQ as a mixed integer linear program (MILP).

MCNF with Minimum Quantities (MCNFMQ)

Input: Network $N = (V, E)$ with $s, t \in V$, capacities $u_e \forall \, e \in E$, a demand $d \in \mathbb{N}$
and minimum quantity lots $\lambda_e \neq 0$ on edges $(s, v) \in E$.

Task: Find $x = (x_1, \ldots, x_m)$ and a feasible flow $f = (f_1, \ldots, f_m)$ that satisfies

$$\sum_{e=(s,u) \in E} f_e = \sum_{e=(w,t) \in E} f_e = val(f) = d,$$
$$\lambda_e x_e \leq f_e \leq x_e u_e \qquad \forall \, e \in E$$

and minimizes

$$\sum_{e \in E} c_e \cdot f_e.$$

4.1 Complexity of MCNFMQ

To show the NP-completeness of MCNFMQ, we use the known completeness of the problem SubsetSum for the reduction. SubsetSum is a variant of the Knapsack problem and is indeed NP-complete as shown by [3].

SubsetSum

Input:	$A = \{a_1, \dots, a_n\}, a_i \in \mathbb{N}$
	$k \in \mathbb{N}$
Question:	Does there exist an $I \subset \{1, \dots, n\}$ such that $\sum_{i \in I} a_i = k$?

Theorem 1. *MCNFMQ is NP-complete.*

Proof. Given an instance of SubsetSum, define an instance of MCNFMQ as follows.

- $V = \{s, t, t_0, v_1, \dots, v_n\}$ nodes
- $E \times \mathbb{N}^2 = \{(s, v_1, a_1, a_1), \dots, (s, v_n, a_n, a_n)\} \cup \{(v_1, t_0, 0, \infty), \dots$ $\dots, (v_n, t_0, 0, \infty)\} \cup \{(t_0, t, 0, k)\}$ edges with MQ lot and upper capacity.

For the network $N = (V, E)$ with MQ lots and upper capacities as defined and all $c_e = 0$, it is straightforward to check that the given instance of SubsetSum is a Yes instance if and only if there is a feasible flow in the corresponding MCNFMQ.

□

4.2 Algorithm for MCNFMQ

Because of the computational hardness of MCNFMQ, it is justified to look at possible super-polynomial algorithms. First, we introduce the notion of a *configuration* which is a tuple of decision variables, one for each MQ-constrained edge, forcing the flow on this edge to be zero or above the MQ lot. Each such tuple defines a new problem that turns out to have an MCNF equivalent. Our algorithm is a Branch-&-Bound method on this set of configurations.

The algorithm works as follows. We start with a completely relaxed MCNFMQ (which is an MCNF) and efficiently compute its solution. On edges where the solution violates the MQ constraint, we branch to obtain two new configuration. After applying Wagner's transformation [8] we have corresponding MCNFs that can be solved efficiently with the methods discussed earlier.

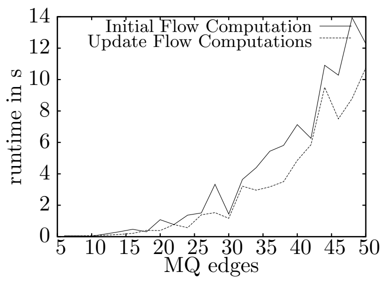
We increased the performance of our algorithm dramatically by implementing an update step, using the previously computed solution of the parent configuration for the new setting instead of computing the solution to the new MCNFs from scratch. It is a rerouting procedure, where the flow is increased (decreased) along shortest (longest) s-t-paths in the residual (flow) network to first match the new MQ constraint, followed by the elimination of cycles with total negative cost in the residual network to regain optimality, and decrease (increase) the flow again along longest (shortest) s-t-paths in the flow (residual) network to reach feasibility.

5 Experimental Results

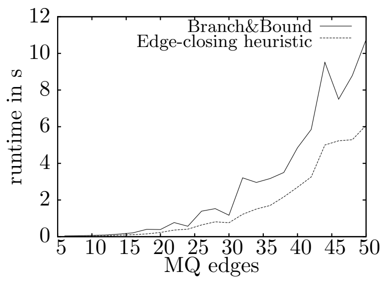
We compared our Branch-&-Bound (B&B) method with an established heuristic that has been used in the past. That method basically also starts with a relaxed solution, but then only closes edges with violated MQ-constraints and does no backtracking. As soon as a feasible solution is found, it is returned but neither its optimality nor the discovery of any existing feasible solution is guaranteed.

For our experiments, we had a data set of an international company to our avail and did also generate similarly-structured artificial networks. These are basically a fully connected graph where each of the vertices is to be thought of as a depot and is connected to the source and the sink. The MQ-constrained edges are the ones connected to the source and are all assigned the same MQ lot. Costs and capacities of all edges were randomly chosen and above the MQ lot where applicable.

Our findings are that the average optimality gap of the rather simple heuristic is about 4% for our models, tested against the B&B method that is guaranteed to compute the exact solution. The update procedure proved to be effective as on average the time needed for all updates during the exploration of the B&B tree roughly equaled the time needed for the inevitable computation of the very first relaxed solution as can be seen Figure 1(a). With this, the B&B method takes only slightly more time than the heuristic in our experiments as depicted in Figure 1(b) while usually computing solutions for about 3 times as many configurations during its execution.



(a) Execution time of B&B split into computation of initial solution and update phase



(b) Comparison of B&B and edge-closing heuristic considering execution time of the update phase

Figure 1: Execution time comparisons on randomly generated networks with two models, averaged over 30 runs.

6 Conclusions and Outlook

We introduced a new type of constraints in the context of Minimum Cost Network Flow that has obvious applications to real world problems. We were able to prove the hardness of the general MCNFMQ problem and subsequently devised a tailored Branch-&-Bound algorithm with an update method that showed to be very efficient in practice. That way, the computational complexity of real world instances was basically reduced to the well-studied classical MCNF problem for which very fast algorithms have already been implemented [4]. The same is true for the subproblem of identifying cycles of negative length in the residual network we used for the update step [1]. This makes our approach suitable for actual problems in logistics.

Possible future work is to investigate more on different cost models to account for economies of scale or other effects. There has been some work on complexity and algorithms for network flow problem with nonlinear cost functions (,e.g., [6], [2]) but we think that this is worth extending. While the resulting problems are mostly hard, it would still be interesting to devise tailored algorithms for practical use.
We would be interested in seeing our approach extended to multi-commodity problems where the constraints are coupled. It is also a remaining question wether the hardness of MCNFMQ holds if all possible minimum quantity lots are known to be the same.

Acknowledgement: The author thanks Prof. Dr. A. Taraz and Dr. Mirko Eickhoff for helpful discussions. Part of this work was done while the author was at Axxom Software AG, Munich, Germany.

References

[1] Boris Vasilievich Cherkassky and Andrew Vladislav Goldberg. Negative-cycle detection algorithms. *Mathematical Programming*, 85:277–311, 1999.

[2] Dalila Benedita Machado Martins Fontes, Eleni Hadjiconstantinou, and Nicos Christofides. A branch-and-bound algorithm for concave network flow problems. *Journal of Global Optimization*, 34(1):127–155, 2006.

[3] Michael Randolph Garey and David Stifler Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness.* W. H. Freeman, January 1979.

[4] Andrew Vladislav Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22:1–29, 1997.

[5] Andrew Vladislav Goldberg and Robert Endre Tarjan. Finding minimum-

cost circulations by canceling negative cycles. *Journal of the ACM*, 36(4):873–886, 1989.

[6] Dorit Simona Hochbaum. Complexity and algorithms for nonlinear optimization problems. *Annals of Operations Research*, 153(1):257–296, 2007.

[7] James Berger Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41(2):338–350, 1993.

[8] Harvey Maurice Wagner. On a class of capacitated transportation problems. *Management Science*, 5(3):304–318, 1959.