Control and Cybernetics

vol. **34** (2005) No. 4

A scaling out-of-kilter algorithm for minimum cost flow

by

Laura Ciupală

Faculty of Mathematics and Informatics, Transilvania University of Braşov
Iuliu Maniu 50, Romania

Abstract: The out-of-kilter algorithm is one of the basic algorithms that solve the minimum cost flow problem. Its drawback is that it can improve the objective function at each iteration by only a small value. Consequently, it runs in pseudo-polynomial time. In this paper, we describe a new out-of-kilter algorithm for minimum cost flow that runs in polynomial time. Our algorithm is a scaling algorithm and improves the objective function at each time by a "sufficiently large" value.

Keywords: network flow, minimum cost flow, shortest path, scaling technique.

1. Introduction

Let $G = (N, A)$ be a directed network, defined by a set N of n nodes and a set A of m arcs. Each arc $(x, y) \in A$ has a capacity $c(x, y)$ and a cost $b(x, y)$. We associate with each node $x \in N$ a number $v(x)$ which indicates its supply or demand depending on whether $v(x) > 0$ or $v(x) < 0$. The minimum cost flow problem is to determine the flow $f(x, y)$ on each arc $(x, y) \in A$ so as to

$$\text{minimize} \sum_{(x,y)\in A} b(x, y)f(x, y)$$

subject to

$$\sum_{y|(x,y)\in A} f(x, y) - \sum_{y|(y,x)\in A} f(y, x) = v(x) \qquad \text{for all } x \in N$$

$$0 \le f(x, y) \le c(x, y) \qquad \text{for all } (x, y) \in A.$$

A flow f satisfying the last two conditions is a feasible flow.

Let $\bar{c} = \max(\max\{v(x)|x \in N\}, \max\{c(x, y)|(x, y) \in A, c(x, y) < \infty\})$.

The residual network $G(f)$ corresponding to a flow f is defined as follows. We replace each arc $(x, y) \in A$ by two arcs (x, y) and (y, x). The arc (x, y) has

cost $b(x, y)$ and residual capacity $r(x, y) = c(x, y) - f(x, y)$ and the arc (y, x) has cost $b(y, x) = -b(x, y)$ and residual capacity $r(y, x) = f(x, y)$. The residual network consists only of arcs with positive residual capacity.

Without any loss of generality (see Ahuja, Magnanti and Orlin, 1993, for details), we shall assume that the minimum cost flow problem satisfies the following assumptions:

1. All data (cost, supply/demand and capacity) are integers.
2. The network contains no directed negative cost cycle of infinite capacity.
3. All arc costs are nonnegative.
4. The supplies/demands at the nodes satisfy the condition $\sum_{x \in N} v(x) = 0$ and the minimum cost flow problem has a feasible solution.
5. The network contains an uncapacitated directed path (i.e. each arc in the path has infinite capacity) between every pair of nodes.

We associate a real number $\pi(x)$ with each node $x \in N$. We refer to $\pi(x)$ as the potential of node x. For a given set of node potentials π, we define the reduced cost of an arc (x, y) as $b^{\pi}(x, y) = b(x, y) - \pi(x) + \pi(y)$.

The reduced costs are applicable to the residual network as well as to the original network.

THEOREM 1.1 (Ahuja, Magnanti and Orlin, 1993) *A feasible solution f^* is an optimal solution of the minimum cost flow problem if and only if some set of node potentials π satisfy the following reduced cost optimality conditions:*

$b^{\pi}(x, y) \geq 0$ *for every arc (x, y) in $G(f^*)$.*

2. The out-of-kilter algorithm

This algorithm is one of the basic algorithms for minimum cost flow problem. The name of the algorithm reflects the fact that arcs either satisfy the optimality conditions (and we say that they are in-kilter) or do not (and we say that they are out-of-kilter).

The kilter number $k(x, y)$ of an arc (x, y) in the residual network $G(f)$ is defined in the following manner:

$$k(x, y) = \begin{cases} 0 & \text{if} \quad b^{\pi}(x, y) \geq 0 \\ r(x, y) & \text{if} \quad b^{\pi}(x, y) < 0\,. \end{cases}$$

The kilter number of an arc (x, y) in the residual network is the magnitude of the change in residual capacity (or, equivalently, in flow) required to make the arc an in-kilter arc while keeping $b^{\pi}(x, y)$ fixed. The sum $K = \sum_{(x,y) \text{ in } G(f)} k(x, y)$ of all kilter numbers provides us with a measure of how far the current solution is from optimality. The smaller is the value of K, the closer the current solution is to being an optimal solution.

The out-of-kilter algorithm starts with a feasible flow f and a set of node potentials $\pi = 0$. The algorithm maintains all of in-kilter arcs as in-kilter arcs

and successively transforms the out-of-kilter arcs into in-kilter arcs by changing node potentials and by augmenting flow on appropriate directed cycles. The algorithm terminates when all arcs in the residual network become in-kilter, i.e. when the current flow is a minimum cost flow.

THEOREM 2.1 (Ahuja, Magnanti and Orlin, 1993) *The out-of-kilter algorithm runs in $O(m\bar{c}S(n,m))$ time, where $S(n,m)$ is the time required to solve a shortest path problem with nonnegative arc lengths.*

3. The scaling out-of-kilter algorithm

The out-of-kilter algorithm has the drawback that it can perform a lot of iterations that might decrease the sum K of the kilter numbers by a small amount. Our algorithm, that we will call scaling out-of-kilter algorithm, uses scaling technique and decreases K by a "sufficiently large" value at each time. Consequently, its running time is substantially better than the running time of the out-of-kilter algorithm.

The scaling out-of-kilter algorithm starts with a feasible flow f and a set of node potentials $\pi = 0$. Like the out-of-kilter algorithm, our algorithm successively transforms all the out-of-kilter arcs into in-kilter arcs, maintaining all the in-kilter arcs as in-kilter arcs. The scaling out-of-kilter algorithm terminates when all arcs in the residual network are in-kilter, consequently the current flow is a minimum cost flow.

Unlike the out-of-kilter algorithm, our algorithm performs several scaling phases for different values of a parameter Δ. We refer to a scaling phase with a specific value of Δ as a Δ-scaling phase. Initially, $\Delta = 2^{\lfloor \log \bar{c} \rfloor}$. During each Δ-scaling phase, after each operation performed by the algorithm (i.e. updating node potentials or augmenting flow) the sum of kilter numbers decreases by at least Δ units. When the residual network contains no arc whose kilter number is at least Δ, the algorithm reduces the value of Δ by a factor of 2 and repeats the same process. Eventually, $\Delta = 1$ and, at the end of 1-scaling phase, the current flow is a minimum cost flow because it satisfies the reduced cost optimality condition.

The Δ-residual network $G(f,\Delta)$ is defined as the subgraph of $G(f)$ consisting of those arcs whose residual capacity is at least Δ.

The scaling out-of-kilter algorithm is formally described as follows:

```
ALGORITHM SCALING OUT-OF-KILTER;
BEGIN
  π := 0;
  Δ := 2^⌊log c̄⌋;
  establish a feasible flow f in the network;
  define the Δ-residual network G(f, Δ) and compute the kilter numbers
    of arcs;
```

```
WHILE Δ ≥ 1 DO
BEGIN
  WHILE the Δ-residual network G(f, Δ) contains an out-of-kilter arc
  DO BEGIN
    select an out-of-kilter arc (p, q) in G(f, Δ);
    define the length of each arc (x, y) in G(f, Δ) as max{0, b^π (x, y)};
    let d(·) denote the shortest path distances from node q to all other
      nodes in G(f, Δ) − {(q, p)};
    let P denote a shortest path from node q to node p;
    FOR all nodes x in N DO
      π'(x) := π(x) − d(x);
    IF b^π'(p, q) < 0 THEN
    BEGIN
      W := P ∪ {(p, q)};
      r(W) := min{r(x, y)|(x, y) ∈ W};
      augment r(W) units of flow along W;
      update f and G(f, Δ);
    END;
    π := π';
    update the reduced costs;
  END;
  Δ := Δ/2;
END:
END.
```

The correctness of the scaling out-of-kilter algorithm follows from the correctness of the out-of-kilter algorithm, that can be found in Ahuja, Magnanti and Orlin (1993).

THEOREM 3.1 *The scaling out-of-kilter algorithm solves the minimum cost flow problem in $O(m \log \bar{c} S(n, m))$ time, where $S(n, m)$ is the time required to solve a shortest path problem with n nodes, m arcs and nonnegative arc lengths.*

Proof. At the end of a 2Δ-scaling phase, the 2Δ-residual network $G(f, 2\Delta)$ contains no out-of-kilter arcs. At the beginning of the Δ-scaling phase, the Δ-residual network $G(f, \Delta)$ might contain out-of-kilter arcs. The residual capacities of these arcs (x, y) satisfy the following inequalities: $\Delta \leq r(x, y) < 2\Delta$. Therefore, the sum of the kilter numbers of the arcs in $G(f, \Delta)$ is at most $2m\Delta$ at the beginning of the Δ-scaling phase. Evidently, at its end, the sum of the kilter numbers of the arcs in $G(f, \Delta)$ becomes zero.

During the Δ-scaling phase after each updating node potentials, the sum of the kilter numbers of the arcs in $G(f, \Delta)$ decreases by at least Δ units, either the algorithm augments the flow along the directed cycle W or not. If it does not augment the flow then the arc (p, q) selected as an out-of-kilter arc in $G(f, \Delta)$

must become an in-kilter arc after updating node potentials. Therefore, the sum of the kilter numbers of the arcs in $G(f, \Delta)$ decreases by at least Δ units. If the algorithm augments the flow along the directed cycle W, the kilter number of the out-of-kilter arc (p, q) decreases by at least Δ units because $r(W) \geq \Delta$. Since at the beginning of the Δ-scaling phase, the sum of the kilter numbers of the arcs in $G(f, \Delta)$ is at most $2m\Delta$ and it becomes zero at its end, the algorithm performs at most $O(m)$ updatings of potentials per scaling phase. Since we need to solve a shortest path problem for each updating of node potentials, the complexity of a scaling phase is $O(mS(n, m))$, where $S(n, m)$ is the time required to solve a shortest path problem with n nodes, m arcs and nonnegative arc lengths. The scaling out-of-kilter algorithm runs in $O(m \log \bar{c} S(n, m))$ time because it performs $O(\log \bar{c})$ scaling phases. ■

If we solve the shortest path problem using Fibonacci heap implementation of Dijkstra's algorithm then $S(n, m) = O(m + n \log n)$. Consequently, our scaling out-of-kilter algorithm solves the minimum cost flow problem in $O(m(m + n \log n) \log \bar{c})$ time.

4. Conclusions and remarks

In this paper, using the scaling technique, we developed a new polynomial algorithm for the minimum cost flow problem. Our algorithm runs in $O(m(m + n \log n) \log \bar{c})$ time and, unlike most of minimum cost flow algorithms, can be easily modified in order to solve a minimum cost flow problem in a network with positive lower bounds.

Scaling is a powerful technique that can be used to obtain more efficient algorithms for other classes of network flow problems, for example minimum flow problems, dynamic flow problems etc.

References

Ahuja, R., Magnanti, T. and Orlin, J.(1993) *Network Flow. Theory, Algorithms and Applications.* New Jersey, Prentice Hall.

Ciupală, L.(2004) About universal maximal dynamic flows. *Annals of University of Bucharest* **53** (1), 115-124.

Ciurea, E. and Ciupală, L.(2004) Sequential and parallel algorithms for minimum flows. *Journal of Applied Mathematics and Computing* **15** (1-2), 53-78.

Ciurea, E. and Ciupală, L. (2001) Algorithms for minimum flows. *Computer Science Journal of Moldova* **9** (3), 275-290.

Hoppe, B. and Tardos, E.(1994) Polynomial time algorithms for some evacuation Problems. *Proceedings of the Fifth Annual ACM- SIAM Symposium on Discrete Algorithms*, 433-441.

Sokkalingam, P.T., Ahuja, R. and Orlin, J.(2001) New Polynomial-Time Cycle-Canceling Algorithms for Minimum Cost Flows. http://web.mit.edu/jorlin/www/papers.html