The 7th Balkan Conference on Operational Research
"BACOR 05"
Constanta, May 2005, Romania

A C-SCALING ALGORITHM FOR THE MINIMUM FLOW PROBLEM

ELEONOR CIUREA
LAURA CIUPALĂ

Transilvania University Braşov

Abstract
We present a scaling decreasing path algorithm for the minimum flow problem, which is a network flow problem that was not treated so often as the maximum flow problem in the literature on network flow. Our algorithm always decreases flow along paths from the source node to the sink node with sufficiently large residual capacity and it runs in $O(m^2 \log \bar{c})$ *time.*

Keywords*: Network flow, network algorithms*

1. MINIMUM FLOW PROBLEM

The literature on network flow problem is extensive. Over the past 50 years researchers have made continuous improvements to algorithms for solving several classes of problems. From the late 1940s through the 1950s, researchers designed many of the fundamental algorithms for network flow, including methods for maximum flow and minimum cost flow problems. In the next three decades, there are many research contributions concerning improving the computational complexity of network flow algorithms by using enhanced data structures, techniques of scaling the problem data etc. The algorithms designed in the 1990s did not improve essentially the algorithms already known at that time. The minimum flow problem was not treated so often as the maximum flow problem in the literature on network flow.

We consider a capacitated network $G=(N,A,l,c,s,t)$ with a nonnegative capacity $c(i,j)$ and with a nonnegative lower bound $l(i,j)$ associated with each arc $(i,j)\in A$. We distinguish two special nodes in the network G: a source node s and a sink node t.

A flow is a function $f:A\rightarrow R_+$ satisfying the next conditions:

$$f(i,N)-f(N,i)=\begin{cases} v, & i=s \\ 0, & i\neq s,t \\ -v, & i=t \end{cases} \qquad (1.1.a)$$

$$l(i,j)\leq f(i,j)\leq c(i,j)\ \forall (i,j)\in A \qquad (1.1.b)$$

for some $v\geq 0$, where

$$f(i,N)=\sum_{j|(i,j)\in A} f(i,j)$$

and

$$f(N,i)=\sum_{j|(j,i)\in A} f(j,i)$$

We refer to v as the *value* of the flow f.

The minimum flow problem is to determine a flow f for which v is minimized.

For the minimum flow problem, the *residual capacity* $r(i,j)$ of any arc $(i,j)\in A$, with respect to a given flow f, is given by $r(i,j)=c(j,i)-f(j,i)+f(i,j)-l(i,j)$. By convention, if $(j,i)\notin A$ then we add arc (j,i) to the set of arcs A and we set $l(j,i)=0$ and $c(j,i)=0$. The residual capacity of the arc (i,j) represents the maximum amount of flow from the node i to node j that can be cancelled. The network $G_f=(N,A_f)$ consisting only of the arcs with positive residual capacity is referred to as the *residual network* (with respect to the flow f).

A *cut* is a partition of the node set N into two subsets S and $\overline{S}=N-S$; we represent this cut using the notation $[S,\overline{S}]$. We refer to a cut $[S,\overline{S}]$ as an *s-t cut* if $s\in S$ and $t\notin\overline{S}$. We refer to an arc (i,j) with $i\in S$ and $j\in\overline{S}$ as a *forward arc* of the cut, and an arc (i,j) with $i\in\overline{S}$ and $j\in S$ as a *backward arc* of the cut. Let $(S,\overline{S})$ denote the set of forward arcs in the cut, and let $(\overline{S},S)$ denote the set of backward arcs.

For the minimum flow problem, we define the *capacity* $c[S,\overline{S}]$ of an *s-t* cut $[S,\overline{S}]$ as the sum of the lower bounds of the forward arcs minus the sum of the capacities of the backward arcs. That is,

$$c[S,\overline{S}]=l(S,\overline{S})-c(\overline{S},S)$$

We refer to an *s-t* cut whose capacity is maximum among all *s-t* cuts as a *maximum cut*.

Let $\bar{c}=\max\{c(i,j)|(i,j)\in A\}$.

Theorem 1.1 (Min-Flow Max-Cut Theorem). If there is a feasible flow in the network, the value of the minimum flow from a source node s to a sink node t in a capacitated network with nonnegative lower bounds equals the capacity of the maximum *s-t* cut.

This theorem can be proved in a manner similar to the proof of the Max-Flow Min-Cut Theorem. For details see [1].

We refer to a path in G from the source node s to the sink node t as a *decreasing path* if it consists only of arcs with positive residual capacity. Clearly, there is an one-to-one correspondence between decreasing paths in G and directed paths from s to t in G_f.

Given a decreasing path P in G, we can decrease the current flow f in the following manner:

$$f(i,j) = \begin{cases} f(i,j) - r, & \text{if } (i,j) \text{ is a forward arc in } P \\ f(i,j) + r, & \text{if } (i,j) \text{ is a backward arc in } P \\ f(i,j), & \text{if } (i,j) \text{ is not an arc in } P \end{cases}$$

where $r=\min\{r_1,r_2\}$, $r_1=\min\{f(i,j) - l(i,j) \mid (i,j)$ is a forward arc in $P\}$, $r_2=\min\{c(i,j) - f(i,j) \mid (i,j)$ is a backward arc in $P\}$. We refer to r as the residual capacity of the decreasing path P.

Theorem 1.2 (Decreasing Path Theorem). A flow f is a minimum flow if and only if the residual network G_f contains no directed path from the source node to the sink node.

This theorem can be proved in a manner similar to the proof of the Augmenting Path Theorem. For details see [1].

Theorem 1.3 If f is a flow of value v in the network G, $[S, \bar{S}]$ is an s-t cut and f' is a flow of value v', with $v' \le v$ then $v - v' \le r[S, \bar{S}]$.

Proof. By Theorem 1.1, $v' \ge c[S, \bar{S}] = l(S, \bar{S}) - c(\bar{S}, S)$. But $v = f(S, \bar{S}) - f(\bar{S}, S)$. Consequently, $v - v' \le f(S, \bar{S}) - f(\bar{S}, S) - l(S, \bar{S}) + c(\bar{S}, S) = r[S, \bar{S}]$.

The minimum flow problem in a network can be solved in two phases:

(1) establishing a feasible flow

(2) from a given feasible flow, establish the minimum flow.

1.1 ESTABLISHING A FEASIBLE FLOW

The problem of determining a feasible flow consists in finding a function $f: A \to \mathbf{R}_+$ that satisfies conditions (1.1.a) and (1.1.b). First, we transform this problem into a circulation problem by adding an arc (t,s) of infinite capacity and zero lower bound. This arc carries the flow sent from node s to node t back to node s. Clearly, the minimum flow problem admits a feasible flow if and only if the circulation problem admits a feasible flow. Because these two problems are equivalent, we focus on finding a feasible circulation if it exists in the transformed network $\tilde{G} = (N, \tilde{A}, \tilde{l}, \tilde{c}, s, t)$, where

$\tilde{A} = A \cup \{(t,s)\}$

$\tilde{l}(i,j) = l(i,j)$, for every arc $(i,j) \in A$

$\tilde{l}(t,s) = 0$

$\tilde{c}(i,j) = c(i,j)$, for every arc $(i,j) \in A$

$\tilde{c}(t,s) = \infty$

The feasible circulation problem is to identify a flow $\tilde{f}$ satisfying these following constraints:

$$\tilde{f}(i,N) - \tilde{f}(N,i) = 0, \text{ for every node } i \in N. \quad (1.2.a)$$

$$\tilde{l}(i,j) \le \tilde{f}(i,j) \le \tilde{c}(i,j), \text{ for every arc } (i,j) \in A \quad (1.2.b)$$

By replacing $\tilde{f}(i,j) = \tilde{f}'(i,j) + \tilde{l}(i,j)$ in (1.2.a) and (1.2.b) we obtain the following transformed problem:

$$\tilde{f}'(i,N) - \tilde{f}'(N,i) = \tilde{b}(i), \text{ for every node } i \in N$$

$$0 \le \tilde{f}'(i,j) \le \tilde{c}(i,j) - \tilde{l}(i,j), \text{ for every arc } (i,j) \in A$$

with the supplies/demands $\tilde{b}(\cdot)$ at the nodes defined by

$$\tilde{b}(i) = \tilde{l}(N,i) - \tilde{l}(i,N).$$

Clearly, $\sum_{i \in N} \tilde{b}(i) = 0$. We can solve this feasible flow problem by solving a maximum flow problem defined in a transformed network. We introduce two new nodes: a source node s' and a sink node t'. For each node i with $\tilde{b}(i) > 0$ we add an arc (s',i) with capacity $\tilde{b}(i)$ and for each node i with $\tilde{b}(i) < 0$ we add an arc (i,t') with capacity $-\tilde{b}(i)$. Than we solve a maximum flow problem in this transformed network. If the maximum flow saturates all the sources and the sink arcs, then the initial problem has a feasible solution (which is the restriction of the maximum flow that saturates all the sources and sink arcs to the initial set of arcs A); otherwise it is infeasible. For details see [1].

1.2 ESTABLISHING A MINIMUM FLOW

There are two approaches for solving maximum flow problem: (1) using augmenting path algorithms and (2) using preflow-push algorithms. The algorithms of both classes can be modified in order to obtain minimum flow algorithms (see [6,7]). For the minimum flow problem there is a third approach that consists in finding a maximum flow from the sink node to the source node in the residual network (see [5,7]). In the next section we present a scaling decreasing path algorithm for the minimum flow problem.

2. CAPACITY SCALING ALGORITHM

This algorithm always decreases flow along a path with a "sufficiently large" residual capacity. To define the capacity scaling algorithm for minimum flow problem, let us introduce a parameter $\bar{r}$ and, with respect to a given flow f, define the $\bar{r}$-residual network as a network containing arcs whose residual capacity is at least $\bar{r}$. Let $G_f(\bar{r})$ denote the $\bar{r}$-residual network. Note that $G_f(1)=G_f$ and $G_f(\bar{r})$ is a subgraph of G_f.

Let us refer to a phase of the algorithm during which $\bar{r}$ remains constant as a scaling phase and a scaling phase with a specific value of $\bar{r}$ as a $\bar{r}$-scaling phase. Observe that in a $\bar{r}$-scaling phase, each decreasing path has the residual capacity at least $\bar{r}$.

The capacity scaling algorithm for the minimum flow problem is the following:

CAPACITY-SCALING ALGORITHM;
BEGIN
  let f be a feasible flow in network G;
  $\bar{r} := 2^{\lfloor \log \bar{c} \rfloor}$;
  WHILE $\bar{r} \ge 1$ DO

```
  BEGIN
    WHILE G_f(r̄) contains a directed path from the source node s to the sink node t DO
    BEGIN
      identify a decreasing path P from the source node s to the sink node t;
      g:=min{r(i,j)|(i,j)∈P};
      decrease g units of flow along P;
      update the residual network G_f(r̄);
    END
    r̄ := r̄ /2;
  END;
END.
```

Actually, the algorithm terminates with optimal residual capacities. From these residual capacities we can determine a minimum flow in several ways. For example, we can make a variable change: For all arcs (i,j), let $c'(i,j)=c(i,j)-l(i,j)$, $r'(i,j)=r(i,j)$ and $f'(i,j)=f(i,j)-l(i,j)$. The residual capacity of arc (i,j) is $r(i,j)=c(j,i)-f(j,i)+f(i,j)-l(i,j)$. Equivalently,

$$r'(i,j)=c'(j,i)-f'(j,i)+f'(i,j).$$

Similarly,

$$r'(j,i)=c'(i,j)-f'(i,j)+f'(j,i).$$

We can compute the value of f' in the following way:

$$f'(i,j)=\max(r'(i,j)-c'(j,i),0)$$

and

$$f'(j,i)=\max(r'(j,i)-c'(i,j),0).$$

Converting back into the original variables, we obtain the following expressions:

$$f(i,j) = l(i,j) + \max(r(i,j) - c(j,i) + l(j,i), 0)$$

and

$$f(j,i) = l(j,i) + \max(r(j,i) - c(i,j) + l(i,j), 0).$$

Theorem 2.1 If there is a feasible flow in the network G, then the capacity scaling algorithm determines a minimum flow in G.

Proof. The algorithm starts with $\bar{r} := 2^{\lfloor \log \bar{c} \rfloor}$ and halves its value in every scaling phase until $\bar{r}=1$. Consequently, the algorithm performs $1+\lfloor \log \bar{c} \rfloor = O(\log \bar{c})$ scaling phases. In the last scaling phase, $\bar{r}=1$, so $G_f(\bar{r})=G_f$. Thus, the algorithm terminates with a minimum flow in the network G.

Theorem 2.2 If there is a feasible flow in the network G, then the capacity scaling algorithm solves the minimum flow problem in $O(m^2 \log \bar{c})$ time.

Proof. First, we will show that the algorithm decreases the flow at most $2m$ times per scaling phase. Let f' be the flow at the end of the $\bar{r}$-scaling phase and let v' be the value of the flow f'. Furthermore, let S be the set of nodes reachable from node s in $G_{f'}(\bar{r})$. Since $G_{f'}(\bar{r})$ contains no decreasing path from the source node to the sink node, $t \notin S$. Therefore, $[S, \bar{S}]$ forms an s-t cut. The definition of S implies that the residual capacity of every arc in $[S, \bar{S}]$ is strictly less then $\bar{r}$, so the residual capacity of the cut, $r[S, \bar{S}]$ is at most $m\bar{r}$. By Theorem 1.3, $v^*-v' \le m\bar{r}$. In the next scaling phase, each decreasing path

has the residual capacity at least $\bar{r}/2$. Thus, in this scaling phase, the algorithm performs at most 2*m* decreases of the flow.

We can identify a decreasing path in $O(m)$ time and we can update the $\bar{r}$-residual network in $O(m)$ time. Thus, the complexity of a scaling phase is $O(m^2)$. Since the algorithm performs $O(\log \bar{c})$ scaling phases, it follows that it runs in $O(m^2 \log \bar{c})$ time.

BIBLIOGRAPHY

[1] Ravindra, Ahuja; Thomas, Magnanti; James, Orlin (1993), "Network Flows. Theory, algorithms and applications", Prentice Hall, Inc., Englewood Cliffs, New Jersey;

[2] Ravindra, Ahuja; Thomas, Magnanti; James, Orlin(1990), "Some Recent Advances in Network Flows", SIAM Review 33, 175-219;

[3] Ravindra, Ahuja; James, Orlin(1988), "A Fast and Simple Algorithm for the Maximum Flow Problem", Operation Research 37, 748-759;

[4] Ravindra, Ahuja; James, Orlin; Robert, Tarjan(1988), "Improved Time Bounds for the Maximum Flow Problem", SIAM Journal of Computing 18, 939-954;

[5] Laura, Ciupală; Eleonor, Ciurea(2001), "An Approach of the Minimum Flow Problem", The Fifth International Symposium of Economic Informatics, 786-790;

[6] Eleonor, Ciurea; Laura, Ciupală(2001), "Algorithms for Minimum Flows", Computer Science Journal of Moldova, 9 / 3(27), 275-290;

[7] Eleonor, Ciurea; Laura, Ciupală(2004), "Sequential and Parallel Algorithms for Minimum Flows", Journal of Applied Mathematics and Computing, 15 / 1-2, 53-75.