Bulletin of the *Transilvania* University of Braşov • Vol 5(54), No. 2 - 2012
Series III: Mathematics, Informatics, Physics, 125-130

MAXIMUM CUT WITH THE SMALLEST NUMBER OF BACKWARD ARCS

Laura CIUPALĂ[1]

Abstract

In this paper we focus on determining a maximum cut containing the smallest number of backward arcs from among all maximum cuts in a given network.

We will solve this problem by reducing it to a standard maximum cut problem in a transformed network.

2000 *Mathematics Subject Classification:* 90B10, 90C90.
Key words: network flow, maximum cut, minimum flow.

1 The maximum cut problem

The literature on network flow problem is extensive. Over the past 60 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 and maximum cut problems also. In the next decades, there are many research contributions concerning the computational complexity of network flow algorithms by using enhanced data structures, techniques of scaling the problem data. Although they have their own applications, the minimum flow problem and the maximum cut were not treated so often as the maximum flow and the minimum cut problem. There are many problems that occur in economy that can be reduced to minimum flow problems or maximum cut problems.

Let $G = (N, A)$ be a directed graph, 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 lower bound $l(x, y)$. In the directed network $G = (N, A, l, c, s, t)$, two special nodes are specified: s is the source node and t is the sink node.

Let X and Y be two subsets of the node set N. We define the set of arcs $(X, Y) = \{(x, y)|(x, y) \in A,\ x \in X, y \in Y\}$.

For any function $g : N \times N \to \mathbb{R}^+$ and for any function $h : N \to \mathbb{R}^+$ we define

$$g(X, Y) = \sum_{(X,Y)} g(x, y)$$

[1]Faculty of Mathematics and Informatics, *Transilvania* University of Braşov, Romania, e-mail: laura_ciupala@yahoo.com

and

$$h(X) = \sum_X h(x).$$

If $X = \{x\}$ or $Y = \{y\}$ then we will use $g(x, Y)$ or $g(X, y)$ instead of $g(X, Y)$.

A *flow* from the source node s to the sink node t in the directed network $G = (N, A, l, c, s, t)$ is a function $f : A \to \mathbb{R}^+$ which meets the follwing conditions:

$$f(x, N) - f(N, x) = \begin{cases} v, x = s \\ 0, x \neq s, t \\ -v, x = t \end{cases} \tag{1}$$

$$l(x, y) \leq f(x, y) \leq c(x, y), \qquad \forall (x, y) \in A. \tag{2}$$

We refer to v as the *value* of the flow f. A flow whose value is minimum is called a *minimum flow*.

For the minimum flow problem, a *preflow* is a function $f : A \to \mathbb{R}^+$ satisfying relations (2) and the next conditions:

$$f(x, N) - f(N, x) \leq 0, \qquad \forall x \in N \backslash \{s, t\}. \tag{3}$$

Let f be a preflow. We define the *deficit* of a node $x \in N$ in the following manner:

$$e(x) = f(x, N) - f(N, x)$$

Thus, for the minimum flow problem, for any preflow f, we have $e(x) \leq 0, \forall x \in N \backslash \{s, t\}$. We say that a node $x \in N \backslash \{s, t\}$ is active if $e(x) < 0$ and balanced if $e(x) = 0$. A preflow f for which $e(x) = 0, \forall x \in N \backslash \{s, t\}$ is a flow. Consequently, a flow is a particular case of preflow.

Let f be a flow from the source node s to the sink node t in the directed network $G = (N, A, l, c, s, t)$. For the minimum flow problem, the *residual capacity* of the arc (x, y) corresponding to the flow f is defined as $r(x, y) = f(x, y) - l(x, y) + c(y, x) - f(y, x)$. By convention, if an arc $(x, y) \notin A$, then we can add (x, y) to A and we will consider that $l(x, y) = 0$ and $c(x, y) = 0$.

The *residual network* $G(f) = (N, A(f))$ corresponding to flow f contains all those arcs with strictly positive residual capacity.

Let $X \subset N$ be a nonempty subset of the node set N and let $\overline{X} = N \backslash X$.

A *cut* is the following set of arcs $[X, \overline{X}] = (X, \overline{X}) \cup (\overline{X}, X)$, where $(X, \overline{X})$ is the set of forward arcs and $(\overline{X}, X)$ is the set of backward arcs of this cut.

A cut $[X, \overline{X}]$ where $s \in X$ and $t \in \overline{X}$ is named an $s - t$ *cut* .

For the minimum flow problem, we define the capacity $c[X, \overline{X}]$ of the $s - t$ cut $[X, \overline{X}]$ as the difference between the sum of the lower bounds of the forward arcs and the sum of the capacities of the backward arcs. That is, $c[X, \overline{X}] = l(X, \overline{X}) - c(\overline{X}, X)$.

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

The *residual capacity* $r[X, \overline{X}]$ of the $s - t$ cut $[X, \overline{X}]$ is defined as $r[X, \overline{X}] = r(X, \overline{X})$, where $r(X, \overline{X}) = f(X, \overline{X}) - l(X, \overline{X}) + c(\overline{X}, X) - f(\overline{X}, X)$.

Let f be a flow from the source node s to the sink node t in the directed network $G = (N, A, c, s, t)$ and let $[X, \overline{X}]$ be an $s-t$ cut, then $f[X, \overline{X}] = f(X, \overline{X}) - f(\overline{X}, X)$ is the net flow over the cut $[X, \overline{X}]$.

Theorem 1. *If f is a flow of value v from the source node s to the sink node t in the directed network $G = (N, A, l, c, s, t)$ and if $[X, \overline{X}]$ is an $s - t$ cut, then $v = f[X, \overline{X}] \geq c[X, \overline{X}]$.*

Proof. Adding relations (1) for $x \in X$, we obtain
$f(X, N) - f(N, X) = v$.
But,
$f(X, N) - f(N, X) = f(X, X \cup \overline{X}) - f(X \cup \overline{X}, X) = f(X, X) + f(X, \overline{X}) - f(X, X) - f(\overline{X}, X) = f(X, \overline{X}) - f(\overline{X}, X) = f[X, \overline{X}]$.
Thus,
$v = f[X, \overline{X}]$.
But, $f[X, \overline{X}] = f(X, \overline{X}) - f(\overline{X}, X) \geq l(X, \overline{X}) - c(\overline{X}, X) = c[X, \overline{X}]$.
Consequently, we proved that
$v = f[X, \overline{X}] \geq c[X, \overline{X}]$.

□

An important consequence of this theorem is the following:

Theorem 2. *(Min-flow max-cut theorem) If f is a flow of value v from the source node s to the sink node t in the directed network $G = (N, A, l, c, s, t)$ and if $[X, \overline{X}]$ is an $s - t$ cut such that $v = c[X, \overline{X}]$ then f is a minimum flow and $[X, \overline{X}]$ is a maximum $s - t$ cut.*

The min-flow max-cut theorem shows the relationship between minimum flows and maximum cuts. For determining a minimum flow several algorithms were developed in the past two decades. These algorithms can be divided into three classes:

1. decreasing path algorithms
2. preflow algorithms
3. minmax algorithm.

The decreasing path algorithms maintain during their executions the mass balances constraints (1) at every node of the network other than the source or the sink node. These algorithms identify decreasing path and decrease flows on these paths until the network contains no such path, which means that the flow is a minimum flow. By establishing different rules for determining the decreasing paths, one obtains different decreasing path algorithms for minimum flow (see [7], [8]).

The preflow algorithms pull flow along individual arcs. These algorithms do not satisfy the mass balances constraints (1) at intermediate stages. In fact, these algorithms permit the flow leaving a node to be less than the flow entering the node. Any preflow algorithm for the minimum flow problem proceeds by pulling flow from the neighbor nodes of the sink node, creating deficits in these nodes. The basic step in any preflow algorithm is to

select a node with deficit and to try to eliminate its deficit by pulling flow to its neighbors which are closer to the source node. Any preflow algorithm ends when all the intermediate nodes have no deficits, which means that a minimum flow was obtained. By establishing different rules for selecting the nodes with deficit, one obtains different preflow algorithms for minimum flow (see [4], [5], [7], [8]).

The minimax algorithm determines a minimum flow from the source node to the sink node by establishing a maximum flow from the sink node to the source node in the residual network (see [2], [6]).

If we have established a minimum flow in a network, it is easy to determine a maximum cut, based on the following theorem.

Theorem 3. *Let f be a flow from the source node s to the sink node t in the directed network $G = (N, A, l, c, s, t)$. Then f is a minimum flow if and only if the residual network $G(f) = (N, A(f))$ corresponding to flow f contains no directed path from s to t.*

Consequently, if a minimum flow f is determined, we can easily establish a maximum $s - t$ cut $[X, \overline{X}]$ in the following manner: X is the set of nodes reachable from s in the residual network $G(f) = (N, A(f))$ and $\overline{X} = N \backslash X$.

2 An algorithm for determining a maximum cut with the smallest number of backward arcs

In the previous section we have shown how a maximum cut can be determined. In this section we will focus on a more complex problem, which is the problem of establishing a maximum cut with the smallest number of backward arcs. We will transform this problem into a standard maximum cut problem in a transformed network.

Let $G = (N, A, l, c, s, t)$ be a directed network in which we need to establish a maximum $s - t$ cut with the smallest number of backward arcs. We construct the directed network $G' = (N, A, l', c', s, t)$, where $l'(x, y) = m \cdot l(x, y)$, $c'(x, y) = m \cdot c(x, y) + 1$, $\forall (x, y) \in A$.

Let $[X, \overline{X}]$ be an $s - t$ cut in G'. Then the capacity of this cut in G' is

$$c'[X, \overline{X}] = m \cdot l(X, \overline{X}) - m \cdot c(\overline{X}, X) - \left|(\overline{X}, X)\right|,$$

or, equivalently,

$$c'[X, \overline{X}] = m \cdot c[X, \overline{X}] - \left|(\overline{X}, X)\right|,$$

where $c[X, \overline{X}]$ is the capacity of this cut in G.

The algorithm for determining a maximum cut with the smallest number of backward arcs is the following:

MaxCut_smallest_backward_arc Algorithm;
Begin
determine the transformed network $G' = (N, A, l', c', s, t)$;
determine a maximum $s - t$ cut $[X, \overline{X}]$ in G';
$[X, \overline{X}]$ is a maximum $s - t$ cut with the smallest number of backward arcs in G;
end.

Theorem 4. *(Correctness theorem) The algorithm determines correctly a maximum $s-t$ cut with the smallest number of backward arcs in network $G = (N, A, l, c, s, t)$.*

Proof. The algorithm determines a maximum $s-t$ cut $[X, \overline{X}]$ in the transformed network $G' = (N, A, l', c', s, t)$. We will prove that $[X, \overline{X}]$ is a maximum $s-t$ cut with the smallest number of backward arcs in the directed network $G = (N, A, l, c, s, t)$ based on the fact that any maximum $s-t$ cut in G' is a maximum $s-t$ cut with the smallest number of backward arcs in G.

Let $[X, \overline{X}]$ be a maximum $s-t$ cut in G'. It follows that its capacity in G', which is

$$c'[X, \overline{X}] = m \cdot l(X, \overline{X}) - m \cdot c(\overline{X}, X) - \left|(\overline{X}, X)\right|,$$

or, equivalently,

$$c'[X, \overline{X}] = m \cdot c[X, \overline{X}] - \left|(\overline{X}, X)\right|,$$

has to be maximum.

Ad absurdum, $[X, \overline{X}]$ is not a maximum $s-t$ cut with the smallest number of backward arcs in G. This means that there is another $s-t$ cut $[Y, \overline{Y}]$ in G which is a maximum $s-t$ cut with the smallest number of backward arcs in G. Two cases are possible:

1. $c[Y, \overline{Y}] > c[X, \overline{X}]$. In this case the capacity of the $s-t$ cut $[Y, \overline{Y}]$ in G' is

   $$c'[Y, \overline{Y}] = m \cdot c[Y, \overline{Y}] - \left|(\overline{Y}, Y)\right| > m \cdot c[X, \overline{X}] - \left|(\overline{X}, X)\right| = c'[X, \overline{X}],$$

   which is a contradiction to $[X, \overline{X}]$ being a maximum $s-t$ cut in G'.

2. $c[Y, \overline{Y}] = c[X, \overline{X}]$ and $\left|(\overline{Y}, Y)\right| < \left|(\overline{X}, X)\right|$. In this case the capacity of the $s-t$ cut $[Y, \overline{Y}]$ in G' is

   $$c'[Y, \overline{Y}] = m \cdot c[Y, \overline{Y}] - \left|(\overline{Y}, Y)\right| > m \cdot c[X, \overline{X}] - \left|(\overline{X}, X)\right| = c'[X, \overline{X}],$$

   which, again, is a contradiction to $[X, \overline{X}]$ being a maximum $s-t$ cut in G'.

Consequenlty, if $[X, \overline{X}]$ is a maximum $s-t$ cut in G', then it is a maximum $s-t$ cut with the smallest number of backward arcs in G. Thus, the algorithm terminates with a maximum $s-t$ cut with the smallest number of backward arcs.

□

Theorem 5. *(Complexity theorem) The algorithm determines a maximum $s-t$ cut with the smallest number of backward arcs in network G in $M(n, m)$ time, where $M(n, m)$ is the time needed to determine a maximum $s-t$ cut in a network with n nodes and m arcs.*

Proof. The complexity of the algorithm for determining a maximum $s-t$ cut with the smallest number of backward arcs in network G is the complexity of determining a maximum $s-t$ cut in the transformed network G', which is $M(n, m)$. □

References

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

[2] Bang-Jensen, J. and Gutin, G., *Digraphs, Theory, Algorithms and Applications*, Springer-Verlag, London, 2001.

[3] Ciupală, L., *A deficit scaling algorithm for the minimum flow problem*, Sadhana **31** No. 3 (2006), 1169-1174.

[4] Ciupală, L. and Ciurea, E., *A highest-label preflow algorithm for the minimum flow problem*, Proceedings of the 11th WSEAS International Conference on Computers, Greece, July 2007, 565-569, 2007.

[5] Ciupală, L. and Ciurea, E., *An algorithm for the minimum flow problem*, The Sixth International Conference of Economic Informatics, Romania, May 2003, 167-170, 2003.

[6] Ciupală, L. and Ciurea, E., *An approach of the minimum flow problem*, The Fifth International Symposium of Economic Informatics, Romania, May 2001, 786-790, 2001.

[7] Ciurea, E. and Ciupală, L., *Sequential and parallel algorithms for minimum flows*, Journal of Applied Mathematics and Computing **15** No.1-2 (2004), 53-78.

[8] Ciurea, E. and Ciupală, L., *Algorithms for minimum flows*, Computer Science Journal of Moldova **9** No.3(27) (2001), 275-290.