Bulletin of the *Transilvania* University of Braşov • Vol 6(55), No. 1 - 2013
Series III: Mathematics, Informatics, Physics, 83-88

WAVE ALGORITHM FOR MAXIMUM FLOW IN BIPARTITE NETWORKS

Laura CIUPALĂ[1]

Abstract

In this paper, we develop a wave algorithm for determining a maximum flow in a bipartite network. This algorithm is a special implementation of the generic preflow algorithm for bipartite networks ([1]). It performs passes over active nodes. In each pass, it examines all the active nodes in nondecreasing order of their distance labels. The algorithm ends when there are no more active nodes, which means that a maximum flow was established.

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

1 Introduction

Network flow problems are a group of network optimization problems with widespread and diverse applications. The literature on network flow problems 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. In the next 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.

One of the reasons for which the maximum flow problem and that minimum cost flow problem were studied so intensively is the fact that they arise in a wide variety of situations and in several forms.

For solving a maximum flow problem there are two approaches:

1. using augmenting path algorithms
2. using preflow algorithms

All these algorithms can be modified in order to become more efficient when applied on bipartite networks.

In this paper, we develop a wave algorithm for determining a maximum flow in a bipartite network. This algorithm is a special implementation of the generic

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

preflow algorithm for bipartite networks ([1]). The wave preflow algorithm is a hybrid between the FIFO preflow algorithm and the highest-label preflow algorithm. It examines the active nodes in nonincreasing order of their distance labels and the node examination terminates when either the node excess becomes zero or the node is relabeled. The wave preflow algorithm for minimum flow runs in $O(n_1^2 n_2)$ time.

2 Notation and definition

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 nonnegative capacity $c(x, y)$. In the directed network $G = (N, A, c, s, t)$, 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

$$g(X, Y) = \sum_{(X,Y)} g(x, y) \quad \text{and} \quad h(X) = \sum_{X} h(x).$$

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 $G = (N, A, c, s, t)$ is a function $f : A \to \mathbb{R}^+$ which meets the follwing conditions:

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

$$0 \leq f(x, y) \leq c(x, y), \qquad \forall (x, y) \in A. \tag{2}$$

We refer to v as the *value* of flow f. A flow whose value is maximum is called a *maximum flow*. A *preflow* is a function $f : A \to \mathbb{R}^+$ satisfying relations (2) and the next conditions:

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

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

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

Thus, for any preflow f, we have $e(x) \geq 0, \forall x \in N \backslash \{s, t\}$. 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 $G = (N, A, c, s, t)$. The *residual capacity* of arc (x, y) corresponding to flow f is defined as $r(x, y) = c(x, y) - f(x, y) + f(y, x)$ and it is the maximum amount of additional flow that can be sent from x to y using both arcs (x, y) and (y, x). By

convention, if an arbitrary arc $(x, y) \notin A$, then we can add (x, y) to A and we will consider that $c(x, y) = 0$.

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

A network $G = (N, A)$ is called *bipartite* if its node set N can be partitioned into two subsets N_1 and N_2 , such that all arcs have one endpoint in N_1 and the other in N_2.

We consider a bipartite capacitated network $G = (N, A, c, s, t)$. We distinguish two special nodes in network G: a source node s and a sink node t. We assume without loss of generality that $s \in N_2$ and $t \in N_1$. If $s \in N_1$, then we could create a new source node $s\prime \in N_2$ and add a new arc $(s\prime, s)$ with sufficiently large capacity. If $t \in N_2$, then we could create a new sink node $t\prime \in N_1$ and add a new arc $(t, t\prime)$ with sufficiently large capacity.

Let $n = |N|$, $n_1 = |N_1|$, $n_2 = |N_2|$, $m = |A|$ and $C = max\{c(i, j)|(i, j) \in A\}$. We can assume without loss of generality that $n_1 \ll n_2$.

In the residual network G_f, the *distance function* $d : N \to \mathcal{N}$ with respect to a given preflow f is a function from the set of nodes to the nonnegative integers. We say that a distance function is *valid* if it satisfies the following conditions:

$$d(t) = 0$$

$$d(i) \leq d(j) + 1, \text{ for every arc}(i, j) \in A(f).$$

We refer to $d(i)$ as the distance label of node i.

We say that the distance labels are *exact* if, for each node $i, d(i)$ equals the length of the shortest path from node s to node i in the residual network.

We refer to an arc (i, j) from the residual network as an *admissible arc* if $d(j) = d(i) + 1$; otherwise it is *inadmissible*.

Let $G = (N, A, c, s, t)$ be a bipartite directed network, $N = N_1 \cup N_2$. Any path in network G and also in the residual network $G(f)$, that is a bipartite network also, can have at most $2n_1$ arcs. Consequently, if we set $d(s) = 2n_1 + 1$ then the residual network will never contain a directed path from the source node s to the sink node t.

Lemma 1. *[1] In the bipartite directed network $G = (N, A, c, s, t)$, for any node $i \in N$, $d(i) < 4n_1 + 1$.*

For determining a maximum flow in regular networks, several algorithms were developed in the last decades. These algorithms can be divided into two classes:

1. augmenting path algorithms
2. preflow algorithms.

The augmenting 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 augmenting paths and augment the flow along

these paths until the network contains no such path, which means that the flow is a maximum flow. By establishing different rules for determining the augmenting paths, one obtains different augmenting path algorithms for maximum flow (see[1]).

The preflow algorithms push 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 exceed the flow entering the node. Any preflow algorithm for the maximum flow problem proceeds by pushing flow from the source node s to its neighbor nodes, creating excesses in these nodes. The basic step in any preflow algorithm is to select a node with excess and to try to eliminate its excess by pushing flow to its neighbors which are closer to the sink node. Any preflow algorithm ends when all the intermediate nodes have no excess, which means that a maximum flow was obtained. By establishing different rules for selecting the nodes with excess, one obtains different preflow algorithms for maximum flow (see [1]).

Algorithms from both classes can be modified in order to determine a maximum flow in a bipartite network. In the next section we will develop a wave algorithm for determining a maximum flow in a bipartite network. This algorithm is a special implementation of the generic preflow algorithm for bipartite networks ([1]). The wave preflow algorithm is a hybrid between the FIFO preflow algorithm and the highest-label preflow algorithm. It permits only node in N_1 to become active and it examines the active nodes in nonincreasing order of their distance labels and the node examination terminates when either the node excess becomes zero or the node is relabeled. The wave preflow algorithm for maximum flow runs in $O(n_1^2 n_2)$ time.

3 Wave algorithm for maximum flow in bipartite networks

The wave algorithm for maximum flow in bipartite networks is a special implementation of the generic preflow algorithm.

The highest-label preflow algorithm (described in [1]) always examines an active node with the highest distance label. The FIFO preflow algorithm (described in [1]) examines active nodes in FIFO order. The wave algorithm, described in this paragraph, is a hybrid between these two previous preflow algorithms adapted for bipartite networks. It performs passes over active nodes. In each pass, it examines all the active nodes in nonincreasing order of their distance labels (like the highest-label preflow algorithm) and the node examination terminates when either the node excess becomes zero or the node is relabeled (like in the FIFO preflow algorithm). In order to do this, it maintains two priority queues L and L_1, both with priority d. The nodes that become active during the initialization (which are all contained in N_1) are added to L. The algorithm always selects the active node with the highest priority from L and pushes flow from it toward the sink node along a path of length 2, adding the newly active nodes in L_1. Consequently, all these nodes are from N_1. When queue L becomes empty, all active nodes from queue L_1 are moved in L. The

algorithm repeats the same process until both L and L_1 become empty (i.e., until during a pass it relabels no node). Consequently, there are no active nodes and the preflow is a flow. Moreover, it is a maximum flow.

The wave preflow algorithm for the maximum flow problem is the following:

Wave Preflow Algorithm;
Begin
let $f = 0$;
determine the residual network $G(f)$;
compute the exact distance labels d in the residual network $G(f)$;
$L = \varnothing$;
for each arc $(s, i) \in A$ **do**
begin
$f(i, t) = c(i, t)$;
if $(e(i) > 0)$ and $(i \neq t)$ **then**
add i to the rear of L;
end;
$d(s) = 2n_1 + 1$;
$L_1 = \varnothing$;
while $(L \neq \varnothing)$ and $(L_1 \neq \varnothing)$ **do**
begin
if $L = \varnothing$ **then**
begin
$L = L_1$;
$L_1 = \varnothing$;
end;
remove node i from the front of queue L;
push/relabel(i);
end
end
end.
procedure push/relabel(i);
begin
$B = false$;
repeat
if there is an admissible arc (i, j) in the residual network **then**
if there is an admissible arc (j, k) in the residual network **then**
begin
push g =min$\{e(i), r(i, j), r(k, j)\}$ units of flow along the path $i-j-k$;
update the excess of nodes i and k;
if $(k \notin L_1)$ and $(k \neq s)$ and $(k \neq t)$ **then**
add k to the rear of L_1;
end;
else $d(j)$ =min$\{d(k)|(j, k) \in A(f)\} + 1$;

```
        else begin
                d(i) =min{d(j)|(i, j) ∈ A(f)} + 1;
                B = true;
            end;
        until e(i) = 0 or B;
        if e(i) > 0 then
            add i to the rear of L1;
    end;
```

Theorem 1. *(Correctness theorem) The wave preflow algorithm computes correctly a maximum flow in the bipartite network G = (N, A, c, s, t).*

Proof. The correctness of the wave preflow algorithm follows from the correctness of the generic preflow algorithm, whose specific implementation it is. □

Theorem 2. *The wave preflow algorithm for bipartite networks performs $O(n_1^2)$ passes over active nodes.*

Proof. To determine an upper bound of the number of passes performed by the algorithm we will use the potential function $\Phi = \max\{d(i)|i \text{ is an active node}\}$. The initial value of Φ is at most $4n_1$. During an arbitrary pass over the active node, one of the following 3 cases might appear:

1. The algorithm performs at least one relabel of an active node. In this case Φ increases. The total increase in Φ caused by relabeling active nodes is, considering Lemma 1, at most $4n_1^2$.

2. The algorithm doesn't relabel any active node, but performs at least one relabel of an inactive node. In this case the value of Φ doesn't change.

3. The algorithm doesn't relabel any (active or inactive) node. In this case the value of Φ decreases by at least 2 because the excess of every active node is moved closer to the sink along paths of length 2.

Combining these 3 cases, it follows that the algorithm performs $O(n_1^2)$ passes over active nodes. □

An important consequence of this theorem is the follwing:

Theorem 3. *(Complexity theorem) The wave preflow algorithm runs in $O(n_1^2 n_2)$ time.*

References

[1] Ahuja, R., Magnanti, T., Orlin, J., *Network flow. Theory, algorithms and applications*, Prentice Hall, New Jersey, 1993.

[2] Bang-Jensen, J., Gutin, G., *Digraphs, theory, algorithms and applications*, Springer-Verlag, London, 2001.