Bulletin of the *Transilvania* University of Braşov • Vol 9(58), No. 1 - 2016
Series III: Mathematics, Informatics, Physics, 111-118

WAVE ALGORITHM FOR MAXIMUM FLOW IN SEMI-BIPARTITE NETWORKS

Laura CIUPALĂ[1]

Abstract

In this paper we develop a new algorithm for solving the maximum flow problem in semi-bipartite networks. This algorithm is a specific implementation of the generic algorithm described in [4], obtained by performing passes through the active nodes. During such a pass, all the active nodes are examined in nonincreasing order of their exact distance labels, and their excesses are, partially or totally, moved closer to the sink node. After $O(n_1^2)$ passes, all the active node excesses are moved to the sink node or back to the source node and a maximum flow is obtained.

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

1 Introduction

Among the most studied network optimization problems an impotant place is reserved to the network flow problems since the middle of the last century. A reason for this is, certainly, the fact that these problems arise when solving practical problems from very different domains.

The network flow problems on which the researchers were focused in the last six decades include the maximum flow problems. There are many algorithms, divided in two classes: augmenting path algorithms and preflow algorithms, for determining a maximum flow in a network. In the last decades, the research contributions consisted mostly in improving the computational time of the maximum flow algorithms by using enhanced data structures, techniques of scaling the problem data etc ([1], [3], [6], [7], [8], [9]).

An alternative way to improve the computational time of an algorithm is to use the particularities of the problem or, in our case, of the network ([2]). In this paper, we will focus on semi-bipartite networks, a particular case of networks. This

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

type of networks is useful when modelling several problems arising in a variety of domains.

In this paper, we develop a wave algorithm for establishing a maximum flow in a semi-bipartite network, as a special implementation of the generic preflow algorithm for semi-bipartite networks developed in [4]. The wave preflow algorithm has some common features with the FIFO preflow algorithm described in [5]. But unlike this algorithm, it examines the active nodes in nonincreasing order of their distance labels. As in the execution of the FIFO preflow algorithm, the node examination terminates when either the node excess becomes zero or the node is relabeled. The wave preflow algorithm determines a maximum flow in a semi-bipartite network in $O(n_1^2 n)$ 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

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

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 $G = (N, A, c, s, t)$ 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 the flow f. A flow whose value is maximum is 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 the arc (x, y) corresponding to the 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.

A network $G = (N, A)$ is called *semi-bipartite* if its node set N can be partitioned into two subsets N_1 and N_2, such that no arc has both its endpoints in N_2. Thus, a semi-bipartite network can contain arcs having both their endpoints in N_1. Consequently, as its name implies, the notion of semi-bipartite network is less restrictive than the notion of bipartite network.

We consider a semi-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 $s \in N_2$. 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.

Let $n = |N|$, $n_1 = |N_1|$, $n_2 = |N_2|$, $m = |A|$ and $C =$max$\{c(i, j)|(i, j) \in A\}$.

In the residual network $G(f)$, the *distance function* $d : N \rightarrow \mathbb{N}$ 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 semi-bipartite directed network, $N = N_1 \cup N_2$. Any path in the network G or in the residual network $G(f)$, that is also a semi-bipartite network, can have at most $4n_1$ arcs. Consequently, if we set $d(s) = 4n_1 + 1$ then the residual network will never contain an admissible directed path from the source node s to the sink node t.

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

When elaborating an algorithm for solving a given problem, there is always a quite difficult task consisting in establishing an equilibrium between its generality and its efficiency. An algorithm that can be applied only to a particular type of networks should be, of course, more efficient than one applicable to any network. In this paper, we will develop a wave algorithm for maximum flow in semi-bipartite networks, which are networks meeting the additional constraint that there are no arcs with both endpoints in the node set N_2, but, as their name implies, they are not so restrictive as the bipartite networks.

In the generic preflow algorithm for maximum flow in semi-bipartite networks developed in [4], the only nodes that are allowed to become active are those from the node set N_1. In order to do this, this algorithm pushes flow on individual admissible arcs having both endpoint in N_1 or along admissible paths of length two having both the starting node and the ending node in N_1. The generic preflow algorithm for maximum flow in semi-bipartite networks runs in $O(n_1^2 m)$ time. Because it is a generic algorithm, this algorithm doesn't specify any rule for selecting the active node from N_1 from which it performs a push if possible or a relabel operation otherwise. We can impose different rules for the active node selection, each of them yielding different specific implementations of the generic preflow algorithm. One of these is the algorithm described in the next section.

3 Wave preflow algorithm for maximum flow in semi-bipartite networks

The generic preflow algorithm for a maximum flow in a semi-bipartite network [4] starts by saturating all the successors of the source node s. In this way these nodes become active. The existance of an active node means that the preflow isn't a flow. So, the the main step of the generic preflow algorithm is to select, without a specified rule, an active node, say node i, and to diminuate or eliminate its excess by pushing it closer to the sink node t. This closeness is measured using the exact distance labels. Consequently, the flow is pushed only along admissible arcs and only to nodes in the node set N_1, because only these nodes are allowed to become active. If node i remains active after such a push, it isn't compulsory for the algorithm to select it again in the next iteration. This process is continuated until there are no more active nodes, which means that the preflow became a flow. Moreover, it became a maximum flow.

We can establish the following rule: if during an iteration the algorithm selects an active node, say i, and it performs a push after that the node remains active, then it is mandatory that the algorithm selects the node i in the following iteration. These succesive selections of an active node i until either it becomes inactive, either it is relabelled, constitute an *active node examination.*

The wave preflow algorithm for maximum flow in semi-bipartite networks is a specific implementation of the generic preflow algorithm described above. After saturating the outcoming arcs from the source node s and, consequently, creating excesses in the successor nodes of s, the wave algorithm performs passes over the active nodes. In each pass, the active nodes are examined in nonincreasing order of their exact distance labels. In order to do this, two priority queues L and L_1 are maintained, both of them having the priority d. The nodes that become active during saturating the outcoming arcs from the source node s are added to the queue L. While this queue isn't empty, the algorithm removes the highest priority node, say i, from it. If there is an admissible arc outgoing from i, whose head node is in N_1 or an admissible path of length two from i to a node in N_1, then the algorithm pushes flow from i and, if in this way a new node becomes active, it adds it to the queue L_1. If there isn't an admissible arc outgoing from i, whose head node is in N_1 nor an admissible path of length two from i to a node in N_1, then the node i is relabeled and added to the queue L_1. When the queue L becomes empty, a pass through the active node is completed and all the nodes from L_1 are moved to L. The algorithm repeats the same process until both queues L and L_1 are empty, which means that the preflow became a flow. Moreover, it became a maximum flow.

The wave preflow algorithm for maximum flow in semi-bipartite networks is the following:

```
**Wave Preflow Algorithm;**
**Begin**
    $f = 0$;
    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**
        $f(s, i) = c(s, i)$;
        **if** $(e(i) > 0)$ and $(i \neq t)$ **then**
                add $i$ to the queue $L$ with priority $d(i)$;
        **end;**
    **end;**
    $d(s) = 4n_1 + 1$;
    $L_1 = \varnothing$ ;
    **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 $i$ with the highest priority from $L$;
              push/relabel($i$);
        end
    end.


procedure push/relabel($i$);
begin
    $B = false$;
    repeat
    if there is an admissible arc $(i, j)$ in $G(f)$ then
        if $j \in N_1$ then begin
            push $g$ =min$\{e(i), r(i, j)\}$ units of flow on the arc $(i, j)$;
            if $(j \notin L_1)$ and $(j \neq s)$ and $(j \neq t)$ then
                add $j$ to queue $L_1$ with priority $d(j)$;
        end;
        else
            if there is an admissible arc $(j, k)$ in $G(f)$ then begin
                push $g$ =min$\{e(i), r(i, j), r(j, k)\}$ units of flow along the
                path $i - j - k$;
                if $(k \notin L_1)$ and $(k \neq s)$ and $(k \neq t)$ then
                    add $k$ to queue $L_1$ with priority $d(k)$;
            end;
            else $d(j)$ =min$\{d(k) | (j, k) \in A(f)\} + 1$;
    else begin
        $d(i)$ =min$\{d(j) | (i, j) \in A(f)\} + 1$;
        $B = true$;
    end
    until $e(i) = 0$ or $B$;
    if $e(i) > 0$ then
        add $i$ to queue $L_1$ with priority $d(i)$;
end;
```

Theorem 1. *(Correctness theorem) The wave preflow algorithm computes correctly a maximum flow in the semi-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 for maximum flow in semi-bipartite networks ([4]), whose specific implementation it is. □

Theorem 2. *The wave preflow algorithm for maximum flow in semi-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 relabelling 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 1 because the excess of every active node is moved closer to the sink along paths of length 1 or 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 following:

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

References

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

[2] Ahuja, R., Orlin, J., Stein, C., Tarjan, R., *Improved Algorithms for Bipartite Network Flow*, SIAM Journal on Computing **23** (1994), no.5, 906-933.

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

[4] Ciupală, L., *A generic preflow algorithm for maximum flow in semi-bipartite networks*, Bulletin of the *Transilvania* University of Braşov **7(56)** (2014), no. 1, 103-108.

[5] Ciupală, L., *FIFO preflow algorithm for maximum flow in semi-bipartite networks*, Bulletin of the *Transilvania* University of Braşov **8(57)** (2015), no. 1, 117-122.

[6] S. Fujishige, *A maximum flow algorithm using MA ordering*, Operation Research Letters **31(3)** (2003), 176-178.

[7] S. Fujishige, S. Isotani, *New maximum flow algorithms by MA orderings and scaling*, Journal of the Operational Research Society of Japan **46(3)** (2003), 243-250.

[8] S. Kumar, P. Gupta, *An incremental algorithm for the maximum flow problem*, Journal of Mathematical Modelling and Algorithms **2(1)** (2003), 1-16.

[9] A. Schrijver, *On the history of the transportation and maximum flow problems*, Mathematical Programming **91(3)** (2002), 437-445.