Bulletin of the *Transilvania* University of Braşov • Vol 8(57), No. 1 - 2015
Series III: Mathematics, Informatics, Physics, 117-122

FIFO PREFLOW ALGORITHM FOR MAXIMUM FLOW IN SEMI-BIPARTITE NETWORKS

Laura CIUPALĂ[1]

Abstract

In this paper, we develop a special implementation of the generic algorithm for determining a maximum flow in a semi-bipartite network introduced in [3]. This generic algorithm allows only nodes in N_1 to be active. For this reason, it performs pushes on individual arcs having both endpoints in N_1 or admissible paths of length 2 whose starting and ending nodes are both contained in N_1. Because it is a generic algorithm, it doesn't specify any rule for selecting active nodes. In this paper, we will impose the FIFO rule for the active nodes selection and we will obtain a specific implementation of the generic algorithm for maximum flow in semi-bipartite networks, which has a better running time than the generic algorithm.

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

1 Indroduction

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: the source node s and the sink node t.

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

$$h(X) = \sum_{X} h(x).$$

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

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 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. The *excess* of a node $x \in N$ is defined in the following manner:

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

Thus, for any preflow f, we have $e(x) \geq 0, \forall x \in N \backslash \{s, t\}$. A node $x \in N \backslash \{s, t\}$ is called *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, 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 the 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* $d : N \to \mathcal{N}$ 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 $d(i) = d(j) + 1$; 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 $2n_1$ arcs. Consequently, if we set $d(s) = 2n_1+1$ then the residual network will never contain an admissible directed path from the source node s to the sink node t.

Lemma 1. *[3] In the semi-bipartite directed network $G = (N, A, c, s, t)$, for any node $i \in N$, $d(i) < 4n_1 + 1$.*

When developing a new algorithm for solving a specified problem, it is quite difficult to establish a balance between the generality of the algorithm and its efficiency when apllying it on particular networks (that arise in real world). In this paper, we try to find this balance by developing an algorithm for determining maximum flow in semi-bipartite networks, which are less restrictive than the bipartite networks. The algorithm that we will develop uses the particulatities of a semi-bipartite network and, consequently, it is more efficient than the corresponding algorithm for maximum flow in regular networks.

The generic preflow algorithm (see [3]) allows only the nodes in N_1 to become active. In order to do this, it pushes flow on individual admissible arcs or along paths consisting of two admissible arcs. The generic preflow algorithm for maximum flow in semi-bipartite networks runs in $O(n_1^2 m)$ time. Being a generic algorithm, it doesn't specify any rule for selecting the active node 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.

2 FIFO preflow algorithm for maximum flow in semi-bipartite networks

If the generic preflow algorithm for maximum flow in a semi-bipartite network selects in a iteration an active node, say i, it can perform a push that leaves i

still active, but it isn't mandatory that the algorithm select the same node in the next iteration. We can impose the following rule: if in an iteration the algorithm selects an active node, say i, and performs a push after that the node remains active, then it is compulsory 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 the *active node examination*.

If we impose the restriction that the active nodes are examined in FIFO order, we obtain a specific implementation of the generic algorithm for maximum flow in semi-bipartite networks, called FIFO preflow algorithm. For an easy selection of the active nodes in FIFO order, a queue L of active nodes is maintained. The FIFO preflow algorithm for maximum flow in semi-bipartite networks is the following:

FIFO 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
$f(s, i) = c(s, i)$;
if $(e(i) > 0)$ and $(i \neq t)$ **then**
add i to the rear of the queue L;
end;
$d(s) = 4n_1 + 1$;
while $(L \neq \varnothing)$ **do**
begin
remove the node i from the front of the queue 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 $(j \notin L)$ and $(j \neq s)$ and $(j \neq t)$ **then**
add j to the rear of L;
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 ∉ L) and (k ≠ s) and (k ≠ t) then
                    add k to the rear of L;
            end;
            else d(j) =min{d(k)|(j, k) ∈ 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 L;
    end;
```

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

Proof. The correctness of the FIFO preflow algorithm for maximum flow in semi-bipartite networks is a straight consequence of the correctness of the generic preflow algorithm for maximum flow in semi-bipartite networks, proved in [3]. □

Theorem 2. *The FIFO preflow algorithm determines a maximum flow in a semi-bipartite network in $O(n_1^3)$ time.*

Proof. We prove this theorem in a manner similar to the way the running time of the FIFO preflow-push algorithm for maximum flow in regular networks is proved in [1]. First we divide the node examinations in phases. The first phase contains the node examinations of the nodes that became active when saturating the outgoing arcs from the source node s in the beginning of the algorithm. The second phase consists of the node examinations of the nodes that are active at the end of the first phase. In the third phase the nodes that are active at the end of the second phase are examinated and so on.

We will use the potential function $\Phi = \max\{d(i)|i \in L\}$, where L is the set of active nodes. The initial value of Φ is at most $2n_1$, at the end of the algorithm $\Phi = 0$ and during its execution one of the following 2 cases might appear:

1. The algorithm performs at least one relabel operation during a phase. In this case Φ might increase at most by the maximum increase in any distance label. Using Lemma 1 and the fact that only nodes in N_1 can be active, it follows that the total increase in Φ caused by all relabel operations over all the phases is at most $4n_1^2$.

2. The algorithm performs no relabel operation during a phase. Then the excesses of the nodes that were active at the beginning of the phase (and became inactive at the end of it) are moved closer to the sink, i.e. to nodes with smaller distance labels. In this case Φ decreases by at least 1 unit.

Combining these 2 cases it follows that the algorithm performs at most $4n_1^2 + 2n_1$ phases. Consequently, it runs in $O(n_1^3)$ 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.

[3] Ciupală, L., *A generic preflow algorithm for maximum flow in semi-bipartite networks*, Bulletin of the *Transilvania* University of Braşov **7(56)** (2014), 103-108.

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

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

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

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