Annals of the University of Craiova, Mathematics and Computer Science Series
Volume 43(2), 2016, Pages 200–209
ISSN: 1223-6934

The maximum flows in bipartite dynamic networks. The static approach

Camelia SCHIOPU

Abstract. In this paper we study maximum flow algorithms for bipartite dynamic networks. We resolve this problem by rephrasing into a problem in bipartite static network. In a bipartite static network several maximum flow algorithms can be substantially improved. The basic idea in this improvement is a two arcs push rule. At the end of the article we present an example.

2010 Mathematics Subject Classification. 0B10, 90C35, 05C35, 68R10.
Key words and phrases. bipartite dynamic network flow, maximum flow, bipartite static network flow.

1. Introduction

The theory of flow is one of the most important parts of Combinatorial Optimization. The static network flow models arise in a number of combinatorial applications that on the surface might not appear to be optimal flow problems at all. The issue also arises directly in problems as far reaching as machine scheduling, the assignment of computer modules to computer processor, tanker scheduling etc. [1]. However, in some applications, time is an essential ingredient [3], [4], [5]. In this case we need to use dynamic network flow model. On the other hand, the bipartite static network also arises in practical context such as baseball elimination problem, network reliability testing etc. and hence it is of interest to find fast flow algorithms for this class of networks [1], [2], [6].

The dynamic approach of maximum flows problem in bipartite dynamic networks with lower bounds zero is treated in the paper [7]. The static approach of maximum flows problem in bipartite dynamic networks with positive lower bounds is treated in the paper [8].

In this paper we present the static approach of maximum flows problem in bipartite dynamic networks with lower bounds zero. This problem has not been treated so far. Further on, in Section 2 we discuss some basic notions and results for maximum flow problem in general static networks. Section 3 deals with the maximum flow problem in general dynamic networks. In Section 4 we present algorithms for flow problems in bipartite static network and in Section 5 we discuss the maximum flow problem in bipartite dynamic networks. Section 6 deals with an example for the problem presented in Section 5.

Received June 2, 2015. Revised October 26, 2016.

2. Terminology and preliminaries

In this section we discuss some basic notations and results used throughout the paper.

Let $G = (N, A, u)$ be a general static network with the set of nodes $N = \{1, \ldots, n\}$, the set of arcs $A = \{a_1, \ldots, a_k, \ldots, a_m\}$, $a_k = (i, j)$, $i, j \in N$, the upper bound (capacity) function u, $u : A \to \mathbb{N}$ with $\mathbb{N}$ the natural number set and with 1 the source node, n the sink node.

For a given pair of subset X, Y of the set of nodes N of a network G we use the notation:

$$(X, Y) = \{(i, j) | (i, j) \in A, i \in X, j \in Y\}$$

and for a given function f on set of arcs A we use the notation:

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

A flow is a function $f : A \to \mathbb{N}$ satisfying the next conditions:

$$f(i, N) - f(N, i) = \begin{cases} v, & \text{if } i = 1 \\ 0, & \text{if } i \neq 1, n \\ -v, & \text{if } i = n \end{cases} \tag{1a}$$

$$0 \leq f(i, j) \leq u(i, j), \quad (i, j) \in A \tag{1b}$$

for some $v \geq 0$. We refer to v as the value of the flow f.

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

We further assume, without loss of generality, that if $(i, j) \in A$ then $(j, i) \in A$ (if $(j, i) \notin A$ we consider that $(j, i) \in A$ with $u(j, i) = 0$).

A preflow f is a function $f : A \to \mathbb{N}$ satisfying the next conditions:

$$f(N, i) - f(i, N) \geq 0, i \in N - \{1, n\} \tag{2a}$$

$$0 \leq f(i, j) \leq u(i, j), (i, j) \in A \tag{2b}$$

For a preflow f the excess of each node $i \in N$ is

$$e(i) = f(N, i) - f(i, N) \tag{3}$$

and if $e(i) > 0$, $i \in N - \{1, n\}$ then we say that the node i is an active node.

Given a flow (preflow) f, the residual capacity $r(i, j)$ of any arc $(i, j) \in A$ is $r(i, j) = u(i, j) - f(i, j) + f(j, i)$. The residual network with respect to the flow (preflow) f is $\tilde{G} = (N, \tilde{A}, r)$ with $\tilde{A} = \{(i, j) | (i, j) \in A, r(i, j) > 0\}$. In the residual network $\tilde{G} = (N, \tilde{A}, r)$ we define the distance function $d : N \to \mathbb{N}$. We say that a distance function is valid if it satisfies the following two conditions:

$$d(n) = 0 \tag{4a}$$

$$d(i) \leq d(j) + 1, (i, j) \in \tilde{A} \tag{4b}$$

We refer to $d(i)$ as the distance label of node i. We say that an arc $(i, j) \in \tilde{A}$ is admissible if satisfies the condition that $d(i) = d(j) + 1$; we refer to all other arcs as inadmissible. We also refer to a path from node 1 to node t consisting entirely of admissible arcs as an admissible path.

In the next presentation we assume familiarity with maximum flow algorithms, and we omit many details. The reader interested in further details is urged to consult the book [1].

3. Maximum flows in dynamic networks

Dynamic network models arise in many problem settings, including production distribution systems, economic planning, energy systems, traffic systems, and building evacuation systems.

Let $G = (N, A, u)$ be a static network with the set of nodes $N = \{1, \dots, n\}$, the set of arcs $A = \{a_1, \dots, a_m\}$, the upper bound (capacity) function u, 1 the source node and n the sink node. Let $\mathbb{N}$ be the natural number set and let $H = \{0, 1, \dots, T\}$ be the set of periods, where T is a finite time horizon, $T \in \mathbb{N}$. We use state the transit time function $h : A \times H \to \mathbb{N}$ and the time upper bound function $q : A \times H \to \mathbb{N}$. The parameter $h(i, j; t)$ is the transit time needed to traverse an arc (i, j). The parameter $q(i, j; t)$ represents the maximum amount of flow that can travel over arc (i, j) when the flow departs from node i at time t and arrives at node j at time $\theta = t + h(i, j; t)$.

The maximal dynamic flow problem for T time periods is to determine a flow function $g : A \times H \to \mathbb{N}$, which should satisfy the following conditions in the dynamic network $D = (N, A, h, q)$:

$$\sum_{t=0}^{T}(g(1, N; t) - \sum_{\tau} g(N, 1; \tau)) = w \tag{5a}$$

$$g(i, N; t) - \sum_{\tau} g(N, i; \tau) = 0, i \neq 1, n, \ t \in H \tag{5b}$$

$$\sum_{t=0}^{T}(g(n, N; t) - \sum_{\tau} g(N, n; \tau)) = -w \tag{5c}$$

$$0 \leq g(i, j; t) \leq q(i, j; t), \quad (i, j) \in A \quad , \quad t \in H \tag{6}$$

$$max \quad w, \tag{7}$$

where $\tau = t - h(k, i; \tau)$, $w = \sum_{t=0}^{T} v(t)$, $v(t)$ is the flow value at time t and $g(i, j; t) = 0$ for all $t \in \{T - h(i, j; t) + 1, \dots, T\}$.

Obviously, the problem of finding a maximum flow in the dynamic network $D = (N, A, h, q)$ is more complex than the problem of finding a maximum flow in the static network $G = (N, A, u)$. Fortunately, this issue can be solved by rephrasing the problem in the dynamic network D into a problem in the static network $R_1 = (V_1, E_1, u_1)$ called the reduced expanded network.

The static expanded network of dynamic network $D = (N, A, h, q)$ is the network $R = (V, E, u)$ with $V = \{i_t | i \in N, t \in H\}$, $E = \{(i_t, j_\theta) | (i, j) \in A, t \in \{0, 1, \dots, T - h(i, j; t)\}, \theta = t + h(i, j; t), \theta \in H\}$, $u(i_t, j_\theta) = q(i, j; t), (i_t, j_\theta) \in E$. The number of nodes in the static expanded network R is $n(T + 1)$ and number of arcs is limited by $m(T + 1) - \sum_{A} \overset{\circ}{h}(i, j)$, where $\overset{\circ}{h}(i, j) = min\{h(i, j; 0), \dots, h(i, j; T)\}$. It is easy to see that any flow in the dynamic network D from the source node 1 to the sink node n is equivalent to a flow in the static expanded network R from the source nodes $1_0, 1_1, \dots, 1_T$ to the sink nodes $n_0, n_1, \dots, n_T$ and vice versa. We can further reduce the multiple source, multiple sink problem in the static expanded network R to a single source, single sink problem by introducing a supersource node 0 and a supersink node $n + 1$ constructing the static super expanded network $R_2 = (V_2, E_2, u_2)$, where

$V_2 = V \cup \{0, n+1\}$, $E_2 = E \cup \{(0, 1_t) | t \in H\} \cup \{(n_t, n+1) | t \in H\}$, $u_2(i_t, j_\theta) = u(i_t, j_\theta)$, $(i_t, j_\theta) \in E$, $u_2(0, 1_t) = u_2(n_t, n+1) = \infty$, $t \in H$.

We construct the static reduced expanded network $R_1 = (V_1, E_1, u_1)$ as follows: we define the function $h_2 : E_2 \longrightarrow \mathbb{N}$, with $h_2(0, 1_t) = h_2(n_t, n+1) = 0$, $t \in H$, $h_2(i_t, j_\theta) = h(i, j; t)$, $(i_t, j_\theta) \in E$. Let $d_2(0, i_t)$ be the length of the shortest path from the source node 0 to the node i_t, and $d_2(i_t, n+1)$ the length of the shortest path from node i_t to the sink node $n+1$, with respect to h_2 in the network R_2. The computation of $d_2(0, i_t)$ and $d_2(i_t, n+1)$ for all $i_t \in V$ are performed by means of the usual shortest path algorithms. The network $R_1 = (V_1, E_1, u_1)$ have $V_1 = \{0, n+1\} \cup \{i_t | i_t \in V, d_2(0, i_t) + d_2(i_t, n+1) \leq T\}$, $E_1 = \{(0, 1_t) | d_2(1_t, n+1) \leq T, t \in H\} \cup \{(i_t, j_\theta) | (i_t, j_\theta) \in E, d_2(0, i_t) + h_2(i_t, j_\theta) + d_2(j_\theta, n+1) \leq T\} \cup \{(n_t, n+1) | d_2(0, n_t) \leq T, t \in H\}$ and u_1 are restrictions of u_2 at E_1.

Next, we construct the static reduced expanded network $R_1 = (V_1, E_1, u_1)$ using the notion of dynamic shortest path. The dynamic shortest path problem is presented in [3]. Let $d(1, i; t)$ be the length of the dynamic shortest path at time t from the source node 1 to the node i, and let $d(i, n; t)$ be the length of the dynamic shortest path at time t from the node i to the sink node n, with respect to h in the dynamic network D. Let us consider $H_i = \{t | t \in H, d(1, i; t) \leq t \leq T - d(i, n; t)\}, i \in N$, and $H_{i,j} = \{t | t \in H, d(1, i; t) \leq t \leq T - h(i, j; t) - d(j, n; \theta)\}, (i, j) \in A$. The multiple source, multiple sinks static reduced expanded network $R_0 = (V_0, E_0, u_0)$ has $V_0 = \{i_t | i \in N, t \in H_i\}$, $E_0 = \{(i_t, j_\theta) | (i, j) \in A, t \in H_{i,j}\}$, $u_0(i_t, j_\theta) = u_1(i, j; t), (i_t, j_\theta) \in E_0$. The static reduced expanded network $R_1 = (V_1, E_1, u_1)$ is constructed from the network R_0 as follows: $V_1 = V_0 \cup \{0, n+1\}, E_1 = E_0 \cup \{(0, 1_t) | 1_t \in V_0\} \cup \{(n_t, n+1) | n_t \in V_0\}, u_1(0, 1_t) = u_1(n_t, n+1) = \infty$, $1_t, n_t \in V_0$ and $u_1(i_t, j_\theta) = u_0(i_t, j_\theta)$, $(i_t, j_\theta) \in E_0$.

We notice that the static reduced expanded network $R_1(R_0)$ is always a partial subnetwork of the static super expanded network $R_2(R)$. In references [4], [5] it is shown that a dynamic flow for T periods in the dynamic network D is equivalent with a static flow in a static reduced expanded network R_1. Since an item released from a node at a specific time does not return to the location at the same or an earlier time, the static networks R, R_2, R_0, R_1 cannot contain any circuit, and therefore, are always acyclic.

In the most general dynamic model, the parameter $h\,(i) = 1$ is the waiting time at node i, and the parameter $q(i; t)$ is the upper bound for flow $g(i; t)$ that can wait at node i from time t to $t+1$. This most general dynamic model is not discussed in this paper.

The maximum flow problem for T time periods in the dynamic network D formulated in conditions (5), (6), (7) is equivalent with the maximum flow problem in the static reduced expanded network R_0 as follows:

$$f_0(i_t, V_0) - f_0(V_0, i_t) = \begin{cases} v_t, & \text{if } i_t = 1_t, t \in H_1 \\ 0, & \text{if } i_t \neq 1_t, n_t, t \in H_1, t \in H_n \\ -v_t, & \text{if } i_t = n_t, t \in H_n \end{cases} \tag{8a}$$

$$0 \leq f_0(i_t, j_\theta) \leq u_0(i_t, j_\theta), \quad (i_t, j_\theta) \in E_0 \tag{9}$$

$$max \sum_{H_1} v_t, \tag{10}$$

where by convention $i_t = 0$ for $t = -1$ and $i_t = n+1$ for $t = T+1$.

In stationary case the dynamic distances $d(1,i;t)$, $d(i,n;t)$ become static distances $d(1,i)$, $d(i,n)$.

4. Flows in bipartite static networks

In this section we consider that the static network $G = (N, A, u)$ is a bipartite static network. A bipartite network has the set of nodes N partitioned into two subsets N_1 and N_2, so that for each arc $(i,j) \in A$, either $i \in N_1$ and $j \in N_2$ or $i \in N_2$ and $j \in N_1$. Let $n_1 = |N_1|$ and $n_2 = |N_2|$. Without any loss of generality, we assume that $n_1 \leq n_2$. We also assume that the source node 1 belongs to N_2 (if the source node 1 belonged to N_1, then we could create a new source node $1^{'} \in N_2$, and we could add an arc $(1^{'}, 1)$ with $u(1^{'}, 1) = \infty$). A bipartite network is called unbalanced if $n_1 << n_2$ and balanced otherwise.

The observation of Gusfield, Martel, and Fernandez-Baca [6] that time bounds for several maximum flow algorithms automatically improves when the algorithms are applied without modification to unbalanced networks. A careful analysis of the running times of these algorithms reveals that the worst case bounds depend on the number of arcs in the longest node simple path in the network. We denote this length by L. For general network, $L \leq n-1$ and for a bipartite network $L \leq 2n_1 + 1$. Hence, for unbalanced bipartite network $L << n$. Column 3 of Table 1 summarizes these improvements for several network flow algorithms.

Algorithm	*Running time, general network*	*Running time, bipartite network*	*Running time modified version*
Maximumflows			
Dinic	n^2m	n_1^2m	*does not apply*
Karazanov	n^3	n_1^2n	$n_1m + n_1^3$
FIFOpreflow	n^3	n_1^2n	$n_1m + n_1^3$
Highestlabel	$n^2\sqrt{m}$	$n_1n\sqrt{m}$	n_1m
Excessscaling	$nm + n^2\log\bar{u}$	$n_1m + n_1n\log\bar{u}$	$n_1m + n_1^2\log\bar{u}$

Table 1. Several maximum flows algorithms.

Ahuja, Orlin, Stein, and Tarjan [2] obtained further running time improvements by modifying the algorithms. This modification applies only to preflow push algorithms. They called it the two arcs push rule. According to this rule, always push flow from a node in N_1 and push flow on two arcs at a time, in a step called a bipush, so that no excess accumulates at nodes in N_2. Column 4 of Table 1 summarizes the improvements obtained using this approach.

5. Maximum flows in bipartite dynamic networks

In this Section the dynamic network $D = (N, A, h, q)$ is bipartite.

We construct the static reduced expanded network $R_0 = (V_0, E_0, u_0)$ with $V_0 = W_1 \cup W_2$, $W_1 = \{k_\tau | k \in N_1, \tau \in H_k\}$, $W_2 = \{k_\tau | k \in N_2, \tau \in H_k\}$.

Theorem 5.1. *If the dynamic network $D = (N, A, h, q)$ is bipartite, then the static reduced expanded network $R_0 = (V_0, E_0, u_0)$ is bipartite.*

Proof. If the dynamic network $D = (N, A, h, q)$ is bipartite we have $N = N_1 \cup N_2$ and for each arc $(i, j) \in A$, either $i \in N_1$ and $j \in N_2$ or $i \in N_2$ and $j \in N_1$. For the static reduced expanded network $R_0 = (V_0, E_0, u_0)$ we have $V_0 = W_1 \cup W_2$, $W_1 = \{k_\tau | k \in N_1, \tau \in H_k\}$, $W_2 = \{k_\tau | k \in N_2, \tau \in H_k\}$. It results that for each arc $(i_t, j_\theta) \in E_0$, either $i_t \in W_1$ and $j_\theta \in W_2$ or $i_t \in W_2$ and $j_\theta \in W_1$. Therefore the network $R_0 = (V_0, E_0, u_0)$ is bipartite. □

Let w_1, w_2, ε_0 be $w_1 = |W_1|$ $w_2 = |W_2|$, $\varepsilon_0 = |E_0|$. If $n_1 << n_2$ then obvious that $w_1 << w_2$. In the static bipartite network R_0 we determine a maximum flow f_0 with a generalization of bipartite FIFO preflow algorithm.

We recall that the FIFO preflow algorithm might perform several saturating pushes followed either by a nonsaturating push or relabeled operation. We refer to this sequence of operations as a node examination. The algorithm examines active nodes in the FIFO order. The algorithm maintains the list Q of active nodes as a queue. Consequently, the algorithm selects a node i from the front of Q, performs pushes from this node, and adds newly active nodes to the rear of Q. The algorithm examines node i until either it becomes inactive or it is relabeled. In the latter case, we add node i to the rear of the queue Q. The algorithm ends when the queue Q of active nodes is empty. The reader interested in further details is urged to consult paper [2].

The modified version of FIFO preflow algorithm for maximum flow in bipartite is called bipartite FIFO preflow algorithm. A bipush is a push over two consecutive admissible arcs. It moves excess from a node $i_t \in W_1$ to another node $k_\tau \in W_1$. This approach means that the algorithm moves the flow over the path $\tilde{D} = (i_t, j_\theta, k_\tau), j_\theta \in W_2$, and ensures that no node in W_2 ever has any excess. A push of α units from node i_t to node j_θ decreases both $e(i_t)$ and $r_0(i_t, j_\theta)$ by α units and increases both $e(j_\theta)$ and $r_0(j_\theta, i_t)$ by α units.

We specify that maintain the arc list $E_0^+(i_t) = \{(i_t, j_\theta) | (i_t, j_\theta) \in E_0\}$. We can arrange the arcs in these lists arbitrarily, but the order, once decided, remains unchanged throughout the algorithm. Each node i has a current arc, which is an arc in $E_0^+(i_t)$ and is the next candidate for admissibility testing. Initially, the current arc of node i_t is the first arc in $E_0^+(i_t)$. Whenever the algorithm attempts to find an admissible arc emanating from node i_t, it tests whether the node's current arc is admissible. If not, it designates the next arc in the arc list as the current arc. The algorithm repeats this process until it either finds admissible arc or it reaches the end of the arc list.

The generalization bipartite FIFO preflow (GBFIFOP) algorithm is presented in Figure 1.

```
1: ALGORITHM GBFIFOP;
2: BEGIN
3: PREPROCESS
4: while $Q \neq \emptyset$ do
5:    select the node $i_t$ from the front of $Q$;
6:    BIPUSH/RELABE($i_t$);
7: end while
8: END.
```

```
PROCEDURE PREPROCESS;
BEGIN
$f_0 := 0; Q := \emptyset$;
compute the exact distance labels $d(i_t)$;
**for** $t \in H_1$ **do**
   $f_0(1_t, j_\theta) := u_0(1_t, j_\theta)$ and adds node $j_\theta$ to the rear of $Q$ for all $(1_t, j_\theta) \in E_0$
   $d(1_t) := 2w_2 + 1$;
**end for**
END.
```

```
PROCEDURE BIPUSH/RELABEL($i_t$);
BEGIN
select the first arc $(i_t, j_\theta)$ in $E_0^+(i_t)$ with $r_0(i_t, j_\theta) > 0$;
$\beta := 1$;
**repeat**
   **if** $(i_t, j_\theta)$ is admissible arc **then**
      select the first arc $(j_\theta, k_\tau)$ in $E_0^+(j_\theta)$ with $r_0(j_\theta, k_\tau) > 0$;
      **if** $(j_\theta, k_\tau)$ is admissible arc **then**
         push $\alpha := min\{e(i_t), r_0(i_t, j_\theta), r_0(j_\theta, k_\tau)\}$
         units of flow over the arcs $(i_t, j_\theta), (j_\theta, k_\tau)$;
         **if** $k_\tau \notin Q$ **then**
            adds node $k_\tau$ to the rear of $Q$;
         **end if**
      **else**
         **if** $(j_\theta, k_\tau)$ is not the last arc in $E_0^+(j_\theta)$ with $r_0(j_\theta, k_\tau) > 0$ **then**
            select the next arc in $E_0^+(j_\theta)$
         **else**
            $d(j_\theta) := min\left\{d(k_\tau) + 1 | (j_\theta, k_\tau) \in E_0^+(j_\theta), r_0(j_\theta, k_\tau) > 0\right\}$
         **end if**
      **end if**
      **if** $e(i_t) > 0$ **then**
         **if** $(i_t, j_\theta)$ is not the last arc in $E_0^+(j_\theta)$ with $r_0(i_t, j_\theta) > 0$ **then**
            select the next arc in $E_0^+(i_t)$
         **else**
            $d(i_t) := min\left\{d(j_\theta) + 1 | (i_t, j_\theta) \in E_0^+(j_\theta), r_0(i_t, j_\theta) > 0\right\}$;
            $\beta := 0$;
         **end if**
      **end if**
   **end if**
**until** $e(i_t) = 0$ or $\beta = 0$
**if** $e(i_t) > 0$ **then**
   adds node $i_t$ to the rear of $Q$;
**end if**
END.
```

FIGURE 1

We notice that any path in the residual network $\tilde{R_0} = (V_0, \tilde{E_0}, r_0)$ can have at most $2w_2 + 1$ arcs. Therefore we set $d(1_t) := 2w_2 + 1$ in PROCEDURE PREPROCES.

The correctness of the GBFIFOP algorithm results from correctness of the algorithm for maximum flow in bipartite network [2].

Theorem 5.2. *The GBFIFOP algorithm which determines a maximum flow into the bipartite dynamic network $D = (N, A, h, q)$ has the complexity $O(n_1mT^2 + n_1^3T^3)$.*

Proof. In Section 3 we specify that the maximum flow problem for T time periods in the dynamic network $D = (N, A, h, q)$ is equivalent with the maximum flow problem in the static reduced expanded network $R_0 = (V_0, E_0, u_0)$. The networks D and R_0 are bipartite with $N = N_1 \cup N_2$, $V_1 = W_1 \cup W_2$. We have $n_1 = |N_1|$, $n_2 = |N_2|$, $m = |A|$, $w_1 = |W_1|$, $w_2 = |W_2|$,$\varepsilon_0 = |E_0|$. The bipartite FIFO preflow algorithm determines a maximum flow into the bipartite static network $G = (N_1 \cup N_2, A, u)$ in $O(n_1m + n_1^3)$ how is specified in table from Section 4. We apply the generalization bipartite FIFO preflow algorithm in the static reduced expanded bipartite network R_0. Hence, the algorithm has the complexity $O(w_1\varepsilon_0 + w_1^3)$. From Section 4 we have $w_1 = n_1T$ and $\varepsilon_0 \leq mT$. As a result the algorithm has the complexity $O(n_1mT^2 + n_1^3T^3)$. □

6. Example

The support digraph of the bipartite dynamic network is presented in Figure 2 and time horizon being set $T = 5$, therefore $H = \{0, 1, 2, 3, 4, 5\}$. The transit times $h(i, j; t) = h(i, j), t \in H$ and the upper bounds (capacities) $q(i, j; t) = q(i, j), t \in H$ for all arcs are indicate in Table 2.

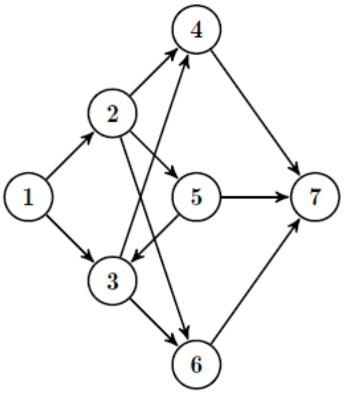

FIGURE 2. The support digraph of network $D = (N, A, h, q)$.

(i,j)	$(1,2)$	$(1,3)$	$(2,4)$	$(2,5)$	$(2,6)$	$(3,4)$	$(3,6)$	$(4,7)$	$(5,3)$	$(5,7)$	$(6,7)$
$h(i,j)$	1	1	3	1	2	3	1	1	1	1	1
$q(i,j)$	12	10	8	3	3	4	5	12	3	4	10

TABLE 2. The functions h, q.

We have $N_1 = \{2, 3, 7\}$ and $N_2 = \{1, 4, 5, 6\}$.

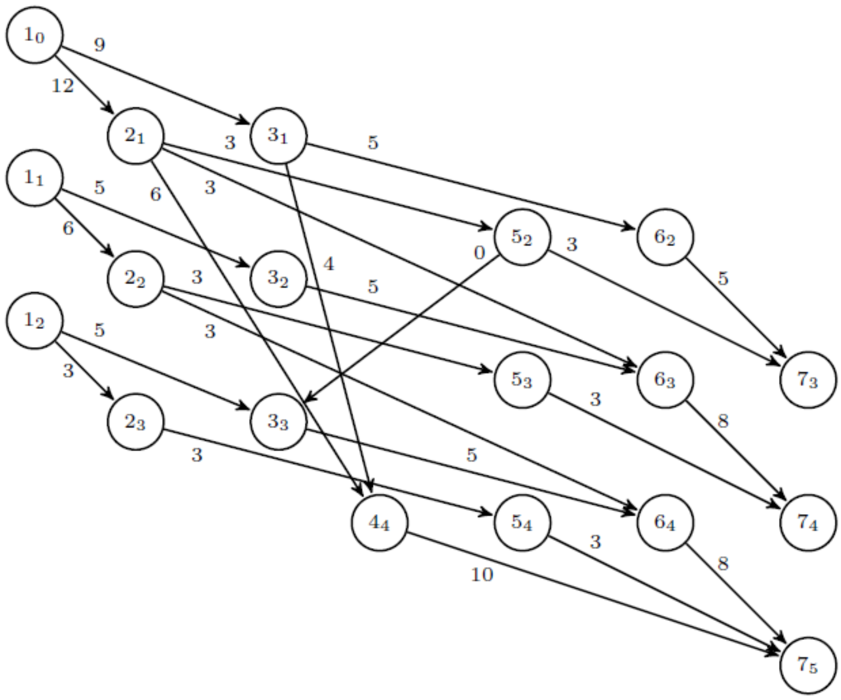

FIGURE 3. The network $R_0 = (V_0, E_0, f_0)$.

Applying the GBFIFOP algorithm we obtain the flows $f_0(i_t, j_\theta)$ which are indicated in Figure 3. We have $W_1 = \{2_1, 2_2, 2_3, 3_1, 3_2, 3_3, 7_3, 7_4, 7_5\}$ and $W_2 = \{1_0, 1_1, 1_2, 4_4, 5_2, 5_3, 5_4, 6_2, 6_3, 6_4\}$. A minimum $(1_0, 1_1, 1_2) - (7_3, 7_4, 7_5)$ cut in the static network R_0 is $\left[Y_0, \bar{Y_0}\right] = (Y_0, \bar{Y_0}) \bigcup (\bar{Y_0}, Y_0)$ with $Y_0 = \{1_0, 1_1, 1_2, 2_2, 2_3, 3_1, 3_2, 3_3\}$ and $\bar{Y_0} = \{2_1, 4_4, 5_2, 5_3, 5_4, 6_2, 6_3, 6_4, 7_3, 7_4, 7_5\}$. Hence, $\left[Y_0, \bar{Y_0}\right] = \{(1_0, 2_1), (2_2, 5_3), (2_2, 6_4), (2_3, 5_4), (3_1, 6_2), (3_1, 4_4), (3_2, 6_4)\} \cup \{(5_2, 3_3)\}$. We have $w_0 = f_0(Y_0, \bar{Y_0}) - f_0(\bar{Y_0}, Y_0) = 40 - 0 = 40 = u_0(Y_0, \bar{Y_0})$. Hence, f_0 is a maximum flow.

7. Conclusions

In this paper we have developed an algorithm for maximum flows problem in bipartite dynamic networks with lower bounds zero. This problem has not been treated so far. We demonstrate the fact that if the dynamic network $D = (N, A, h, q)$ is bipartite, then the static reduced expanded network $R_0 = (V_0, E_0, u_0)$ is bipartite. Therefore we solved the maximum flows problem in bipartite dynamic networks with lower bound zero by rephrasing into a problem in bipartite static network. We have extended the bipartite FIFO preflow algorithm of Ahuja et al. [2] for the static reduced expanded network $R_0 = (V_0, E_0, u_0)$ which is a network with multiple source and multiple sinks. For the generalization bipartite FIFO preflow algorithm we have presented the complexity. In Section 6 we presented an example for the clarity of paper.

Many interesting flow problems in bipartite dynamic networks are still open: the generalization of the highest label preflow push algorithm, the generalization of the excess scaling algorithm, the parametric maximum flow problem, the minimum cost flow problem. Other research directions are possible.

References

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

[2] R. Ahuja, J. Orlin, C. Stein, R. Tarjan, Improved algorithms for bipartite network flows, *SIAM Journal of Computing* **23** (1994), 906–933.

[3] X. Cai, D. Sha, C. Wong, *Time-varying Nework Optimization*, Springer, 2007.

[4] E. Ciurea, Second best temporally repeated flow, Korean *Journal of Computational and Applied Mathematics* **9** (2002), no. 1, 77–86.

[5] L. Ford, D. Fulkerson, *Flow in Networks*, Princeton University Press, Princenton, New Jersey, 1962.

[6] D. Gusfield, C. Martel, D. Fernandez-Baca, Fast algorithms for bipartite network flow, *SIAM Journal of Computing* **16** (1987), 237–251.

[7] C. Schiopu, The maximum flows in bipartite dynamic networks, *Bulletin of the Transilvania University of Braşov* **7(56)** (2014), no. 1, 193–202.

[8] C. Schiopu, E. Ciurea, The maximum flows in bipartite dynamic networks with lower bounds. The static approach, *Proceedings in IEEE Xplore of the 6th International Conference on Computers, Coumincations and Control* (2016), 10–15.

[9] W. Wilkinson, An algorithm for universal maximal dynamic flows in network, *Operation Research* **19** (1971), 1602–1612.

(Camelia Schiopu) Department of Mathematics and Computer Science, Transilvania University of Brasov, 29 Eroilor, Brasov, 500036, Romania
E-mail address: camelia.s@unitbv.ro