Bulletin of the *Transilvania* University of Braşov • Vol 12(61), No. 1 - 2019
Series III: Mathematics, Informatics, Physics, 177-198
https://doi.org/10.31926/but.mif.2019.12.61.1.14

MAXIMUM FLOWS IN BIPARTITE DYNAMIC NETWORKS

Camelia SCHIOPU[1]

Dedicated to the 75th birthday of Professor Eleonor Ciurea

Abstract

In this paper we study the maximum flow in bipartite dynamic network and make a synthesis of the papers [17], [18], [19], [20]. We solve this problem by dynamic approach and static approach. In a bipartite static network the several maximum algorithms can be substantially improved. The basic idea in these improvements is a two arcs push rule in case of maximum algorithms. For these problems we give examples.

2000 *Mathematics Subject Classification:* 0B10, 90C35, 05C35, 68R10.

Key words: bipartite static network, bipartite dynamic network, maximum flow.

1 Introduction

The problem of flows in network is the fundamental and the most important part of graph theory and combinatorial optimization. The static network flow models arise directly in problems as far reaching as machine scheduling, the assignment of computer modules to computer processor, tanker scheduling etc. [1]. In the network flow literature the difference between dynamic flow and static flow is given by the crossing time of the arc flow in the network. In some applications, time is an essential ingredient [3], [4], [7], [9], [14], [21], [22]. 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 [2], [13].

The static approach of maximum flow problem in bipartite dynamic networks with lower bounds zero is treated in the paper [18] and the dynamic approach is treated in the paper [17]. The static approach of maximum flows problem in bipartite dynamic networks with positive lower bounds is treated in the paper [19], [20].

[1]Faculty of Mathematics and Informatics, *Transilvania* University of Braşov, Romania, e-mail: camelia.s@unitbv.ro

In this paper we present a syhthesis of the problem of maximum flows in bipartite dynamic networks from paper [17], [18], [19], [20]. Further on, in Section 2 we discuss some basic notions and results for maximum flow problem in general static networks and in dynamic networks. Section 3 deals with the maximum flow problem in bipartite dynamic networks with lower bound zero, static and dynamic approach, and the maximum flow problem in bipartite dynamic networks with lower bounds positive (static approach) and we give some examples for the problems presented.

2 Terminology and preliminaires

2.1 Maximum flows in general static networks

In this section we discuss some notions and results about maximum flows in static networks.

Let $G = (N, A, u, l)$ be a general static network with the set of nodes $N = \{1, \ldots, n\}$ where 1 is the source node and n is the sink node, 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, the lower bound function l, $l : A \to \mathbb{N}$ with $\mathbb{N}$ the natural number set. The capacity $u(i, j)$ of (i, j) from A represent the maximum quantity that can cross the arc, the lower bound $l(i, j)$ of (i, j) form A represent the minimum quantity that can cross the arc and $v(i)$ represent the value of node i.

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

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

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

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.

Many efficient algorithms have been developed to solve this problem and are presented in the book of authors Ahuja, Magnanti and Orlin [1] and in the paper of other authors presented in the bibliography.

Definition 1. *For the maximum flow problem we define the capacity of the cut* $1 - n$ $[X, \bar{X}]$ *as follows:*

$$c[X, \bar{X}] = \sum_{(X,\bar{X})} u(i, j) - \sum_{(\bar{X},X)} l(j, i) \tag{2}$$

If we use the notation $\sum_{(X,\bar{X})} b(i, j) = b(X, \bar{X})$ then for the capacity of cut $[X, \bar{X}]$ we can use the relation:

$$c[X, \bar{X}] = u(X, \bar{X}) - l(\bar{X}, X) \tag{3}$$

If $l(i,j) = 0$ for all $(i,j) \in A$ then $c[X,\bar{X}] = u(X,\bar{X})$ is the relationship of defining the capacity of a cut if the lower bound is null.

Definition 2. *A $1-n$ cut whose capacity is minimum between all the $1-n$ cuts is called the minimum cut and is noted with $[X^*,\bar{X}^*]$.*

Theorem 1. *If f is a flow with value v in the network $G = (N,A,l,u)$ and $[X,\bar{X}]$ a $1-n$ cut then f checks the relations:*

$$v = f[X,\bar{X}] \leq c[X,\bar{X}],\ f[X,\bar{X}] = f(X,\bar{X}) - f(\bar{X},X) \tag{4}$$

Theorem 2. *In a network $G = (N,A,l,u)$ the maximum flux value from the source node 1 to the sink node n is equal with the $1-n$ minimum cut capacity $[X^*,\bar{X}^*]$, namely:*

$$v = f[X^*,\bar{X}^*] = c[X^*,\bar{X}^*] \tag{5}$$

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

$$f(N,i) - f(i,N) \geq 0, i \in N - \{1,n\} \tag{6a}$$

$$l(i,j) \leq f(i,j) \leq u(i,j), (i,j) \in A \tag{6b}$$

A pseudoflow is a function satisfying only the constraint 6b.

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

$$e(i) = f(N,i) - f(i,N) \tag{7}$$

and if $e(i) > 0$, $i \in N - \{1,n\}$ then we say that node i is an active node. If $e(i) = 0$ then i is called a balanced node. A preflow which satisfies the condition $e(i) = 0, i \in N - \{1,n\}$ is a flow. So, a flow is a particular case of preflow.

Given a flow (preflow) f, the residual capacity $r(i,j)$ of any arc $(i,j) \in A$ is $r(i,j) = u(i,j) - f(i,j) + f(j,i) - l(j,i)$. 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

$$d(n) = 0 \tag{8a}$$

$$d(i) \leq d(j) + 1, (i,j) \in \tilde{A} \tag{8b}$$

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 k consisting entirely of admissible arcs as an admissible path.

Whereas the maximum flow problem with zero lower bounds always has a feasible solution (since the zero flow is feasible), the problem with non-negative lower bounds could be infeasible. Therefore, the maximum flow problem with non-negative lower bounds can be solved in two phases:

(P1) this phase determines a feasible flow if one exists;

(P2) this phase converts a feasible flow into a maximum flow.

The problem in each phase essentially reduces to solving a maximum flow problem with zero lower bounds. Consequently, it is possible to solve the maximum flow problem with non-negative lower bounds by solving two maximum flow problems, each with zero lower bounds. For more details see the book [1].

The flows $f(i,j)$ and $f(j,i)$, are calculated with the relations: $f(i,j) = l(i,j) + \max\{0, u(i,j) - r(i,j) - l(i,j)\}$ and $f(j,i) = l(j,i) + \max\{0, u(j,i) - r(j,i) - l(j,i\}$.

To determine the maximum flow we can use the increasing path algorithms or preflows algorithms.

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].

2.2 Maximum flows in bipartite static networks

Definition 3. *We say that $G = (N, A)$ is a bipartite graph if the set of nodes N can be partitioned into two disjoint subsets N_1 and N_2 so for each arc $(i, j) \in A$ we have either $i \in N_1$ and $j \in N_2$, or $i \in N_2$ and $j \in N_1$, where $N_1 \neq \emptyset$, $N_2 \neq \emptyset$, $N_1 \cap N_2 = \emptyset$, $N_1 \cup N_2 = N$.*

We denote with $G = (N_1 \cup N_2, A, l, u)$ the bipartite network corresponding to the bipartite graph introduced by the definition 3. Let $n_1 = |N_1|$ and $n_2 = |N_2|$. We deduct that $|N| = n_1 + n_2$.

One of the problems is whether or not a graph is a bipartite graph. Fortunately, there are many simple ways to solve this problem. One of these is based on the following idea that characterizes bipartite networks, Jungnickel [15].

Property 1. *A graph G is a bipartite graph if and only if every cylce in G contains an even number of arcs.*

By improving the complexity of the Dinic algorithm [8] and Karzanov and Naor [16], Gusfield, Martel and Fernandez-Baca [13] described maximal flow algorithms on bipartite networks. Ahuja, Orlin, Stein and Tarjan [2] have made significant improvements and have shown that it is possible to obtain a new time limit for bipartite networks. In paper [13] Gusfield, Martel and Fernandez-Baca describe problems that can be solved with flow in networks in bipartite graphs.

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 [2].

The observation of Gusfield, Martel, and Fernandez-Baca [13] 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 a 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.
Ahuja, Orlin, Stein, and Tarjan [2] obtained further running time improvements

Algorithm	*Running time, general network*	*Running time, bipartite network*	*Running time modified version*
Maximum flows			
Dinic	n^2m	n_1^2m	*does not apply*
Karazanov	n^3	n_1^2n	$n_1m + n_1^3$
FIFO preflow	n^3	n_1^2n	$n_1m + n_1^3$
Highest label	$n^2\sqrt{m}$	$n_1n\sqrt{m}$	n_1m
Excess scaling	$nm + n^2 \log \bar{u}$	$n_1m + n_1n \log \bar{u}$	$n_1m + n_1^2 \log \bar{u}$

Table 1: Several maximum flows algorithms

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.

We recall that the FIFO preflow algorithm might perform several saturating pushes followed either by a nonsaturating push or relabeled operation [1]. 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 terminates when the queue Q of active nodes is empty. (see [1])

The modified version of FIFO preflow algorithm for maximum flow in bipartite network is called bipartite FIFO preflow algorithm. A bipush is a push over two consecutive admissible arcs. It moves excess from a node $i \in N_1$ to another node $k \in N_1$. This approach means that the algorithm moves the flow over the path $\tilde{D} = (i, j, k), j \in N_2$, and ensures that no node in N_2 ever has any excess. A push of α units from node i to node j decreases both $e(i)$ and $r_0(i, j)$ by α units and increases both $e(j)$ and $r_0(j,i)$ by α units (see [2]). We specify that for $l = 0$ and $f = 0$ we get $r = u$.

In the paper [2] the following bipartite FIFO preflow (BFIFOP) algorithm is presented :

```
ALGORITHM BFIFOP;
BEGIN
   PREPROCESS;
   while Q ≠ ∅ do
      BEGIN
         select the node i from the front of Q
         BIPUSH/RELABEL(i)
      END
END.
```

```
PROCEDURE PREPROCESS;
BEGIN
   f =: 0; Q := ∅;
   push u(1, j) units of flow on each (1, j) ∈ A and add j to rear of Q if
   e(j) > 0 and j ≠ n;
   compute the exact distance labels d(i) from t to i in the residual network;
   d(1) = 2n_1 + 1;
END;
```

```
PROCEDURE BIPUSH/RELABEL(i);
BEGIN
   if there is an admissible arc (i, j)
     then BEGIN
             select an admissible arc (i, j);
             if there is an admissible arc (j, k)
                then BEGIN
                        select an admissible arc (j, k);
                        push α := min {e(i), r(i, j), r(j, k)} units of flow along
                        the path (i, j, k) and adds k to Q if k ∉ Q and k ≠ n;
                     END
                else
                     d(j) := min{d(k) + 1|(j, k) ∈ A, r(j, k) > 0}
          END
     else
          d(i) := min{d(j) + 1|(i, j) ∈ A, r(i, j) > 0}
END;
```

Figure 1: The FIFO bipartite preflux algorithm (BFIFOP) for the maximum flow

Theorem 3. *The FIFO preflux algorithm establishes a maximum flow in the network* $G = (N_1 \cup N_2, A, u)$.

Theorem 4. *The FIFO preflux algorithm has complexity* $n_1^2 n$.

For more information see [2]. We remark the fact that we have used the notations from this paper and have specified that this algorithm runs on networks G with $l = 0$, a single source node 1, a single sink node n.

2.3 Maximum flows in general dynamic networks

Definition 4. *A network* $D = (N, A, h, e, q, H)$ *with* N *the set of nodes,* A *the set of arcs,* h *the transit time function* $h : A \times H \to \mathbb{N}$, *the time lower bound function* $e : A \times H \to \mathbb{N}$, *the time upper bound function* $q : A \times H \to \mathbb{N}$ *and* H *the set of time periods* $H = \{0, 1, \ldots, T\}$ *is called dynamic network.*

Definition 5. *A network* $D = (N, A, h, e, q, H)$ *is a stationary dynamic network if functions* h, e, q *for all arcs* $(i, j) \in A$ *are time independent* $h(i, j; \theta) = h(i, j)$, $e(i, j; \theta) = e(i, j)$ *and* $q(i, j; \theta) = q(i, j)$, $\forall (i, j) \in A$ *and* $\theta \in \{0, 1, \ldots, T\}$.

Dynamic network models arise in many problem settings, especially in economic problems, such as production-distribution systems and economic planning.

Let $D = (N, A, h, q, H)$ be a dynamic network with $e = 0$, the set of nodes $N = \{1, \ldots, n\}$, the set of arcs $A = \{a_1, \ldots, a_m\}$, 1 the source node and n the sink node. 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 i at time t and arrives at 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 dynamic network $D = (N, A, h, e = 0, q, H)$:

$$\sum_{t=0}^{T} (g(1, N; t) - \sum_{\tau} g(N, 1; \tau)) = w \tag{9a}$$

$$g(i, N; t) - \sum_{\tau} g(N, i; \tau) = 0, i \neq 1, n,\ t \in H \tag{9b}$$

$$\sum_{t=0}^{T} (g(n, N; t) - \sum_{\tau} g(N, n; \tau)) = -w \tag{9c}$$

$$0 \leq g(i, j; t) \leq q(i, j; t), \quad (i, j) \in A \quad , \quad t \in H \tag{10}$$

$$max \quad w, \tag{11}$$

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 $t \in \{T - h(i, j; t) + 1, \ldots, T\}$.

Obviously, the problem of finding a maximum flow in the dynamic network $D = (N, A, h, e = 0, q, H)$ 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 $R = (V, E, u)$ called the reduced expanded network, [4], [9].

The static expanded network of dynamic network $D = (N, A, h, e = 0, q, H)$ is a network $R = (V, E, u)$ with $V = \{i_t | i \in N, t \in H\}$, $E = \{(i_t, j_\theta) | (i, j) \in A, t \in \{0, 1, \ldots, T - h(i, j; t)\}, \theta = t + h(i, j; t), \theta \in H\}$, $u(i_t, j_\theta) = q(i, j; t), (i_t, j_\theta) \in E$. The number of nodes in the static expanded network R is $n(T+1)$ and number of arcs is bounded by $m(T+1) - \sum_A \overset{\circ}{h}(i, j)$, where $\overset{\circ}{h}(i, j) = \min\{h(i, j; 0), \ldots, h(i, j; T)\}$. 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 $1_0, 1_1, \ldots, 1_T$ to the sink nodes $n_0, n_1, \ldots, n_T$ 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 $V_2 = V \cup \{0, n + 1\}$, $E_2 = E \cup \{(0, 1_t) | t \in H\} \cup \{(n_t, n + 1) | t \in H\}$, $u_2(i_t, j_\theta) = u(i_t, j_\theta)$, $(i_t, j_\theta) \in E$, $l_2(0, 1_t) = l_2(n_t, n + 1) = 0$, $u_2(0, 1_t) = u_2(n_t, n + 1) = \infty$, $t \in H$.

Now, 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 network R_2. The computation of $d_2(0, i_t)$ and $d_2(i_t, n + 1)$ for all $i_t \in V$ is performed by means of the usual shortest path algorithms. The network $R_1 = (V_1, E_1, u_1)$ has $V_1 = \{0, n+1\} \cup \{i_t | i_t \in V, d_2(0, i_t) + d_2(i_t, n+1) \leq T\}$, $E_1 = \{(0, 1_t) | d_2(1_t, n+1) \leq T, t \in H\} \cup \{(i_t, j_\theta) | (i_t, j_\theta) \in E, d_2(0, i_t) + h_2(i_t, j_\theta) + d_2(j_\theta, n + 1) \leq T\} \cup \{(n_t, n + 1) | d_2(0, n_t) \leq T, t \in H\}$ and u_1 is restriction of u_2 at E_1[19].

We remark that the static reduced expanded network R_1 is always a partial subnetwork of static super expanded network R_2. In references [7], [9] it is shown that a dynamic flow for T time periods in the dynamic network D with $e = 0$ 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 that location at the same or an earlier time, the static networks R, R_2, R_1 cannot contain any circuit, and therefore they 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 upper bound for flow $g(i; t)$ that can wait at node i from time t to $t + 1$.

The maximum flow problem for T time periods in the dynamic network D as stated in the conditions (9), (10), (11) is equivalent with the maximum flow problem in the static reduced expanded network R_1, as follows:

$$\sum_{j_\theta} f_1(i_t, j_\theta) - \sum_{k_\tau} f_1(k_\tau, i_t) = \begin{cases} v_1(i_t), & \text{if } i_t = 1_t \\ 0, & \text{if } i_t \neq 1_t, n_t \\ -v_1(i_t), & \text{if } i_t = n_t \end{cases} \tag{12a}$$

$$0 \le f_1(i_t, j_\theta) \le u_1(i_t, j_\theta), \quad (i_t, j_\theta) \in E_1 \tag{12b}$$

$$\max v_1, \tag{12c}$$

with $v_1 = \sum_t v_1(1_t)$.

In the case of $h(i,j;t) = h(i,j)$, $t \in H$ the dynamic distances $d(1,i;t), d(i,n;t)$ become static distances $d(1,i), d(i,n)$.

There are two approaches to determining a maximum flow in the dynamic network $D = (N, A, h, e = 0, q, H)$: static approach and dynamic approach. The static approach consists of determining a maximum flux in the static reduced expanded network $R_1 = (V_1, E_1, u_1)$. The dynamic approach is used in the stationary case [23]. It is not necessary to construct a static reduced expanded network to solve the problem of maximum dynamic flow for any T. The maximum dynamic flow in the stationary case can be generated from the f flow of maximum value and minimum time in the static network $D = (N, A, h, e = 0, q, H)$, where $h(i,j)$ is the cost (time) for any arc $(i,j) \in A$[19]. The algorithm for maximum dynamic flow in the stationary case (SMDF) is presented below:

```
ALGORITHM SMDF;
BEGIN
    AMVMCF(G, f)
    ADFEF(f, r(P_1), ..., r(P_k))
    ARF(r(P_1), ..., r(P_k))
END.
```

Figure 2: Algorithm for maximum dynamic flow in stationary dynamic network (SMDF)

The procedure AMVMCF performs the algorithm for minimum cost and maximum value flow f in the network G. For statements we suppose that the algorithm of Klein variant is used (minimum mean cycle canceling algorithm, see [1]). This algorithm has the complexity $O\left(n^2 m^3 \log n\right)$.

The procedure ADFEF performs the algorithm for decomposition of flow f in elementary flows with $r(P_1), ..., r(P_k)$ path flows. Is necessary that $h(P_i) \le T$. This algorithm has complexity $O(m^2)$. The procedure ARF performs the algorithm for send $r(P_i)$ flow, $i = 1 \ldots, k$, starting out from source node 1 at time periods 0 and repeat it after each time period as long as there is enough time left in the horizon for the flow along the path to arrive at the sink node n. This algorithm has complexity $O(kT)$. Hence, the algorithm for stationary maximum dynamic flow has complexity $O(n^2 m^3 \log n)$ (we consider that k$T \le n^2 m^3 \log n$). The flow obtained with SMDF is called a temporally repeated flow for the obvious reason that it consists of repeated shipments along the same flow paths from 1 to n. The maximum value of a temporally repeated flow obtained with SMDF is:

$$v_H = (T+1)v - \sum_A h(i,j)f(i,j) \tag{13}$$

where v is the maximum value of the flow f obtained with AMVMCF.

Theorem 5. *The algorithm correctly calculates the maximum dynamic flow in the $D = (N, A, h, e = 0, q, H)$ network.*

Theorem 6. *The SMDF algorithm has complexity $O(n^2m^3 \log n)$.*

3 Maximum flows in bipartite dynamic networks

3.1 The maximum flows in bipartite dynamic network with $e = 0$

First, we present the static approach of the maximum flows in bipartite dynamic network with $e = 0$. We consider that the dynamic network $D = (N, A, h, e = 0, q, H)$ is bipartite.

We construct the static reduced expanded network $R_0 = (V_0, E_0, l_0, u_0)$ using the shortest path problem presented in [4]. 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 $u_1(i_t, j_\theta) = u_0(i_t, j_\theta)$, $(i_t, j_\theta) \in E_0$.

Theorem 7. *If the dynamic network $D = (N, A, h, e = 0, q, H)$ is bipartite, then the static reduced expanded network $R_0 = (V_0, E_0, u_0)$ is bipartite.*

The proof of this Theorem is presented in the paper [18].

Let w_1, w_2, ε_0 with $w_1 = |W_1|$ $w_2 = |W_2|$, $\varepsilon_0 = |E_0|$. If $n_1 << n_2$ then it is 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.

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, where $\alpha = min\{e(i_t), r_0(i_t, j_\theta), r_0(j_0, i_t)\}$.

We specify that we 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 (GBFIFOP1) algorithm is presented in Figure 3.

ALGORITHM GBFIFOP1;
BEGIN
    PREPROCESS;
    while $Q \neq \emptyset$ **do**
        BEGIN
            select the node i_t from the front of Q;
            BIPUSH/RELABE(i_t);
        END
END.

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

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

Figure 3: The generalized bipartite FIFO preflow algorithm (GBFIFOP1)

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 GBFIFOP1 algorithm results from correctness of the algorithm for maximum flow in bipartite network [2].

Theorem 8. *The GBFIFOP1 algorithm which determines a maximum flow into the bipartite dynamic network* $D = (N, A, h, e = 0, q, H)$*, has the complexity* $O(n_1 m T^2 + n_1^3 T^3)$*[18].*

We present an example for determining the maximum flow in bipartite dynamic network.

The support digraph of the bipartite dynamic network is presented in Figure 4 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 indicated in Table 2.

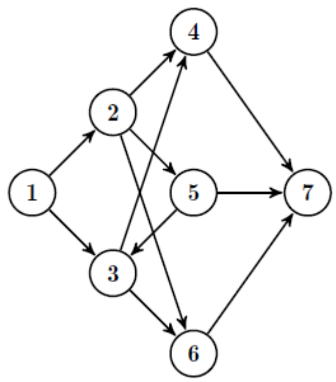

Figure 4: The support digraph of network $D = (N, A, h, e = 0, q, H)$

(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 $N_2 = \{1, 4, 5, 6\}$.

Applying the GBFIFOP1 algorithm we obtain the flows $f_0(i_t, j_\theta)$ which are indicated in Figure 5. 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 $[Y_0, \bar{Y_0}] = (Y_0, \bar{Y_0}) \bigcup (\bar{Y_0}, Y_0)$ 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, $[Y_0, \bar{Y_0}] = \{(1_0, 2_1), (2_2, 5_3), (2_2, 6_4), (2_3, 5_4), (3_1, 6_2), (3_1, 4_4), (3_2, 6_4)\} \cup \{(5_2, 3_3)\}$. 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.

Next, we present the dynamic approach of the maximum flows in bipartite dynamic network with e = 0.

In this section we consider the maximum flows in bipartite dynamic networks in the stationary case i.e. $h(i, j; t) = h(i, j)$, $q(i, j; t) = q(i, j)$, $(i, j) \in A, t \in H$. We use the algorithm SMDF which was presented in Section 2.3. In this Section the dynamic network $D = (N, A, h, q)$ is bipartite.

We consider the bipartite static network $G = (N, A, c, u)$ where $c(i, j) = h(i, j)$, $u(i, j) = q(i, j), (i, j) \in A$. The procedure AMVMCF from algorithm

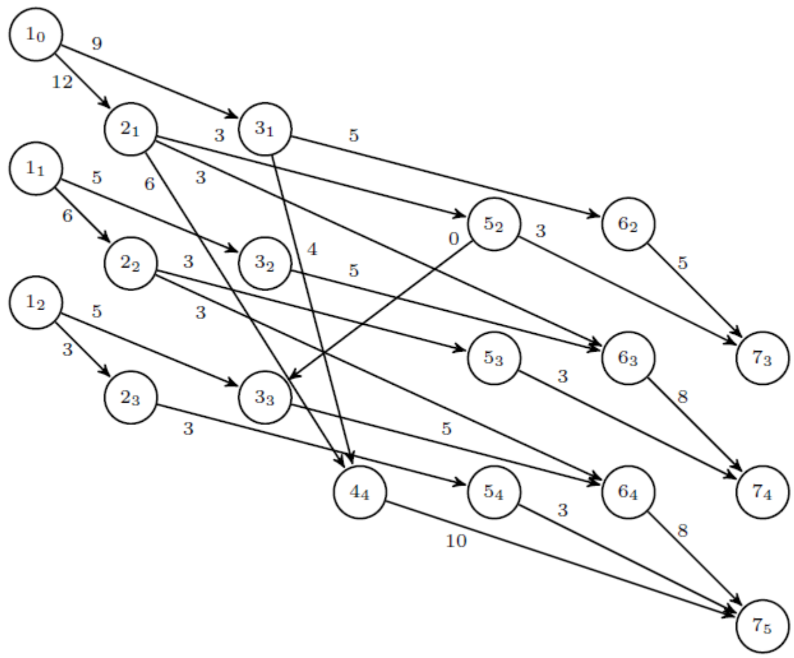

Figure 5: The network $R_0 = (V_0, E_0, f_0)$.

SMDF performs the algorithm for maximum value and minimum cost flow $\overset{*}{f}^{*}$ in bipartite static network. The modified version of cost scaling algorithm for bipartite static network G starts with any feasible flow. In this case the feasible flow is a maximum flow $\overset{*}{f}$. We determine a flow $\overset{*}{f}$ with maximum value with modified version of FIFO preflow which has the complexity $O(n_1m + n_1^3)$. The modified version of cost scaling algorithm has the complexity $O(n_1m + n_1^3 \log(n_1\bar{c})) = O(n_1m + n_1^3 \log(n_1\bar{h}))$.

Theorem 9. *The algorithm SMDF correctly computes the maximum flow in bipartite stationary dynamic network [17].*

Theorem 10. *The algorithm SMDF applied to bipartite stationary dynamic network has the complexity $O(max\{n_1m + n_1^3 \log(n_1\bar{h}), nT\})$ [17].*

We present an example for this problem.

The support digraph of bipartite stationary dynamic network is presented in Figure 6 and time horizon being set $T = 5$, therefore $H = \{0, 1, 2, 3, 4, 5\}$. The transit times $h(i, j)$ and the upper bounds (capacities) $q(i, j)$ for all arcs are indicated in Table 3.

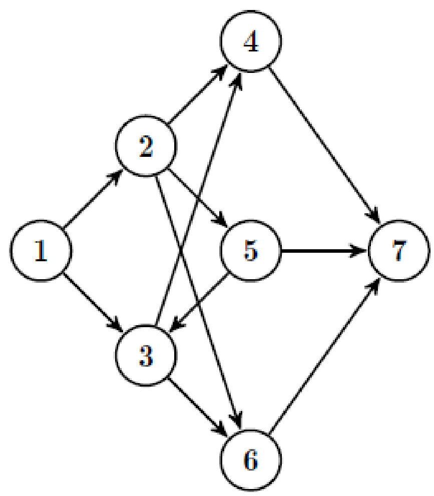

Figure 6: The support digraph of network $D = (N, A, h, q)$

(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
$\overset{*}{f}(i,j)$	12	9	8	1	3	4	5	12	0	1	8
$\overset{*}{f}{}^*(i,j)$	12	9	6	3	3	4	5	10	0	3	8

Table 3: The functions $h, q, \overset{*}{f}, \overset{*}{f}{}^*$

The maximum flow $\overset{*}{f}$ and the maximum flow of minimum cost $\overset{*}{f}{}^*$ obtained with the procedure AMVMCF in bipartite static network $G = (N, A, c, u)$ are presented in Table 3.

Applying the procedure ADFEF we obtain the results which are indicated in Table 4.

P_s	$r(P_s)$	$h(P_s)$	$\gamma(P_s)$
$P_1 = (1,2,5,7)$	3	3	3
$P_2 = (1,3,6,7)$	5	3	3
$P_3 = (1,2,6,7)$	3	4	2
$P_4 = (1,2,4,7)$	6	5	1
$P_5 = (1,3,4,7)$	4	5	1

Table 4: The results of procedure ADFEF

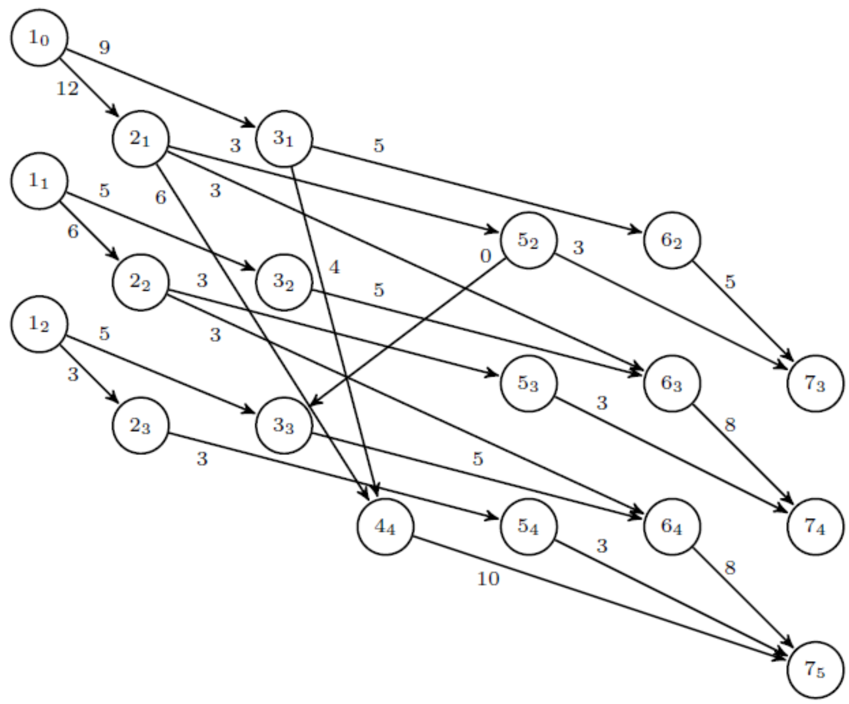

Figure 7: The network $R_0 = (V_0, E_0, u_0)$ with flow $\overset{*}{f}_0 = f_0$

The procedure ARF generates the flow f_0 in network $R_0 = (V_0, E_0, u_0)$. The network R_0 with the flow f_0 are presented in Figure 7. With formula (12) we obtain $\overset{*}{w}_0 = (5+1) \cdot 21 - (1 \cdot 12 + 1 \cdot 9 + 3 \cdot 6 + 1 \cdot 3 + 2 \cdot 3 + 3 \cdot 4 + 1 \cdot 5 + 1 \cdot 10 + 1 \cdot 0 + 1 \cdot 3 + 1 \cdot 8) = 126 - 86 = 40$. A minimum $(1_0, 1_1, 1_2) - (7_3, 7_4, 7_5)$ cut in 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, i.e. $\overset{*}{f}_0 = f_0$ and $\overset{*}{w}_0 = 40 = w_0$.

We remark that $\gamma(P_s) = (T+1) - h(P_s)$.

3.2 Maximum flows in bipartite dynamic network with $e > 0$

Next, we present the problem of maximum flow in bipartite dynamic networks with lower bounds. In this case the dynamic network $D = (N, A, h, e, q, H)$ is bipartite.

We construct the static reduced expanded network $R_0 = (V_0, E_0, l_0, u_0)$ and we notice the fact that the network R_0 is a bipartite network with $V_0 = W_1 \cup W_2$, $W_1 = \{i_t | i \in N_1, t \in H\}$, $W_2 = \{i_t | i \in N_2, t \in H\}$. 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 it is 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 generalize the BFIFOP for a network $R_0 = (V_0, E_0, l_0, u_0)$ where $l_0 > 0$, there are multiple source nodes $1_t, t \in H_1$ and there are multiple sink nodes $n_t, t \in H_n$. Also, we present a pseudocode in detail.

The generalised bipartite FIFO preflow (GBFIFOP2) algorithm is presented below.

ALGORITHM GBFIFOP2;
BEGIN
    PREPROCESS;
    while $Q \neq \emptyset$ **do**
        BEGIN
            select the node i_t from the front of Q;
            BIPUSH/RELABE(i_t);
        END
END.

PROCEDURE PREPROCESS;
BEGIN
    f_0 is a feasible flow in R_0; $Q := \emptyset$;;
    compute the exact distance labels $d(i_t)$;
    for $t \in H_1$ **do**
        BEGIN
            $f_0(1_t, j_\theta) := u_0(1_t, j_\theta)$ and adds node j_θ to the rear of Q for all $(1_t, j_\theta) \in E_0$
            $d(1_t) := 2w_2 + 1$;
        END
END;

PROCEDURE BIPUSH/RELABEL(i_t);
BEGIN
    select the first arc (i_t, j_θ) in $E_0^+(i_t)$ with $r_0(i_t, j_\theta) > 0$;
    $\beta := 1$;
    repeat
      if (i_t, j_θ) is admissible arc;
        then
          BEGIN
            select the first arc (j_θ, k_τ) in $E_0^+(j_\theta)$ with $r_0(j_\theta, k_\tau) > 0$;
            if (j_θ, k_τ) is admissible arc
              then
                BEGIN
                  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_τ to the rear of Q;

end if
END
else
if (j_θ, k_τ) 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_θ) 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
BEGIN
$d(i_t) := min\left\{d(j_\theta) + 1 | (i_t, j_\theta) \in E_0^+(j_\theta), r_0(i_t, j_\theta) > 0\right\}$
$\beta := 0;$
END
end if
end if
END
end if
until $e(i_t) = 0$ or $\beta = 0$
if $e(i_t) > 0$
adds node i_t to the rear of Q
end if
END;

Figure 8: The generalized bipartite FIFO preflow algorithm (GBFIFOP2)

We notice that any path in the residual network $\tilde{R}_0 = (V_0, \tilde{E}_0, r_0)$ can have at most $2w_1 + 1$ arcs. Therefore, we set $d(1_t) := 2w_1 + 1$ in PROCEDURE PREPROCES.

The correctness of the GBFIFOP2 algorithm results from correctness of the algorithm for maximum flow in bipartite network [2].

Theorem 11. *The GBFIFOP2 algorithm which determines a maximum flow into the bipartite dynamic network $D = (N, A, h, e, q, H)$ has the complexity $O(n_1 m T^2 + n_1^3 T^3)$.*

The proof of this Theorem is presented in the papers [19] and [20].

We present an example for determining the maximum flow in bipartite dynamic network with lower bound.

The support digraph of the bipartite dynamic network is presented in Figure 9 and time horizon being set $T = 5$, therefore $H = \{0, 1, 2, 3, 4, 5\}$. The transit times $h(i, j; t) = h(i, j), t \in H$, the lower bounds $e(i, j; t) = e(i, j)$ and the upper bounds (capacities) $q(i, j; t) = q(i, j), t \in H$ for all arcs are indicated in Table 5.

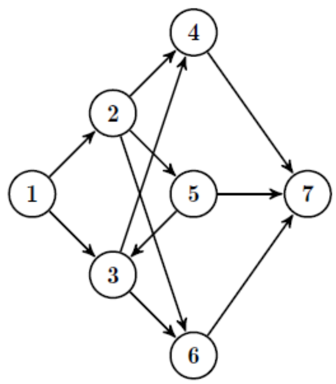

Figure 9: The support digraph of network $D = (N, A, h, e, q, H)$

(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
$e(i,j)$	3	5	1	1	1	0	4	1	0	1	5
$q(i,j)$	12	10	8	3	3	4	5	12	3	4	10

Table 5: The functions h, e, q

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

Applying the GBFIFOP2 algorithm in the first phase and the second phase we obtain the flows $f_0(i_t, j_\theta)$, $f_0^*(i_t, j_\theta)$ (the feasible flow, the maximum flow) which are indicated in Figure 10. 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 $[Y_0, \bar{Y_0}] = (Y_0, \bar{Y_0}) \cup (\bar{Y_0}, Y_0)$ 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 $[Y_0, \bar{Y_0}] = \{(1_0, 2_1), (2_2, 5_3), (2_2, 6_4), (2_3, 5_4), (3_1, 6_2), (3_1, 4_4), (3_2, 6_3)\} \cup \{(5_2, 3_3)\}$. We have $\bar{w}_0 = f_0^*(Y_0, \bar{Y_0}) - f_0^*(\bar{Y_0}, Y_0) = 40 - 0 = 40 = u_0(Y_0, \bar{Y_0})$. Hence f_0^* is a maximum flow.

4 Conclusions

In this paper we have presented algorithms for maximum flow problem in bipartite dynamic networks with lower bounds zero using static approach and dynamic approach and algorithms for maximum flow problem in bipartite dynamic networks with lower bounds positive. We have demonstrated the fact that if the dynamic network $D = (N, A, h, q)$ is bipartite, then the static reduced expanded

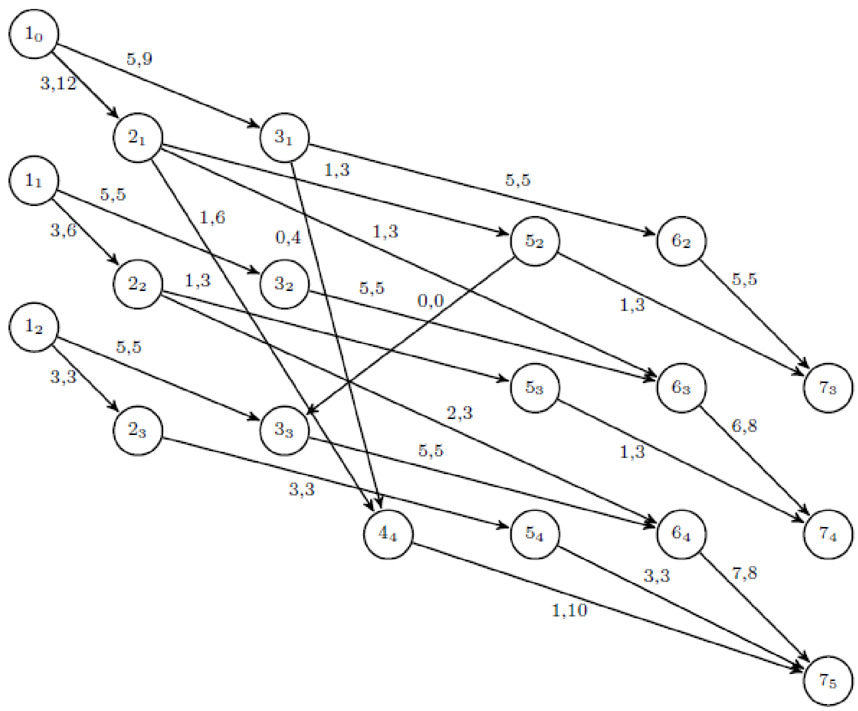

Figure 10: The network $R_0 = (V_0, E_0, f_0, f_0^*)$.

network $R_0 = (V_0, E_0, u_0)$ is bipartite. Therefore, we solved the problems in bipartite dynamic networks by rephrasing into a problem in bipartite static network. We have extended the bipartite FIFO preflow algorithm of Ahuja et al. [2] to 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. For each of the problems mentioned above, we present an example for the clarity of the 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] Ahuja, R., Magnanti, T. and Orlin, J. (1993) *Network flows. Theory, algorithms and applications*, Prentice Hall, Inc., Englewood Clifss, New Jersey

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

[3] Aronson, J. E., *A survey of dynamic network flows*, Annals of Operation Research, **20:1-66**, (1989).

[4] Cai, X., Sha, D. and Wong, C. *Time-varying Network Optimization*, Springer, (2007).

[5] Ciurea, E. and Ciupală, L. (2004) *Sequential and parallel algorithms for minimum flows*, Journal of Applied Mathematics and Computing, vol. **15**, no. 1-2, 53–75.

[6] Ciurea, E. *An algorithm for minimal dynamic flow*, Korean Journal of Computaional and Applied Mathematics **7**, 2, (2000), 259-270.

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

[8] Dinic, E. *Algorithm for solution of a problem of maximum flow in network with power estimation*, Soviet Mathematics Doklady **11** (1970), 1277–1280.

[9] Ford, L. and Fulkerson, D. (1962) *Flows in Networks*, Princenton University Press, Princenton, New Jersey.

[10] Ford, L. and Fulkerson, D. *Maximal flows throgh a network*, Canadian Journal Mathematics **8** (1956), 399–404.

[11] Ciurea, E., Georgescu, O. and Marinescu, D. *Improved algorithms for minimum flows in bipartite networks*, it International Journal of Computers, vol. 2, no. 4, (2008), 351–360.

[12] Georgescu, O. and Ciurea, E. *Decreasing path algorithm for minimum flow. Dynamic tree implementation*, Proceedings of the 12st WSEAS International Conference on Computers, (2008), pp. 235–240.

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

[14] Hamacher, H.W., Tjandra, S.A., *Mathematical modelling of evacuation problems: a state of the art*, Berichte des Fraunhofer ITWM, no. 24 (2001).

[15] Jungnickel, D., *Graphs, networks and algorithms*, Springer, Berlin, (1999).

[16] Karzanov, A. and Naor, J., *Determining the maximal flow in a network by the method of preflows.* Algorithmica **15** (1974), 434-437.

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

[18] Schiopu, C., *The maximum flows in bipartite dynamic networks. The static approach*, Annals of the University of Craiova, Mathematics and Computer Science Series, **43**, no. 2, (2016), 200-209.

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

[20] Schiopu, C. and Ciurea, E. *Two flow problems in dynamic networks*, International Journal of Computers Communications & Control **12(1)** (2017), 1, 103–115.

[21] Skutella, M. *An Introduction to Network Flows Over Time*, In book: Research Trends in Combinatorial Optimization, (2009), 451-482.

[22] Tjandra, S.A., *Dynamic Network Optimization with Application to the Evacuation Problem*, Dissertation, Universitat Kaiserslautern, 2003.

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