INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING
DOI: 10.46300/9106.2020.14.43 Volume 14, 2020

Optimum Flows in Directed Bipartite Dynamic Network. The Static Approach.

Camelia Schiopu and Eleonor Ciurea

Received: April 8, 2020. Revised: July 10, 2020. Accepted: July 13, 2020. Published: July 14, 2020

Abstract- The theory of flows is one of the most important parts of Combinatorial Optimization and it has various applications. In this paper we study optimum (maximum or minimum) flows in directed bipartite dynamic network and is an extension of article [9]. In practical situations, it is easy to see many time-varying optimum problems. In these instances, to account properly for the evolution of the underlying system overtime, we need to use dynamic network flow models. When the time is considered as a variable discrete values, these problems can be solved by constructing an equivalent, static time expanded network. This is a static approach.

Keywords- static network flow, dynamic network flow, bipartite network.

I. Introduction

THE optimum (maximum or minimum) flows problem seeks a feasible solution that sends the optimum amount of flow from a specified source node to another specified sink node. The general static network flow models arises in a number of combinatorial application as far reaching as machine scheduling, the assignment of computer modules to computer processor, tanker scheduling etc. [1], [2]. On the other hand, the bipartite static network also arises in practical context such baseball elimination problem, network reliability testing etc. and hence it is of interest to find fast flow algorithms for this class of networks [3], [4], [5].

In some other applications, the time is an essential ingredient. In these instances we need to use dynamic network flow models [1], [6], [7], [2], [8].

In this paper, we present the optimum (maximum or minimum) flow problem in bipartite dynamic networks with static approach and is an extension of paper [9]. Further on, in Section 2, Section 3, and Section 4 are presented some basic static and dynamic network notations, terminology and results which are necessarily in Section 5, where we present optimum flow in directed bipartite dynamic networks into a static approach. In Section 6 two examples are given and in Section 7 are presented some conclusions.

The problem of optimum flow in bipartite dynamic networks is treated so far only only by the authors of this paper.

II. Optimum flows in general static networks

The notions and the results presented in this section are taked over from works [1], [10], [2].

We consider a connected, antisymmetric graph $G = (N, A)$ without loops with the set of nodes $N = \{1, \ldots, i, \ldots, j, \ldots, n\}$, the set of arcs $A = \{a_1, \ldots, a_k, \ldots, a_m\}$, $a_k = (i, j)$, $i, j \in N$. Let $S = (N, A, l, u, 1, n)$ be a static network with the lower bound function $l : A \to \mathbb{N}$, the upper bound (capacity) function $u : A \to \mathbb{N}$, with $\mathbb{N}$ the natural number set, 1 the source node and n the sink node.

For a given pair of not necessarily disjoint subset $X \subset N$, $Y \subset N$ we use the notation $(X,Y)=\{(i,j)|(i,j) \in A, i \in X, j \in Y\}$ and for a given function $f : A \to \mathbb{N}$ we use the notation $f(X,Y) = \sum_{(X,Y)} f(i,j)$.

A flow vector f is a feasible flow vector in S if it satisfies the following constraints:

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

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

with the value of the flow $v \geq 0$ and $f(i,j) = 0$ for the pairs $(i,j) \notin A$

The optimum (maximum or minimum) flow problem consist in determining a flow f for which v is optimized (maximized or minimized).

We specify that if function f verifies (1a) then f is a flow and if verifies and (1b) f is a feasible flow.

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

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

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

The excess of each node $i \in N - \{1, n\}$ for a preflow f 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 node i is an active node. Similarly, for minimum flow problem, a preflow f is a function $f : A \to \mathbb{N}$ satisfying the next conditions:

$$f(i, N) - f(N, i) \leq 0, i \in N - \{1, n\} \tag{4a}$$

$$l(i, j) \leq f(i, j) \leq u(i, j), (i, j) \in A \tag{4b}$$

The deficit of each node $i \in N - \{1, n\}$ for a preflow is

$$e(i) = f(i, N) - f(N, i) \tag{5}$$

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

For an optimum flow problem we refer to a node i with $e(i) = 0$ as balanced. A preflow f satisfying the condition $e(i) = 0, i \in N - \{1, n\}$ is a flow. Thus, a flow is a particular case of preflow.

A cut is a partition of nodes set N into proper subset X and $Y = N - X$. We represent this cut using notation $[X, Y]$. An arc (i, j) with $i \in X, j \in Y$ is a forward arc of the cut and an arc (j, i) with $i \in X$ and $j \in Y$ is a backward arc of the cut. Let (X, Y) denote the set of forward arcs and let (Y, X) denote the set of backward arcs in the cut. We have $[X, Y] = (X, Y) \cup (Y, X)$. We refer to a cut as a $1 - n$ cut if $1 \in X$ and $n \in Y$.

Whereas the optimum flow problem with zero lower bounds always has a feasible solution (since the zero flow is feasible), the problem with positive lower bounds could be infeasible. Any optimum flow algorithm for problems with non-negative lower bounds has two phase:

(1) to determine a feasible flow if one exists;

(2) convert a feasible flow into an optimum flow.

We consider the network $S' = (N', A', l', u', f')$ with $N' = N$, $A' = A \cup \{(n, 1)\}$, $l'(i, j) = l(i, j)$ $(i, j) \in A$, $l'(n, 1) = 0$, $u'(i, j) = u(i, j)$, $(i, j) \in A$ $u'(n, 1) = \infty$, $f'(i, j) = f(i, j)$, $(i, j) \in A$, $f'(n, 1) = v$. Because $l'(n, 1) < u'(n, 1)$ and $N' = N$ the relation $l'(Y', X') \leq u'(X', Y')$ is equally with the relation $l(Y, X) \leq u(X, Y)$. Thus, we transform the flow problem into a circulation problem by adding the arc $(n, 1)$ to network S. This arc carries the flow send node 1 to node n back to node 1. Clearly, the optimum flow problem admits a feasible flow if and only if the circulation problem admits a feasible flow.

The feasible circulation problem is to identify a flow f' satisfying the following constraints:

$$f'(i, N') - f'(N', i) \leq 0, i \in N' \tag{6a}$$

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

Theorem 1. *Circulation Feasibility Conditions A circulation problem with nonnegative lower bounds is feasible if and only if for every set X of nodes $l(Y, X) \leq u(X, Y)$, $Y = N - X$.*

For the maximum flow problem, we define the capacity $K[X, Y]$ of the $1 - n$ cut $[X, Y]$ as:

$$K[X, Y] = u[X, Y] - l[Y, X] \tag{7}$$

and for the minimum flow problem, we define the capacity $k[X, Y]$ of the $1 - n$ cut $[X, Y]$ as:

$$k[X, Y] = l[Y, X] - u[X, Y] \tag{8}$$

For the maximum flow problem we refer to a $1 - n$ cut which has the minimum capacity among all $1 - n$ cuts as a minimum cut and for the minimum flow problem we refer a $1 - n$ cut which has the maximum capacity among among all $1 - n$ cuts as a maximum cut.

Theorem 2. *Max-Flow Min-Cut Theorem If there exists a feasible flow in the network $S = (N, A, l, u, 1, n)$, the value of the maximum flow from a source node 1 to a sink node n is equal with the capacity of the minimum $1 - n$ cut.*

Theorem 3. *Min-Flow Max-Cut Theorem If there exists a feasible flow in the network $S = (N, A, l, u, 1, n)$, the value of the minimum flow from a source node 1 to a sink node n is equal with the capacity of the maximum $1 - n$ cut.*

The concept of residual network plays a central role in the development of all the optimum flow algorithms. Residual capacity of an arc (i, j) for maximum flow problem is:

$$r(i, j) = u(i, j) - f(i, j) + f(j, i) - l(j, i) \tag{9}$$

and for minimum flow problem is:

$$r(i, j) = u(j, i) - f(j, i) + f(i, j) - l(i, j) \tag{10}$$

We refer to the network $\tilde{S} = (N, \tilde{A}, r, 1, n)$ with $\tilde{A} = \tilde{A}^+ \cup \tilde{A}^-$, $\tilde{A}^+ = \{(i, j) | (i, j) \in A\}$ and $f(i, j) < u(i, j)$, $\tilde{A}^- = \{(j, i) | (i, j) \in A\}$ and $f(i, j) > l(i, j)$ as the residual network (with respect to the flow f).

The optimum flow algorithms works with only residual capacities. These algorithms end with optimal residual capacities. From these residual capacities we can construct optimum flow. For maximum flow we have:

$$f(i, j) = l(i, j) + max\{0, u(i, j) - r(i, j) - \\ -l(i, j)\}, (i, j) \in A \tag{11}$$

and for the minimum flow we have:

$$f(i, j) = l(i, j) + max\{0, r(i, j) - u(j, i) + \\ +l(j, i)\}, (i, j) \in A \tag{12}$$

Clearly, the present results in this section are valid and for bipartite networks.

The reader interested in further details is urged to consult the works [1], [10], [2].

III. Optimum flows in general dynamic networks

The notions and the results presented in this section are take over from works [6], [7], [2], [8], [11], [12], [13].

Most optimum flow models considered in the literature are static, in which the lower bounds and the upper bounds of the arcs are assumed to be constant. In

practical situations, it is easy to see many time-varying optimum flow problems.

Let $D = (N, A, H, h, l, u, 1, n)$ be a general dynamic network, where $N = \{1, \ldots, i, \ldots, j, \ldots, n\}$ is the set of nodes, $A = \{a_1, \ldots, a_k, \ldots, a_m\}$ is the set of arcs, $H = \{0, 1, \ldots, t, \ldots, T\}$ is the set of periods with $T \in \mathbb{N}$, $h : A \times H \to \mathbb{N}$ is the transit time function, $l : A \times H \to \mathbb{N}$ is the time lower bound function, $u : A \times H \to \mathbb{N}$ is the time upper bound function, 1 the source node and n the sink node.

The optimum (maximum or minimum) dynamic flow problem for T time periods is to determine a dynamic flow function $f : A \times H \to \mathbb{N}$, which should satisfy the following conditions in dynamic network D:

$$\sum_{t=0}^{T}(\sum_{j} f(1, j; t) - \sum_{k}\sum_{\tau} f(k, 1; \tau)) = v_d \tag{13a}$$

$$\sum_{j} f(i, j; t) - \sum_{k}\sum_{\tau} f(k, i; \tau) = 0, i \neq 1, n,\ t \in H \tag{13b}$$

$$\sum_{t=0}^{T}(\sum_{j} f(n, j; t) - \sum_{k}\sum_{\tau} f(k, n; \tau)) = -v_d \tag{13c}$$

$$l(i, j; t) \leq f(i, j; t) \leq u(i, j; t), (i, j) \in A, t \in H \tag{14}$$

$$opt \quad v_d, \tag{15}$$

where $\tau = t - h(k, i; t)$, $v_d = \sum_{t=0}^{T} v(t)$, $v(t)$ is the flow value at time t, $f(i, j; t) = 0$ for all $t \in \{T - h(i, j; t) + 1, \ldots, T\}$ and opt is max or min.

In the most general dynamic model, the parameter $h(i) = 1$ is waiting time at node i and the parameters $l(i; t)$, $u(i; t)$ are lower bound and upper bound for flow $f(i; t)$ that can wait at node i from time t at $t + 1$. This most general dynamic model is not discussed in this paper.

Obviously, the problem of finding an optimum flow in dynamic network $D = (N, A, H, h, l, u, 1, n)$ is more complex than the problem of finding an optimum flow in static network $S = (N, A, l, u, 1, n)$. Happily, this complication can be resolved, through static approach, by rephrasing the problem in dynamic network D into a problem in static network $R_0 = (V_0, E_0, l_0, u_0)$ with multiple source nodes and multiple sink nodes. The network R_0 can be obtained by two methods:

1. using static shortcut path [1];
2. using dynamic shortcut path [6].

We present only second method [14].

Let $d(1, i; t)$ be the length of the dynamic shortcut 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 $R_0 = (V_0, E_0, l_0, u_0)$ has $V_0 = \{i_t | i \in N, t \in H_i\}$, $E_0 = \{(i_t, j_\theta) | (i, j) \in A, t \in H_{i,j}\}$ $l_0(i_t, j_\theta) = l(i, j; t)$, $u_0(i_t, j_\theta) = u(i, j; t)$, $(i_t, j_\theta) \in E_0$.

The optimum flow problem for T time periods in the dynamic network D formulated in conditions (13), (14), (15) is equivalent with the optimum flow problem in static reduced expanded network R_0 as follows:

$$f_0(i_t, V_0) - f_0(V_0, i_t) = \begin{cases} v_t, & \text{if } i_t = 1_t,\ t \in H_1 \\ 0, & \text{if } i_t \neq 1_t, n_t, t \in H_1, \\ & \qquad t \in H_n \\ -v_t, & \text{if } i_t = n_t,\ t \in H_n \end{cases} \tag{16}$$

$$l_0(i_t, j_\theta) \leq f_0(i_t, j_\theta) \leq u_0(i_t, j_\theta), \quad (i_t, j_\theta) \in E_0 \tag{17}$$

$$opt \sum_{H_1} v_d, \tag{18}$$

If h, l, u are constant over time, then a dynamic network D is said to be stationary. In this case, the dynamic distances $d(1, i; t)$, $d(i, n; t)$ become static distances $d(1, i)$, $d(i, n)$.

Many notions and results are presented in [1], [6], [7], [2].

IV. Optimum flows in bipartite static networks

In this section we consider that static network $S = (N, A, l, u, 1, n)$ 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, for the maximum flow problem we assume that $n_1 \leq n_2$ and for the minimum flow problem that $n_2 \leq n_1$. Also we assume that $1 \in N_2$ and $n \in N_1$. A bipartite network is called unbalanced if $n_1 << n_2$ for the maximum flow problem and $n_2 << n_1$ for the minimum flow problem. Otherwise the bipartite network is called balanced.

In reference [4], [5] the authors show that time bounds for several optimum flow algorithms automatically improves when the algorithms are applied without modification to unbalanced networks. A carefully 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 by L. For general network, $L \leq n - 1$ and for a bipartite network $L \leq 2n_1 + 1$ for maximum flow problem, $L \leq 2n_2 + 1$ for minimum flow problem. The columns 3 and 5 of Table 1 summarizes these improvements for several network flow algorithms.

The authors of references [3], [4] obtain further running time improvements by modifying the algorithms.

Algorithm	*Running time, general network (optimum)*	*Running time, bipartite network (maximum)*	*Running time modified version (maximum)*	*Running time bipartite network (minimum)*	*Running time modified version (minimum)*
Dinic	n^2m	n_1^2m	*does not apply*	n_2^2m	*does not apply*
Karazanov	n^3	n_1^2n	$n_1m+n_1^3$	n_2^2n	$n_2m+n_2^3$
FIFO preflow	n^3	n_1^2n	$n_1m+n_1^3$	n_2^2n	$n_2m+n_2^3$
Highest label	$n^2\sqrt{m}$	$n_1n\sqrt{m}$	n_1m	$n_2n\sqrt{m}$	n_2m
Excess scaling	$nm+n^2\log\bar{u}$	$n_1m+n_1n\log\bar{u}$	$n_1m+n_1^2\log\bar{u}$	$n_2m+n_2n\log\bar{u}$	$n_2m+n_2^2\log\bar{u}$

Table 1: Several optimum flows algorithms

This modifications applies only to preflow algorithms. The columns 4 and 6 of Table 1 summarizes the improvements obtained using this approach. In Table 1 we use the notation $\bar{u}$. This notation have value $\bar{u} = max\{u(i,j)|(i,j) \in A\}$ [15].

The reader interested in further details is urged to consult the papers [3], [4], [5].

V. Optimum flows in bipartite dynamic networks

In this section the dynamic network $D = (N, A, H, h, l, u, 1, n)$ is bipartite.

We construct the static reduced expanded network $R_0 = (V_0, E_0, l_0, u_0)$ with $V_0 = W_1 \cup W_2$, $W_1 = \{i_t|i \in N_1, t \in H_i\}$, $W_2 = \{i_t|i \in N_2, t \in H_i\}$.

In paper [16] the next theorem is proved.

Theorem 4. *If the dynamic network $D = (N, A, H, h, l = 0, u, 1, n)$ is bipartite, then the static reduced expanded network $R_0 = (V_0, E_0, l_0 = 0, u_0)$ is bipartite.*

From this theorem results: if the dynamic network $D = (N, A, H, h, l > 0, u, 1, n)$ is bipartite, then the static reduced expanded network $R_0 = (V_0, E_0, l_0 > 0, u_0)$ is bipartite.

Let w_1, w_2, e_0 be $w_1 = |W_1|$ $w_2 = |W_2|$, $e_0 = |E_0|$. If $n_2 << n_1$ then obvious that $w_2 << w_1$ and 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 applied into general static network S for a maximum flow f 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 terminates when the queue Q of active nodes is empty. The reader interested in further details is urged to consult the book [1].

The modified version of FIFO preflow algorithm for maximum flow in bipartite static network S 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 $\tilde{P} = (i, j, k), j \in N_2$, and ensures that no node in N_2 ever has any deficit. A push of α units from node i to node j decreases both $e(i)$ and $r(i,j)$ by α units and increases both $e(j)$ and $r(j,i)$ by α units. In the paper [3] is presented the bipartite preflow for maximum flow (BFIFOPM) algorithm. We remark the fact that this algorithm runs on network $S = (N, A, l = 0, u, 1, n)$.

In the paper [14] is presented the generalized of BFIFOPM algorithm 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, $t \in H_n$. This algorithm is presented below.

```
1: ALGORITHM GBFIFOPM;
2: BEGIN
3: PREPROCESS;
4: while Q ≠ ∅ do
5:    select the node i_t from the front of Q;
6:    BIPUSH/RELABEL(i_t);
7: end while;
8: END.
```

```
1: PROCEDURE PREPROCESS;
2: BEGIN
3: f_0 is a feasible flow in R_0; Q := ∅;
4: compute the exact distance labels d(i_t);
5: for t ∈ H_1 do
6:    f_0(1_t, j_θ) := u_0(1_t, j_θ) and adds node j_θ to the rear
      of Q for all (1_t, j_θ) ∈ E_0
7:    d(1_t) := 2w_1 + 1;
8: end for;
9: END;
```

```
1: PROCEDURE BIPUSH/RELABEL(i_t);
2: 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**
    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**
      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;
    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
        $d(j_\theta) := min\{d(k_\tau) + 1 | (j_\theta, k_\tau) \in E_0^+(j_\theta), r_0(j_\theta, k_\tau) > 0\}$;
      end if;
    end if;
    if $e(i_t) > 0$ **then**
      if (i_t, j_θ) is not the last arc in $E_0^+(i_t)$ with $r_0(i_t, j_\theta) > 0$ **then**
        select the next arc in $E_0^+(i_t)$;
      else
        $d(i_t) := min\{d(j_\theta) + 1 | (i_t, j_\theta) \in E_0^+(j_\theta), r_0(i_t, j_\theta) > 0\}$;
        $\beta := 0$;
      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;

We remark that any simple path $P_{1_t n_\theta}$ 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. Also, we specify that in the first phase the feasible flow f_0 is zero flow and in the second phase the feasible flow f_0 is the feasible flow f_0 determined in the first phase.

The correctness of the GBFIFOPM algorithm results from correctness of the BFIFOPM algorithm. From paper [14] we present following theorem.

Theorem 5. *The GBFIFOPM algorithm has the complexity* $O(n_1 m T^2 + n_1^3 T^3)$.

Now, present the minimum flow problem in bipartite dynamic network $D = (N, A, H, h, l, u, 1, n)$.

The modified version of FIFO preflow algorithm for minimum flow in bipartite static network $S = (N, A, l, u, 1, n)$ is called bipartite FIFO preflow for minimum flow (BFIFOPm) algorithm. A bipull is a pull over two consecutive admissible arcs. It moves deficit from a node $j \in N_2$ to another node $k \in N_2$. This approach means that the algorithm moves the flow over the back path $\tilde{P} = (j, i, k), i \in N_1$, and ensures that no node in N_1 ever has any deficit. A pull of α units from node j to node i decreases both $e(i)$ and $r(i, j)$ by α units and increases both $e(j)$ and $r(j, i)$ by α units. In the paper [4] is presented the BFIFOPm algorithm.

In the paper [9] is presented the generalized of BFIFOPm algorithm for a network $R_0 = (V_0, E_0, l_0, u_0)$. This algorithm is presented below.

ALGORITHM GBFIFOPm;
BEGIN
PREPROCESS
while $Q \neq \emptyset$ **do**
  select the node j_θ from the front of Q;
  BIPULL/RELABE(j_θ);
end while
END.

PROCEDURE PREPROCESS;
BEGIN
f_0 is a feasible flow in R_0; $Q := \emptyset$;
compute the exact distance function $\hat{d}$ in residual network $\hat{G}$;
for $\lambda \in H_n$ **do**
  $f_0(j_\theta, n_\lambda) := l_0(j_\theta, n_\lambda)$ and adds node j_θ to the rear of Q for all $(j_\theta, n_\lambda) \in E_0$
  $\hat{d}(n_\lambda) := 2w_2 + 1$;
end for
END.

PROCEDURE BIPULL/RELABEL(j_θ);
BEGIN
select the first arc (i_t, j_θ) in $E_0^-(j_\theta)$ with $\hat{r}_0(i_t, j_\theta) > 0$;
$\beta := 1$;
repeat
  if (i_t, j_θ) is admissible arc **then**
    select the first arc (k_τ, i_t) in $E_0^-(i_t)$ with $\hat{r}_0(k_\tau, i_t) > 0$;
    if (k_τ, i_t) is admissible arc **then**
      pull $\alpha := min\{-e(j_\theta), \hat{r}_0(i_t, j_\theta), \hat{r}_0(k_\tau, i_t)\}$ units of flow over the arcs $(i_t, j_\theta), (k_\tau, i_t)$;
      if $k_\tau \notin Q$ **then**
        adds node k_τ to the rear of Q;
      end if;
    else
      if (k_τ, i_t) is not the last arc in $E_0^-(i_t)$ with $\hat{r}_0(k_\tau, i_t) > 0$ **then**
        select the next arc in $E_0^-(i_t)$
      else
        $\hat{d}(i_t) := min\{d(k_\tau) + 1 | (k_\tau, i_t) \in E_0^-(i_t), \hat{r}_0(k_\tau, i_t) > 0\}$
      end if;
    end if;
    if $e(j_\theta) > 0$ **then**
      if (i_t, j_θ) is not the last arc in $E_0^-(j_\theta)$ with $\hat{r}_0(i_t, j_\theta) > 0$ **then**
        select the next arc in $E_0^-(j_\theta)$
      else
        $\hat{d}(j_\theta) := min\{d(i_t) + 1 | (i_t, j_\theta) \in E_0^-(j_\theta),$

$\hat{r}_0(i_t, j_\theta) > 0\}$
$\beta := 0$;
end if;
end if;
end if;
until $e(j_\theta) = 0$ or $\beta = 0$
if $e(j_\theta) > 0$ **then**
adds node j_θ to the rear of Q;
end if;
END;

We remark that in this case, the correctness of the GBFIFOPm algorithm results from correctness of the BFIFOPm algorithm. From paper [9] we present following theorem.

Theorem 6. *The GBFIFOPm algorithm has the complexity* $O(n_2mT^2 + n_2^3T^3)$.

For more information see [9].

VI. Examples

A. Maximum flow

In this case we consider a bipartite dynamic network D with $n_1 < n_2$.

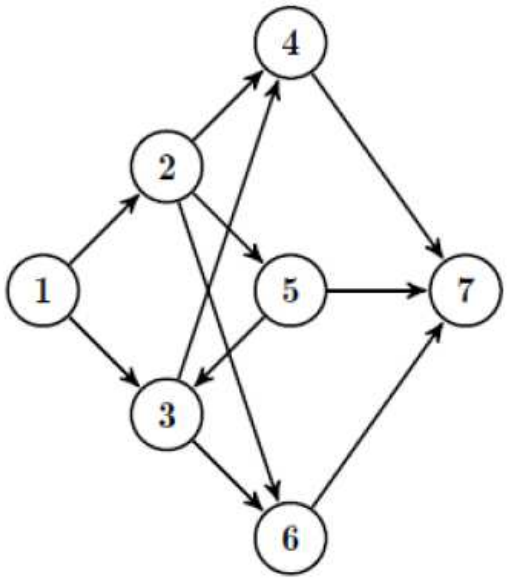

Fig. 1: The support digraph of network $D = (N, A, h, e, q)$

The support digraph of bipartite dynamic network is presented in Figure 1 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 indicate in Table 2.

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

Applying the GBFIFOPM 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 2. 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 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 $\bar{w}_0 = f_0^*(Y_0, \bar{Y_0}) - f_0^*(\bar{Y_0}, Y_0) = 40\text{-}0 = 40 = u_0(Y_0, \bar{Y_0})$. Hence f_0^* is a maximum flow.

(i, j)	$h(i, j)$	$e(i, j)$	$q(i, j)$
$(1, 2)$	1	3	12
$(1, 3)$	1	5	10
$(2, 4)$	3	1	8
$(2, 5)$	1	1	3
$(2, 6)$	2	1	3
$(3, 4)$	3	0	4
$(3, 6)$	1	4	5
$(4, 7)$	1	1	12
$(5, 3)$	1	0	3
$(5, 7)$	1	1	4
$(6, 7)$	1	5	10

Table 2: The functions h, e, q

B. Minimum flow

In this case we consider a bipartite dynamic network D with $n_2 < n_1$.

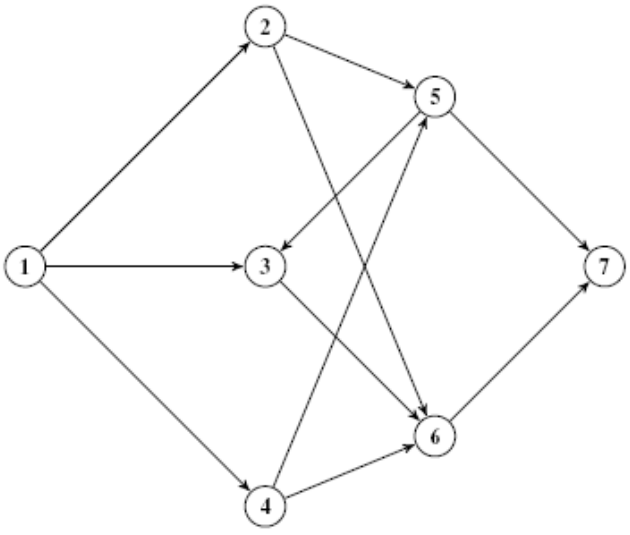

Fig. 3: The support digraph of network $D = (N, A, h, q)$

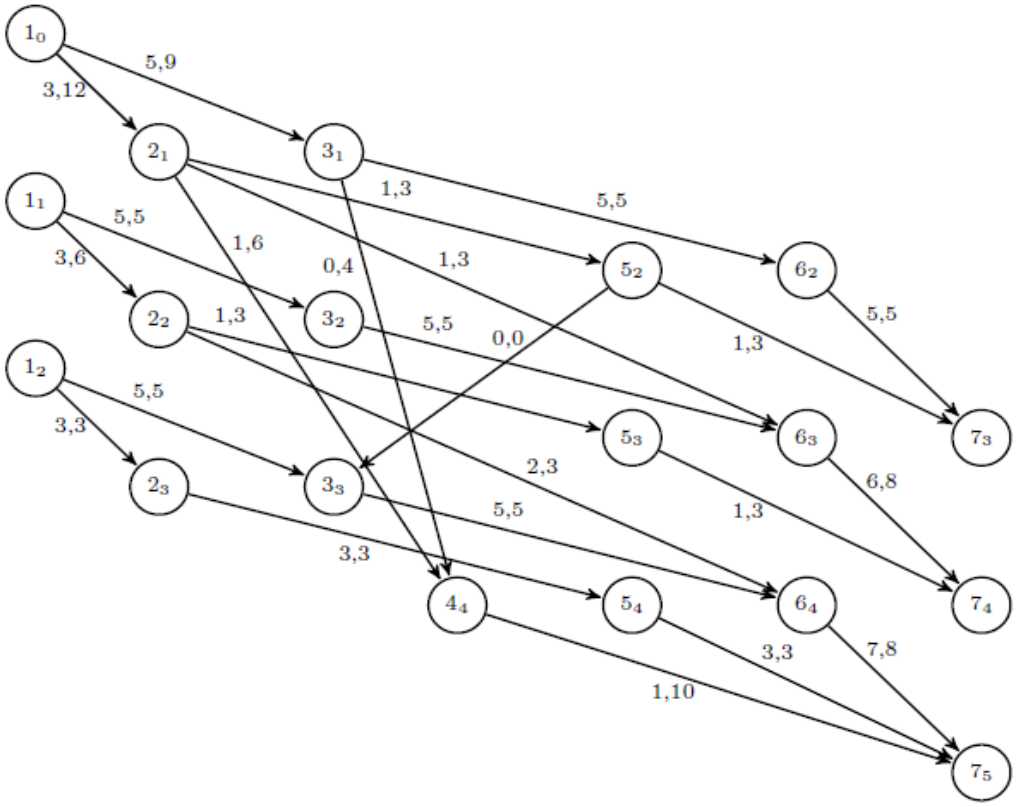

Fig. 2: The network $R_0 = (V_0, E_0, f_0, f_0^*)$.

The support digraph of the bipartite dynamic network is presented in Figure 3 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 indicate in Table 3.

(i, j)	$h(i, j)$	$e(i, j)$	$q(i, j)$
$(1, 2)$	1	5	8
$(1, 3)$	2	1	3
$(1, 4)$	1	2	5
$(2, 5)$	1	3	6
$(2, 6)$	2	2	3
$(3, 6)$	1	1	5
$(4, 5)$	2	1	3
$(4, 6)$	3	1	3
$(5, 3)$	1	2	3
$(5, 7)$	2	0	7
$(6, 7)$	1	3	9

Table 3: The functions h, e, q

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

Applying the GBFIFOPm algorithm in the first phase and the second phase we obtain the flows $f_0(i_t, j_\theta)$, $f_0^*(i_t, j_\theta)$ (feasible flow, minimum flow) which are indicated in Figure B.. We have W_1=$\{2_1,\ 2_2,\ 3_2,\ 3_3,\ 4_1,\ 7_4,\ 7_5\}$ and W_2=$\{1_0,\ 1_1,\ 5_2,\ 5_3,\ 6_3,\ 6_4\}$. A maximum $(1_0, 1_1)$-$(7_4, 7_5)$ 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=$\{1_0,1_1,2_1,2_2\}$ and $\bar{Y_0}$=$\{3_2,3_3,4_1,5_2,5_3,6_3,6_4,7_4,7_5\}$. Hence $\left[Y_0, \bar{Y_0}\right]$= $\{(1_0, 3_2),(1_0, 4_1),(1_1, 3_3),(2_1, 5_2),(2_1, 6_3),(2_2, 5_3),(2_2, 6_4)\} \cup \varnothing$. We have $\bar{w}_0$=$f_0^*(Y_0, \bar{Y_0})$ - $f_0^*(\bar{Y_0},\ Y_0)$=$14 - 0 = 14$ and $l_0(Y_0, \bar{Y_0}) - u_0(\bar{Y_0}, Y_0) = 14 - 0 = 14$. Hence, f_0^* is a minimum flow.

VII. Conclusions

In this paper, we present the optimum (maximum or minimum) flow problem in bipartite dynamic networks with static approach. This problem has not been treated so far. We demonstrate the fact that if the dynamic network $D = (N, A, H, h, l, u)$ is bipartite, then the static reduced expanded network. $R_0 = (V_0, E_0, l_0, u_0)$ is bipartite. Therefore we solved the optimum flow problem in bipartite dynamic network D by rephrasing into a problem in bipartite static network R_0. We have extended the bipartite FIFO preflow algorithm (Ahuja et al., 1994) for the static reduced expanded network R_0 which is a network with $l_0 > 0$ and multiple sources and multiple sinks. For the generalization bipartite FIFO preflow algorithm we have presented the complexity.

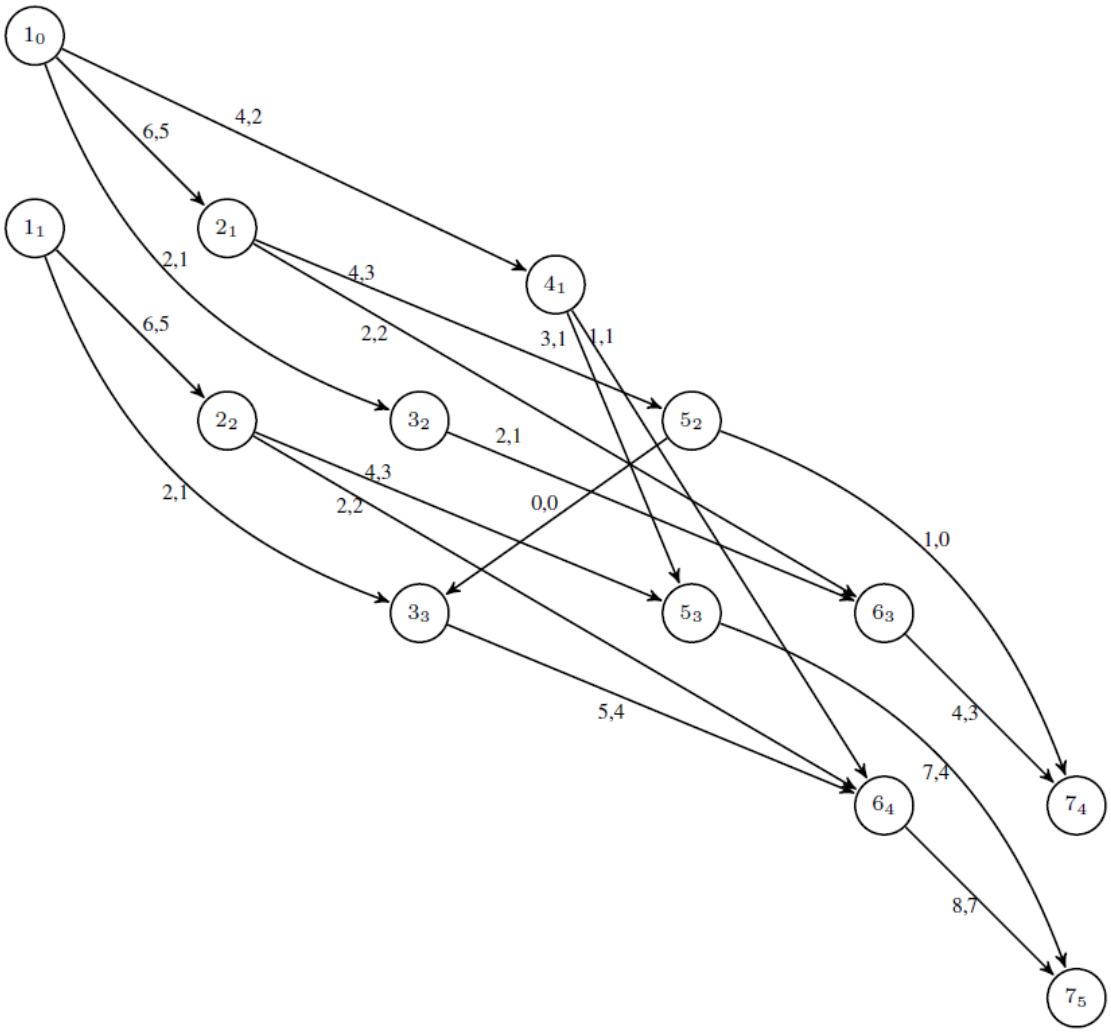

Fig. 4: The network $R_0 = (V_0, E_0, u_0)$.

Many interesting flow problems in bipartite dynamic network are still open: optimum flow problem in bipartite dynamic networks with dynamic approach, the generalization other algorithms for optimum flow in bipartite static networks. 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] Ford, L. and Fulkerson, D. (1962) *Flows in Networks*, Princenton University Press, Princenton, New Jersey.

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

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

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

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

[7] Ciurea, E. (2000) An algorithm for minimal dynamic flow, *Korean Journal of Computational and Applied Mathematics*, vol. 7, no.2, pp. 259–270.

[8] Salehi, H. F., Khadayifar, S. and Raayatpanoh, M.A. (2012) Minimum flow problem on network flows with time-varying bounds, *Applied Mathematical Modelling*, vol. 36, pp. 4414-4421.

[9] Schiopu, C. and Ciurea, E. (2020) The minimum flows in bipartite dynamic networks. The static approach, *International Conference on Mathematics and Computers in Science and Engineering*, IEEE Xplore, 2020, Madrid, Spain.

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

pp. 53–75.

[11] Schiopu, C. and Ciurea, E. (2017) Two flow problems in dynamic networks, *International Journal of Computers Communications & Control*, vol. 12, no. 1, pp. 103-115.

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

[13] Wilkinson, W. (1971) An Algorithm for Universal maximal Dynamic Flows in a Network, *Oper. Res.*, vol. 19, pp. 1602–1612.

[14] Schiopu, C. and Ciurea, E. (2016) The maximum flow in bipartite dynamic networks with lower bounds. The static approach, *International Conference on Computers Communications and Control*, IEEE Xplore, Băile Felix - Oradea, pp. 10-15.

[15] Schiopu, C. (2014) The maximum flows in bipartite dynamic networks, *Bulletin of the Transilvania University of Brasov, Series III: Mathematics, Informatics, Physics*, vol. 7(56), no. 2, pp. 193-202.

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