INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 12(1):103-115, February 2017.

Two Flow Problems in Dynamic Networks

C. Schiopu, E. Ciurea

Camelia Schiopu*, Eleonor Ciurea
Transilvania University of Braşov
Romania, 500091 Braşov, Iului Maniu, 50
camelia.s@unitbv.ro, e.ciurea@unitbv.ro
*Corresponding author: camelia.s@unitbv.ro

Abstract: In this paper we study two flow problems: the feasible flow problem in dynamic networks and the maximum flow problem in bipartite dynamic networks with lower bounds. We mention that the maximum flow problem in bipartite dynamic networks with lower bound was presented in paper [8][a]. For these problems we give examples.
Keywords: dynamic networks, feasible flow, bipartite network, maximum 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 problem 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 the dynamic network flow model. 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 [1], [6].

The maximum flow problem in bipartite dynamic networks with lower bounds was presented in paper [8]. In this paper, which is an extension of [8], we present in addition the feasible flow problem in dynamic networks. These problem have not been treated so far. Further on, in Section 2.1 we discuss some basic notions and results for the maximum flow problem in general static networks with lower bounds. Section 2.2 deals with the maximum flows problem in general dynamic networks with lower bounds. In Section 2.3 we present algorithms for flow problems in bipartite static network and in Section 2.4 we discuss the feasible flows in static networks. In Section 3 we discuss the feasible flows problem in dynamic network and give some examples. Section 4 deals with maximum flows in bipartite dynamic networks with lower bounds and an example.

2 Terminology and Preliminaries

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

2.1 Maximum flows in static networks with lower bounds

Let $G = (N, A, l, u)$ be a 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 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 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:

$$f(X, Y) = \sum_{(X,Y)} f(i, j)$$

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.

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 $l(j, i) = u(j, i) = 0$).

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

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

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

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 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 $r(i, j) = u(i, j) - f(i, j) + f(j, i) - l(j, i)$. 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

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

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

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 k consisting entirely of admissilbe 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 is exists;

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

The problem in each phase essentially reduce 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.

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 dynamic networks with lower bounds

Dynamic network models arise in many problem settings, including production distribution systems, economic planning, energy systems, traffic systems, and building evacuation systems [3], [4], [5].

Let $G = (N, A, l, u)$ be a static network with the set of nodes $N = \{1, \ldots, n\}$, the set of arcs $A = \{a_1, \ldots, a_m\}$, the lower bound function l, 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 $H = \{0, 1, \ldots, T\}$ be the set of periods, where T is a finite time horizon, $T \in \mathbb{N}$. Let use state 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 $q : A \times H \to \mathbb{N}$, $e\,(i, j; t) \leq q(i, j; t)$, for all $(i, j) \in A$ and for all $t \in H$. The parameter $h(i, j; t)$ is the transit time needed to traverse an arc (i, j). The parameters $e(i, j; t)$ and $q(i, j; t)$ represents the minimum and respective 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 $D = (N, A, h, e, q)$:

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

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

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

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

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

where $\tau = t - h(k, i; \tau)$, $\bar{w} = \sum_{t=0}^{T} v(t)$, $v(t)$ 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 $D = (N, A, h, e, q)$ is more complex than the problem of finding a maximum flow in the static network $G = (N, A, l, u)$. Fortunately, this issue can be solved by rephrasing the problem in the dynamic network D into a problem in the static network $R_1 = (V_1, E_1, l_1, u_1)$ called the reduced expanded network.

The static expanded network of dynamic network $D = (N, A, h, e, q)$ is the network $R = (V, E, l, u)$ with $V = \{i_t | i \in N, t \in H\}$, $E = \{(i_t, j_\theta) | (i, j) \in A, t \in \{0, 1, \ldots, T - h(i, j; t)\}$, $\theta = t + h(i, j; t), \theta \in H\}$, $l(i_t, j_\theta) = e(i, j; t)$, $u(i_t, j_\theta) = q(i, j; t), (i_t, j_\theta) \in E$. The number of nodes in the static expanded network R is $n(T+1)$ and number of arcs is limited by $m(T+1) - \sum_{A} \circ h(i, j)$, where $\circ h(i, j) = min\{h(i, j; 0), \ldots, h(i, j; T)\}$. 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 $R_2 = (V_2, E_2, l_2, u_2)$, where $V_2 = V \cup \{0, n + 1\}$, $E_2 = E \cup \{(0, 1_t) | t \in H\} \cup \{(n_t, n + 1) | t \in H\}$, $l_2(i_t, j_\theta) = l(i_t, j_\theta)$, $u_2(i_t, j_\theta) = u(i_t, j_\theta)$, $(i_t, j_\theta) \in E$, $l_2(0, 1_t) = l_2(n_t, n + 1) = 0$, $u_2(0, 1_t) = u_2(n_t, n + 1) = \infty$, $t \in H$.

We construct the static reduced expanded network $R_1 = (V_1, E_1, l_1, u_1)$ 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 $i_t \in V$ are performing by means of the usual shortest path algorithms. The network $R_1 = (V_1, E_1, l_1, u_1)$ 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 l_1, u_1 are restrictions of l_2, u_2 at E_1.

Now, we construct the static reduced expanded network $R_1 = (V_1, E_1, l_1, u_1)$ 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 $d(i, n; t)$ the length of the dynamic shortest path at time t from the node i to the sink node n, with respect to h in dynamic network D. Let as 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 sources, multiple sinks static reduced expanded network $R_0 = (V_0, E_0, l_0, u_0)$ have $V_0 = \{i_t | i \in N, t \in H_i\}$, $E_0 = \{(i_t, j_\theta) | (i, j) \in A, t \in H_{i,j}\}$, $l_0(i_t, j_\theta) = e(i, j; t)$, $u_0(i_t, j_\theta) = u_1(i, j; t), (i_t, j_\theta) \in E_0$. The static reduced expanded network $R_1 = (V_1, E_1, l_1, u_1)$ is constructed from network R_0 as follows: $V_1 = V_0 \cup \{0, n+1\}, E_1 = E_0 \cup \{(0, 1_t) | 1_t \in V_0\} \cup \{(n_t, n+1) | n_t \in V_0\}$, $l_1(0, 1_t) = l_1(n_t, n+1) = 0$, $u_1(0, 1_t) = u_1(n_t, n+1) = \infty$, $1_t, n_t \in V_0$, $l_1(i_t; j_\theta) = l_0(i_t, j_\theta)$ 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 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 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 the location at the same or at an earlier time, the static networks R, R_2, R_0, R_1 cannot contain any circuit, and therefore, are acyclic always.

In the most general dynamic model, the parameter $h(i) = 1$ is waiting time at node i, and the parameter $e(i; t)$, $q(i; t)$ are lower bound and 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:

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

$$l_0(i_t, j_\theta) \leq f_0(i_t, j_\theta) \leq u_0(i_t, j_\theta), \quad (i_t, j_\theta) \in E_0 \tag{9}$$

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

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 $d(1, i)$, $d(i, n)$.

Many notions from this section are presented and in papers [4], [7].

2.3 Maximum flows in bipartite static networks

In this section we consider that the static network $G = (N, A, l, u)$ is 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 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 $l(1', 1) = 0$, $u(1', 1) = \infty$). 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.

Table 1: Several maximum flows algorithms

Algorithm	*Running time, general network*	*Running time, bipartite network*	*Running time modified version*
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	$nm + n^2 \log \bar{u}$	$n_1m + n_1n \log \bar{u}$	$n_1m + n_1^2 \log \bar{u}$

Ahuja, Orlin, Stein, and Tarjan [2] obtained further running time improvements by modifying the algorithms. This modification applies only to preflow push algorithms. They call 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. 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])

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

Algorithm 1 The algorithm for a feasible flow in R_0.

```
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;
  compute the exact distance labels d(i) from t to i;
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;
        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;
```

For more information see [2].

We notice the fact that have used the notations from this paper and specify that this algorithm runs on networks G with $l = 0$, a single source node 1, a single sink node n.

2.4 Feasible flows in static networks

We consider the flow problem satisfying the conditions (1a) and (1b). The condition (1a) is the flow conservation condition and the condition (1b) is the feasibility condition.

Let f be a flow of value v in the static network $G = (N, A, l, u)$. Let be $\bar{G}$ the static network we get by adding the return arc $(n, 1)$ to the network G. We extend the mappings l, u and f to $\bar{G}$ as follows:

$$\bar{l}(n,1)=0, \bar{u}(n,1)=\infty, \bar{f}(n,1)=v \tag{11}$$

Then $\bar{f}$ is a circulation on $\bar{G}$:

$$\bar{f}(i,N)-\bar{f}(N,i)=0, i\in N \tag{12a}$$

$$\bar{l}(i,j)\le \bar{f}(i,j)\le \bar{u}(i,j) \tag{12b}$$

We notice that the static network $\bar{G}=(\bar{N},\bar{A},\bar{l},\bar{u})$ has $\bar{N}=N$, $\bar{A}=A\cup\{(n,1)\}$, $\bar{l}(i,j)=l(i,j)$, $\bar{u}(i,j)=u(i,j)$, $\bar{f}(i,j)=f(i,j)$, $(i,j)\in A$ and usual, ∞ means a sufficiently large number, for example $\bar{u}(1,N)$.

Therefore the feasible flow problem we are looking for corresponding to an feasible circulation on network $\bar{G}$. However, note that the feasible flow problem might well be unsolvable, because there might not be any feasible circulations.

A cut in the static network $G=(N,A,l,u)$ is an arc set $[X,Y]=(X,Y)\cup(Y,X)$ with $X\subset N$, $Y=N-X$, $(X,Y)=\{(i,j)|(i,j)\in A, i\in X, j\in Y\}$, $(Y,X)=\{(j,i)|(j,i)\in A, j\in Y, i\in X\}$. The set (X,Y) denote the set of forward arcs in the cut and the set (Y,X) denote the set of backward arcs in the cut. We refer to a cut $[X,Y]$ as a $1-n$ cut if $1\in X$ and $n\in Y$. For the maximum flow problem the capacity of $1-n$ cut $[X,Y]$ is

$$c[X,Y]=u(X,Y)-l(Y,X) \tag{13}$$

A $1-n$ cut whose capacity is the minimum among all $1-n$ cuts is a minimum cut.

In [1] the next theorems are proved.

Theorem 1. *(Generalized Max-Flow Min-Cut Theorem). The maximum value of the flow from a source node* 1 *to a sink node* n *in a static network* $G=(N,A,l,u)$ *is equal to the capacity of a minimum* $1-n$ *cut.*

Theorem 2. *(Feasible Circulation Theorem). A necessary and sufficient condition for the existence of a feasible circulation on* $G=(N,A,l,u)$ *is that* $l(Y,X)\le u(X,Y)$ *for any cut* $[X,Y]$ *of* G.

Theorem 3. *(Feasible Flow Theorem). A necessary and sufficient condition for the existence of a feasible flow on* $G=(N,A,l,u)$ *is that* $l(Y,X)\le u(X,Y)$ *for all partitions* $N=X\cup Y$ *with* $1\notin Y$ *or* $n\notin X$.

Theorem 3 result from Theorem 2 if transform the flow f on $G=(N,A,l,u)$ into circulation $\bar{f}$ on $\bar{G}=(\bar{N},\bar{A},\bar{l},\bar{u})$.

3 The feasible flow problem in dynamic networks

3.1 The feasible flow in dynamic networks

We consider the flow problem satisfying the conditions (5) and (6). A flow g on the dynamic network $D=(N,A,h,e,q)$ satisfying the conditions (5) and (6) is equivalent with the flow f_0 on the multiple sources, multiple sinks static reduced expanded network $R_0=(V_0,E_0,l_0,u_0)$ satisfying the conditions (8) and (9).

Recall the construction of sets H_i and $H_{i,j}$: $H_i=\{t|t\in H, d(1,i;t)\le t\le T-d(i,n;t)\}$, $i\in N$, and $H_{i,j}=\{t|t\in H, d(1,i;t)\le t\le T-h(i,j;t)-d(j,n;\theta)\}$, $(i,j\in A)$. This sets make the connection between the dynamic network D and the static network R_0. Therefore we reformulate Theorem 1, Theorem 2, Theorem 3 for the static network R_0.

Let V_0 be $V_0=V_{01}\cup\ldots\cup V_{0i}\cup\ldots V_{0n}$ with $V_{0i}=\{i_t|i\in N, t\in H_i\}, i=1,\ldots,n$. We refer to a cut $[X_0,Y_0]$ with $X_0\subset V_0$, $Y_0=V_0-X_0$ as a $V_{01}-V_{0n}$ cut if $V_{01}\subset X_0$ and $V_{0n}\subset Y_0$.

Theorem 4. *(Generalized Dynamic Max-Flow Min-Cut Theorem). The maximum value of the flow from a source node V_{01} to a sink node V_{0n} in a static network $R_0 = (V_0, E_0, l_0, u_0)$ is equal to the capacity of a minimum $V_{01} - V_{0n}$ cut.*

Theorem 5. *(Feasible Dynamic Circulation Theorem). A necessary and sufficient condition for the existance of a feasible circulation on $R_0 = (V_0, E_0, l_0, u_0)$ is that $l_0(Y_0, X_0) \leq u_0(X_0, Y_0)$ for any cut $[X_0, Y_0]$ of R_0.*

Theorem 6. *(Feasible Dynamic Flow Theorem). A necessary and sufficient condition for the existence of a feasible flow on $R_0 = (V_0, E_0, l_0, u_0)$ is that $l_0(Y_0, X_0) \leq u_0(X_0, Y_0)$ for all partitions $V_0 = X_0 \cup Y_0$ with $V_{01} \notin Y_0$ or $V_{0n} \notin X_0$.*

If h, e, q are constant over time, then a dynamic network $D = (N, A, h, e, q)$ is said to be stationary. Ford and Fulkerson [5] have devised an algorithm that generates a maximum flow in the stationary dynamic network $D = (N, A, H, 0, q)$, the case when $e = 0$. The flow obtained with the algorithm Ford and Fulkerson is called temporally repeated flow. Let $c : A \to \mathbb{N}$ be the fucntion cost. For many details is urged to consult the book [5].

There are two inconveniences for the flow problem in the stationary dynamic network $D = (N, A, h, e, q)$. The first is that although in the planar network $G = (N, A, c, l, u)$ with $c = h, l = e, u = q$ exist a feasible flow, it is possible that in the static network $R_0 = (V_0, E_0, l_0, u_0)$ will not be any feasible flow. The second inconvenience consist in the fact that it is possible that the temporally repeated flow in the network R_0 of a feasible and minimum time flow in the network $G = (N, A, c, l, u)$, will not be any feasible in the network R_0.

3.2 Examples

The stationary dynamic network $D = (N, A, h, e, q)$ is presented in Figure 1 and the time horizon set to $T = 5$, therefore $H = \{0, 1, 2, 3, 4, 5\}$. The transit time $h(i, j)$, the lower bound $e(i, j)$ and the upper bounds $q(i, j)$ for all arcs $(i, j) \in A$ are indicated in this order near arcs.

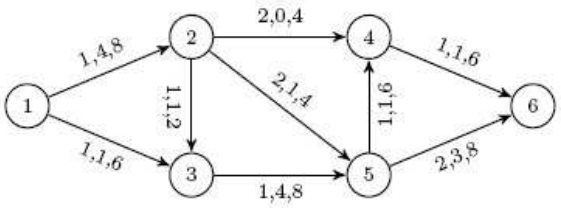

Figure 1: The stationary dynamic network D.

We obtain: $d(1,1) = 0, d(1,6) = 4$, $d(1,2) = 1, d(2,6) = 3$, $d(1,3) = 1, d(3,6) = 3$, $d(1,4) = 3, d(4,6) = 1$, $d(1,5) = 2, d(5,6) = 2$, $d(6,6) = 0$; $H_1 = \{0,1\}$, $H_2 = \{1,2\}$, $H_3 = \{1,2\}$, $H_4 = \{3,4\}$, $H_5 = \{2,3\}$, $H_6 = \{4,5\}$; $H_{1,2} = \{0,1\}$, $H_{1,3} = \{0,1\}$, $H_{2,3} = \{1\}$, $H_{2,4} = \{1,2\}$, $H_{2,5} = \{1\}$, $H_{3,5} = \{1,2\}$, $H_{4,6} = \{3,4\}$, $H_{5,4} = \{2,3\}$, $H_{5,6} = \{2\}$.

The static network $R_0 = (V_0, E_0, l_0, u_0)$ is presented in Figure 2. The lower bounds $l_0(i_t, j_\theta)$ and the upper bounds $u_0(i_t, j_\theta)$ for all arcs $(i_t, j_\theta) \in E_0$ are indicated in this order near arcs.

For all partitions $N = X \cup Y$ with $1 \notin Y$ or $n \notin X$ of the static network $G = (N, A, c = h, l = e, u = q)$ is verified the condition from Theorem 3. Therefore exist a feasible flow on G. Also, is verified Theorem 6 on the network $R_0 = (V_0, E_0, l_0, u_0)$.

Example 1. We replace in the network from Figure 1 $e(1,2) = 4$, $q(2,4) = 4$, $q(2,5) = 4$ with $e(1,2) = 7$, $q(2,4) = 2$, $q(2,5) = 2$. Now, we consider the partition $N = X \cup Y$ with $X = \{2\}$ and

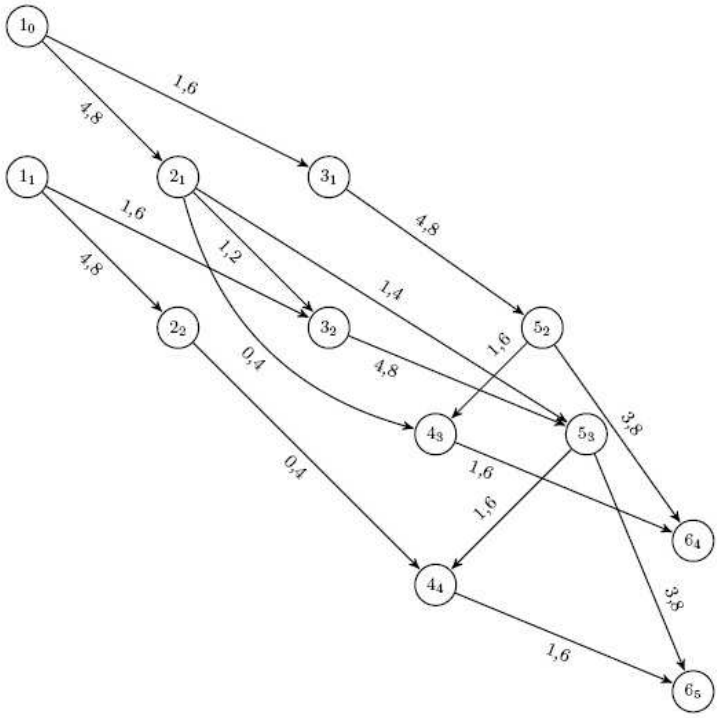

Figure 2: The static network R_0.

$Y = \{1, 3, 4, 5, 6\}$. We have $(X, Y) = \{(2, 3), (2, 4), (2, 5)\}$, $(Y, X) = \{(1, 2)\}$ and in the network $G = (N, A, c = h, l = e, u = q)$ we obtain $l(Y, X) = l(1, 2) = 7 > 6 = u(2, 3) + u(2, 4) + u(2, 5) = u(X, Y)$. Therefore not exist a feasible flow on G.

Example 2. We replace in the network from Figure 1 $q(2, 4) = 4$ with $q(2, 4) = 3$. For $q(2, 4) = 3$ the Theorem 3 is verified. Therefore exist a feasible flow on network $G = (N, A, c = h, l = e, u = q)$. If $q(2, 4) = 3$ then $u_0(2_1, 4_3) = u_0(2_2, 4_4) = 3$. We consider the partition $V_0 = X_0 \cup Y_0$ with $X_0 = \{2_2\}$ and $Y_0 = V_0 - X_0$. We have $(X_0, Y_0) = \{(2_2, 4_4)\}$, $(Y_0, X_0) = \{1_1, 2_2\}$ and $l_0(Y_0, X_0) = l_0(1_1, 2_2) = 4 > 3 = q(2, 4) = u_0(2_2, 4_4) = u_0(X_0, Y_0)$. From Theorem 6 we obtain that the flow problem in the static network R_0 (dynamic network D) is infeasible.

4 Maximum flows in bipartite dynamic networks with lower bounds

4.1 Maximum flows

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

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 generalised bipartite FIFO preflow (GBFIFOP) is presented in Algorithm 2.

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

Theorem 7. *The GBFIFOP algorithm which determines a maximum flow into the bipartite dynamic network $D = (N, A, h, q)$ has the complexity $O(n_1 m T^2 + n_1^3 T^3)$.*

Proof: In Section 3 we specify that the maximum flow problem for T time periods in the dynamic network $D = (N, A, h, e, q)$ is equivalent with the maximum flow problem in the static reduced expanded network $R_0 = (V_0, E_0, l_0, u_0)$. The networks D and R_0 are bipartite with $N = N_1 \cup N_2$, $V_0 = W_1 \cup W_2$. 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 $G = (N_1 \cup N_2, A, l, u)$ in $O(n_1 m + n_1^3)$ how is specified in table from Section 4. We apply the generalizate 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 $w_1 = n_1 T$ and $\varepsilon_0 \leq mT$. As a result the algorithm has the complexity $O(n_1 m T^2 + n_1^3 T^3)$.

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. □

Algorithm 2 The generalised bipartite FIFO preflow (GBFIFOP) algorithm

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

```
PROCEDURE PREPROCESS;
BEGIN
f_0 is a feasible flow in R_0; Q := ∅;
compute the exact distance labels d(i_t);
for t ∈ H_1 do
    BEGIN
        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
        d(1_t) := 2w_1 + 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_θ) > 0;
β := 1;
repeat
    if (i_t, j_θ) is admissible arc then
        BEGIN
        select the first arc (j_θ, k_τ) in E_0^+(j_θ) with r_0(j_θ, k_τ) > 0;
        if (j_θ, k_τ) is admissible arc then
            BEGIN
            push α := min {e(i_t), r_0(i_t, j_θ), r_0(j_θ, k_τ)} units of flow over the arcs
             (i_t, j_θ), (j_θ, k_τ);
            if k_τ ∉ Q then
                adds node k_τ to the rear of Q;
            END
        else if (j_θ, k_τ) is not the last arc in E_0^+(j_θ) with r_0(j_θ, k_τ) > 0 then
            select the next arc in E_0^+(j_θ)
        else
            d(j_θ) := min{d(k_τ) + 1|(j_θ, k_τ) ∈ E_0^+(j_θ), r_0(j_θ, k_τ) > 0};
        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_θ) > 0 then
                select the next arc in E_0^+(j_θ)
            else
                BEGIN
                  d(i_t) := min{d(j_θ) + 1|(i_t, j_θ) ∈ E_0^+(j_θ), r_0(i_t, j_θ) > 0};
                  β := 0;
                END;
        END
until e(i_t) = 0 or β = 0
if e(i_t) > 0 then
    adds node i_t to the rear of Q;
END;
```

4.2 Example

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

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 2.

Applying the GBFIFOP 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 4. 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.

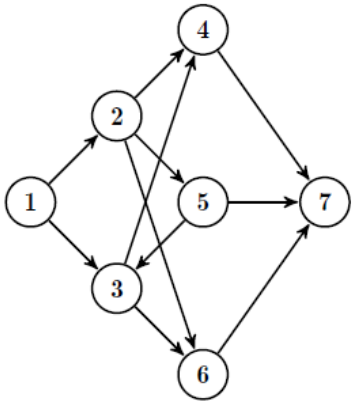

Figure 3: The support digraph of network $D = (N, A, h, e, q)$

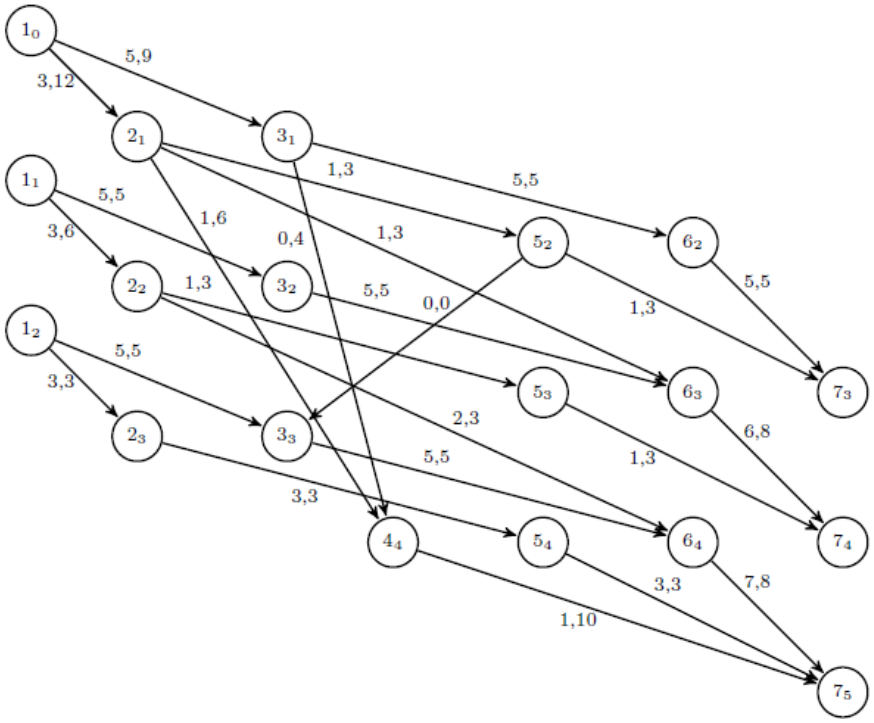

Figure 4: The network $R_0 = (V_0, E_0, f_0, f_0^*)$.

Table 2: The functions h, e, 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
$e(i,j)$	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

Conclusions

In this paper we approached two flow problems: the feasible flow problem in dynamic networks and the maximum flow problem in bipartite dynamic networks with lower bounds. For the maximum flows problem in bipartite dynamic networks with lower bounds we developed an algorithm. These problem have not been treated so far. We demonstrate the fact that if the dynamic network $D = (N, A, h, e, q)$ is bipartite, then the static reduced expanded network $R_0 = (V_0, E_0, l_0, u_0)$ is bipartite. For solving, we have rephrased the maximum flows problem in bipartite dynamic networks with lower bound into a problem in bipartite static network. Then, we have extended the bipartite FIFO preflow algorithm of Ahuja et al. [2] for the static reduced expanded network with multiple source and multiple sinks $R_0 = (V_0, E_0, l_0, u_0)$. Also we have presented the complexity for the generalization bipartite FIFO preflow algorithm. For each of the two problems we have presented one example.

Flow problems in bipartite dynamic networks like: the parametric maximum flow problem, the minimum cost flow problem, the generalization of the highest label preflow push algorithm or the generalization of the excess scaling algorithm are still open for research.

Bibliography

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

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

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

[4] E. Ciurea (2002), Second best temporally repeated flow, *Korean Journal of Computational and Applied Mathematics*, 9(1):77-86.

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

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

[7] C. Schiopu, E. Ciurea (2016), The maximum flows in planar dynamic dynamic networks, *International Journal of Computers Communications & Control*, 11(2):282-291.

[8] C. Schiopu, E. Ciurea, The maximum flows in bipartite dynamic networks with lower bounds. The static approach, *Computers Communications and Control (ICCCC), 2016 6th International Conference on*, IEEE Xplore, e-ISSN 978-1-5090-1735-5, doi: 10.1109/ICCCC.2016.7496731, 10-15.