The Minimum Flows in Bipartite Dynamic Networks. The Static Approach

E Ciurea, C Schiopu

The Minimum Flows in Bipartite Dynamic Networks. The Static Approach.

E. Ciurea C. Schiopu

Abstract

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

Keywords: bipartite dynamic network flow, minimum flow, bipartite static network flow.

1 Introduction

The theory of flow is one of the most important parts of Combinatorial Optimization. The static network flow models arises 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, the time is an essential ingredient [3], [7], [9], . In this case we need to use dynamic network flow model. On the other hand, the bipartite static network also arise in practical context such baseball elimination problem, network reliability testing etc. and hence it is of interes to find fast flow algorithms for this class of networks [2], [8].

The computation of a maximum flow in dynamic networks with lower bound function $e = 0$ has been investigated by many researchers [3] , [7], [10], [11] and other. Unfortunately, the minimum flow problem in dynamic networks was not treated as often as the maximum flow problem. In the paper [4] is presented an algorithm for minimum flow in stationary dynamic networks with lower bounds $e(i, j) \geq 0$ and upper bounds $q(i, j) = \infty$. Salehi H.F., et al. [9] has considered minimum flow problem on a class of dynamic networks called dynamic generative networks, where travel and transmission time are instantaneous but flow generation and consumption are dynamical. The authors consider the lower bounds of the arcs leaving source node are increasing functions of the time, the lower bounds of the arcs entering the sink node are decreasing functions of the time and the lower bounds of the other arcs and upper bounds for all arcs are constant.

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

2 Terminology and preliminaries.

In this section we discuss some basic notations and results used in the rest of the paper.

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 nodes set 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 arcs set 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 minimum flow problem is to determine a flow f for which v is minimum.

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

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

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

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

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

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

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.

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

A cut is set of arcs $[X, \bar{X}] = (X, \bar{X}) \cup (\bar{X}, X)$, $(X, \bar{X}) = \{(i, j) | (i, j) \in A, i \in X, j \in \bar{X}\}$, $X \subset N$, $\bar{X} = N - X$, $(\bar{X}, X) = \{(i, j) | (i, j) \in A, i \in \bar{X}, j \in X\}$. The set $(X, \bar{X})$ denote the set of forward arcs of the cut, and the set $(\bar{X}, X)$ denote the set of backward arcs of the cut. We refer to a cut $[X, \bar{X}]$ as a $1 - n$ cut if $1 \in X$ and $n \in \bar{X}$. For the minimum flow problem, the capacity $c[X, \bar{X}]$ of a $1 - n$ cut is $c[X, \bar{X}] = l(X, \bar{X}) - u(\bar{X}, X)$. We refer to a $1 - n$ cut whose capacity is the maximum among all $1 - n$ cuts as a maximum cut.

Theorem 1. *The value of the minimum flow from the source node* 1 *to the sink node* n *in a network* $G = (N, A, l, u)$ *equals to the capacity of the maximum* $1 - n$ *cut.*

For the minimum flow problem, the residual capacity $\hat{r}(i, j)$ of any arc $(i, j) \in A$ with respect to a given flow (preflow) f is given by $\hat{r}(i, j) = c(j, i) - f(j, i) + f(i, j) - l(i, j)$. The residual network is $\hat{G} = (n, \hat{A}, \hat{r})$ with $\hat{A} = \{(i, j) | (i, j) \in A, \hat{r}(i, j) > 0\}$.

The minimum flow problem in a network $G = (N, A, l, u)$ can be solved in two phases:

(1) estabilish a feasible flow if it exists;

(2) if exists a feasible flow, estabilish a minimum flow;

The solution of first fase is presented in [1]. There are three approaches for solving the minimum flow problem:

(1)using decreasing path algorithms;

(2) using preflow algorithms;

(3) minmax algorithms.

Any directed path from the source nodes 1 to the sink node n in the residual network $\hat{G}$ corresponds to a decreasing path in the original network G. Thus, the decreasing path algorithms for minimum flow problem correspond to the augmenting path algorithms for maximum flow problem.

Also, the preflow algorithms for minimum flow problem correspond to the preflow algorithms for maximum flow problem.

In the next presentation we assume familiarity with minimum flow algorithms, and we omit many details. The reader interested in further details is urged to consult the paper [5].

3 Minimum flows in dynamic networks

Dynamic network models arise in many problem settings, including production distribution systems, economic planning, energy systems, traffic systems, and building evacuation systems [3], [7], [10].

Let $G = (N, A, l, u)$ be a static network with the node set $N = \{1, \ldots, n\}$, the arc set $A = \{a_1, \ldots, a_m\}$, 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 minimum 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 $D = (N, A, h, e, q)$:

$$\sum_{t=0}^{T}(g(1, N; t) - \sum_{\tau} g(N, 1; \tau)) = \bar{w} \tag{4a}$$

$$g(i, N; t) - \sum_{\tau} g(N, i; \tau) = 0, i \neq 1, n,\ t \in H \tag{4b}$$

$$\sum_{t=0}^{T}(g(n, N; t) - \sum_{\tau} g(N, n; \tau)) = -\bar{w} \tag{4c}$$

$$e(i, j; t) \leq g(i, j; t) \leq q(i, j; t), \quad (i, j) \in A \ , \quad t \in H \tag{5}$$

$$min \quad \bar{w}, \tag{6}$$

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 minimum flow in dynamic network $D = (N, A, h, e, q)$ is more complex than the problem of finding a minimum flow in static network $G = (N, A, l, u)$. Happily, this complication can be resolved by rephrasing the problem in dynamic network D into a problem in 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 static expanded network R is $n(T+1)$ and number of arcs is limited by $m(T+1) - \sum_A \mathring{h}(i, j)$, where $\mathring{h}(i, j) = min\{h(i, j; 0), \ldots, h(i, j; T)\}$. It is easy to see that any flow in dynamic network D from the source node 1 to the sink node n is equivalent to a flow in 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 static expanded network R to a single source, single sink problem by introducing a supersource node 0 and a supersink node $n+1$ constructing 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 source, 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 remark the fact 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], [7] 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 an earlier time, the static networks R, R_2, R_0, R_1 cannot contain any circuit, and are therefore 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 dynamic network D formulated in conditions (4), (5), (6) is equivalent with the maximum flow problem in static reduced expanded

network R_0 as follows:

$$f_0(i_t, V_0) - f_0(V_0, i_t) = \begin{cases} v_t, & \text{if } i_t = 1_t, t \in H_1 \\ 0, & \text{if } i_t \neq 1_t, n_t, t \in H_1, t \in H_n \\ -v_t, & \text{if } i_t = n_t, t \in H_n \end{cases} \tag{7a}$$

$$l_0(i_t, j_\theta) \leq f_0(i_t, j_\theta) \leq u_0(i_t, j_\theta), \quad (i_t, j_\theta) \in E_0 \tag{8}$$

$$min \sum_{H_1} v_t, \tag{9}$$

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

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

4 Minimum flows in bipartite static networks

In this section we consider that static network $G = (N, A, l, u)$ is bipartite static network. A bipartite network has the node set 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 $n_2 = |N_2|$. For minimum flow problem, without any loss of generality, we assume that $n_2 \leq n_1$. 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 $n_2 << n_1$ and balanced otherwise.

Algorithm	*Running time, general network*	*Running time, bipartite network*	*Running time modified version*
Dinic	n^2m	n_2^2m	*does not apply*
Karazanov	n^3	n_2^2n	$n_2m + n_2^3$
FIFOpreflow	n^3	n_2^2n	$n_2m + n_2^3$
Highestlabel	$n^2\sqrt{m}$	$n_2n\sqrt{m}$	n_2m
Excessscaling	$nm + n^2 \log \bar{u}$	$n_2m + n_2n \log \bar{u}$	$n_2m + n_2^2 \log \bar{u}$

Table 1: Several minimum flows algorithms

In reference [8] the authors show that the time bounds for several minimum flow algorithms automatically improve 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 $L \leq 2n_2+1$. Hence for unbalanced bipartite network $L << n$. Column 3 of Table 1 summarizes these improvements for several network flow algorithms.

The authors of references [6] obtain further running time improvements by modifying the algorithms. This modification applies only to preflow pull algorithm. They call it the two arcs pull rule. According to this rule, always pull flow from a node in N_2 and pull flow on two arcs at a time, in a step called a bipull, so that no excess accumulates at nodes in N_2. Column 4 of Table 1 summarizes the improvements obtained using this approach. The reader interested in further details is urged to consult the paper [6].

5 Minimum flows in bipartite dynamic networks

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 remark 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 $n_2 << n_1$ then obvious that $w_2 << w_1$. In the static bipartite network R_0 we determine a minimum flow f_0 with a generalization of bipartite FIFO preflow algorithm.

We recall that the FIFO preflow algorithm might perform several saturating pulls followed either by a nonsaturating pull 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 select a node j from the front of Q, performs pulls from this node, and adds newly active nodes to the rear of Q. The algorithm examines node j until either it becomes inactive or it is relabeled. In the latter case, we add node j to the rear of the queue Q. The algorithm terminates when the queue Q of active nodes is empty.

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

We specify that maintain the arc list $E_0^-(j_\theta) = \{(i_t, j_\theta) | (i_t, j_\theta) \in E_0\}$. We can arrange the arcs in these lists arbitrarily, but the order, once decided, remains unchanged throughout the algorithm. Each node j_θ has a current arc, which is an arc in $E_0^-(i_t)$ and is the next candidate for admissibility testing. Initialy, the current arc of node j_θ is the first arc in $E_0^-(j_\theta)$. Whenever the algorithm attempts to find an admissible arc emanating from node j_θ, 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 either to finds admissible arc or reaches the end of the arc list.

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

```
1: ALGORITHM GBFIFOP;
2: BEGIN
3: PREPROCESS
4: while Q ≠ ∅ do
5:     select the node j_θ from the front of Q;
6:     BIPULL/RELABE(j_θ);
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 function d̂ in residual network Ĝ;
5: for λ ∈ H_n do
6:     f_0(j_θ, n_λ) := l_0(j_θ, n_λ) and adds node j_θ to the rear of Q for all (j_θ, n_λ) ∈ E_0
7:     d̂(n_λ) := 2w_2 + 1;
8: end for
9: END.
```

```
PROCEDURE BIPULL/RELABEL(j_θ);
BEGIN
select the first arc (i_t, j_θ) in E_0^-(j_θ) with r̂_0(i_t, j_θ) > 0;
β := 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 r̂_0(k_τ, i_t) > 0;
        if (k_τ, i_t) is admissible arc then
            pull α := min {−e(j_θ), r̂_0(i_t, j_θ), r̂_0(k_τ, i_t)}
            units of flow over the arcs (i_t, j_θ), (k_τ, i_t);
            if k_τ ∉ 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 r̂_0(k_τ, i_t) > 0 then
                select the next arc in E_0^-(i_t)
            else
                d̂(i_t) := min {d(k_τ) + 1|(k_τ, i_t) ∈ E_0^-(i_t), r̂_0(k_τ, i_t) > 0}
            end if
        end if
        if e(j_θ) > 0 then
            if (i_t, j_θ) is not the last arc in E_0^-(j_θ) with r̂_0(i_t, j_θ) > 0 then
                select the next arc in E_0^-(j_θ)
            else
                d̂(j_θ) := min {d(i_t) + 1|(i_t, j_θ) ∈ E_0^-(j_θ), r̂_0(i_t, j_θ) > 0};
                β := 0;
            end if
        end if
    end if
until e(j_θ) = 0 or β = 0
if e(j_θ) > 0 then
    adds node j_θ to the rear of Q;
end if
END.
```

Figure 1: The generalizate bipartite FIFO prefow (GBFIFOP) algorithm

We remark 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 $\hat{d}(n_\lambda) := 2w_2 + 1$ in PROCEDURE PREPROCES.

The correctness of the GBFIFOP algorithm results from correctness of the algorithm for minimum flow in bipartite network [6].

Theorem 2. *The GBFIFOP algorithm which determine a minimum flow into bipartite dynamic networks $D = (N, A, h, q)$ has the complexity $O(n_2 m T^2 + n_2^3 T^3)$.*

Proof. In Section 3 we specify that the minimum flow problem for T time periods in dynamic network $D = (N, A, h, e, q)$ is equivalent with the minimum flow problem in 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 minimum flow into bipartite static network

$G = (N_1 \cup N_2, A, l, u)$ in $O(n_2 m + n_2^3)$ how is specified in table from Section 4. We apply the generalizate bipartite FIFO preflow algorithm in static reduced expanded bipartite network R_0. Hence the algorithm has the complexity $O(w_2\varepsilon_0 + w_2^3)$. From Section 4 we have $w_2 = n_2 T$ and $\varepsilon_0 \leq mT$. As a result the algorithm has the complexity $O(n_2 m T^2 + n_2^3 T^3)$.

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. □

6 Example

The support digraph of bipartite dynamic network is presented in Figure 2 and time horizon being set $T = 5$, therefore $H = \{0, 1, 2, 3, 4, 5\}$. The transit times $h(i, j; t) = h(i, j), t \in H$, 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.

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

Table 2: The functions h, e, q

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

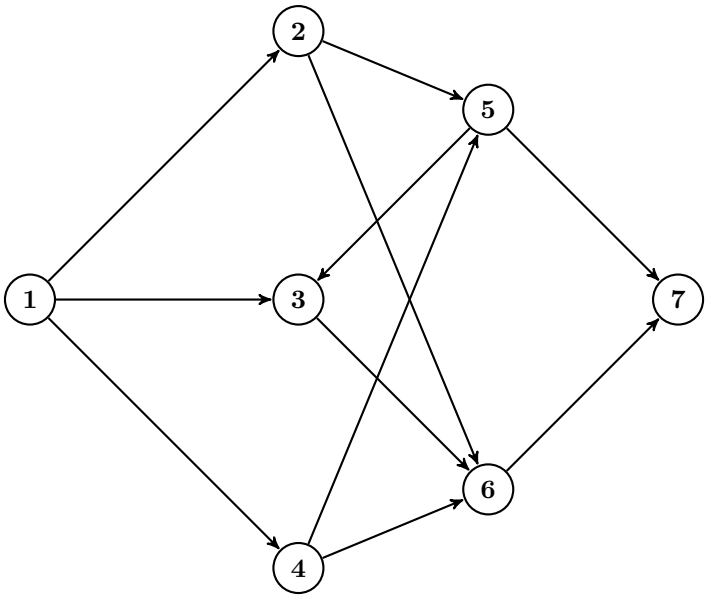

Figure 2: The suport digraph of network $D = (N, A, h, q)$

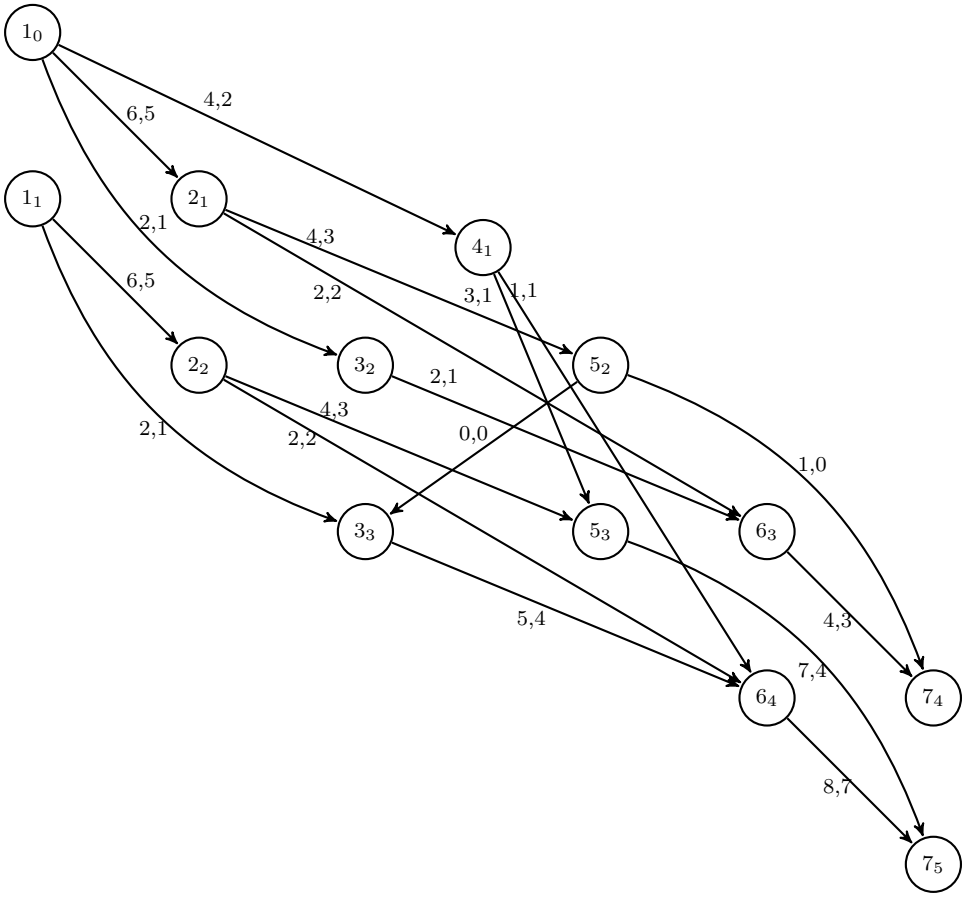

Figure 3: The network $R_0 = (V_0, E_0, u_0)$.

Applying the GBFIFOP algorithm in the first phase and the second phase we obtain the flows $f_0(i_t, j_\theta)$, $f_0^*(i_t, j_\theta)$ (feasible flow, minimum flow) which are indicated in Figure 3. 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 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.

References

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

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

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

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

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

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

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

[9] Salehi, H. F., Khadayifar, S., Raayatpanoh, M.A., (2012); Minimum flow problem on network flows with time-varying bounds, Applied Mathematical Modelling, vol. 36, issue 9, pp. 4414-4421.

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

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