Bulletin of the *Transilvania* University of Braşov • Vol 7(56), No. 2 - 2014
Series III: Mathematics, Informatics, Physics, 193-202

THE MAXIMUM FLOWS IN BIPARTITE DYNAMIC NETWORKS

Camelia SCHIOPU[1]

Communicated to:
International Conference on Mathematics and Computer Science ,
June 26-28, 2014, Braşov, Romania

Abstract

In this paper we study maximum flow algorithms for stationary bipartite dynamic networks. In a bipartite static network the several maximum flow algorithms can be substantially improved. The basic idea in this improvement is a two arcs push rule. This idea is also extended to minimum cost flow. In the end of the paper we present an example.

2000 *Mathematics Subject Classification:* 0B10, 90C35, 05C35, 68R10.
Key words: bipartite dynamic network flow, maximum flow, minimum cost 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 reaching as far 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], [4], [5]. In this case we need to use dynamic network flow model. On the other hand, the bipartite static network also arises in practical context such 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].

In this paper we present the maximum flow problem in bipartite dynamic networks when the dynamic networks are stationary. This problem has not been

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

treated so far. Further on, in Section 2 we discuss some basic notions and results for maximum flow problem in general static networks. Section 3 deals with maximum flow problem in general dynamic networks. In Section 4 we present algorithms for flow problems in bipartite static network and in Section 5 we discuss the maximum flow problem in stationary bipartite dynamic networks. Section 6 deals with an example for a 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 $G = (N, A, u)$ be a general static network with the node set $N = \{1, \ldots, n\}$, the arc set $A = \{a_1, \ldots, a_k, \ldots, a_m\}$, $a_k = (i, j)$, $i, j \in N$, the upper bound (capacity) function u, $u : A \to \mathbb{N}$ with $\mathbb{N}$ the natural number set and with 1 the source node, n the sink node.

For a given pair of subset X, Y of the 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:

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

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

$$f(i, N) - f(N, i) = \begin{cases} v, & \text{if } i = 1 \\ 0, & \text{if } i \neq 1, n \\ -v, & \text{if } i = n \end{cases} \tag{1.1}$$

$$0 \leq f(i, j) \leq u(i, j), \quad (i, j) \in A \tag{1.2}$$

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

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

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

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

$$f(N, i) - f(i, N) \geq 0, i \in N - \{1, n\} \tag{2.1}$$

$$0 \leq f(i, j) \leq u(i, j), (i, j) \in A \tag{2.2}$$

A pseudoflow is a function $f : A \to \mathbb{N}$ satisfying only the constraints (2.2).

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. For any pseudoflow f, we define the imbalance of node $i \in N$ with relation (3) and

if $e(i) > 0$ we refer to $e(i)$ as the excess of node i; if $e(i) < 0$, we call $-e(i)$ the node's deficit.

Given a flow (preflow or pseudoflow) f, the residual capacity $r(i,j)$ of any arc $(i,j) \in A$ is $r(i,j) = u(i,j) - f(i,j) + f(j,i)$. The residual network with respect to the flow (preflow or pseudoflow) f is $\tilde{G} = (N, \tilde{A}, r)$ with $\tilde{A} = \{(i,j)|(i,j) \in A, r(i,j) > 0\}$. In 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 satisties the following two conditions

$$d(n) = 0 \tag{4.1}$$

$$d(i) \leq d(j) + 1, (i,j) \in \tilde{A} \tag{4.2}$$

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

In minimum cost flow problem each arc $(i,j) \in A$ has a cost $c(i,j)$. We assume that the costs are antisymmetric, i.e., $c(i,j) = -c(j,i)$ for each arc $(i,j) \in A$. The minimum cost flow problem can be formulated as follows:

$$min \sum_{A} c(i,j) f(i,j) \tag{5}$$

subject to conditions (1.1) and (1.2).

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

3 Maximum flows in dynamic networks

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

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

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

network $D = (N, A, h, q)$:

$$\sum_{t=0}^{T}(g(1, N; t) - \sum_{\tau} g(N, 1; \tau)) = w \tag{6a}$$

$$g(i, N; t) - \sum_{\tau} g(N, i; \tau) = 0, i \neq 1, n,\ t \in H \tag{6b}$$

$$\sum_{t=0}^{T}(g(n, N; t) - \sum_{\tau} g(N, n; \tau)) = -w \tag{6c}$$

$$0 \leq g(i, j; t) \leq q(i, j; t), \quad (i, j) \in A \quad , \quad t \in H \tag{7}$$

$$max \quad w, \tag{8}$$

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

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

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

Now, we construct the static reduced expanded network $R_1 = (V_1, E_1, u_1)$ using the notion of dynamic shortest path. The dynamic shortest path problem is presented in [3]. Let $d(1, i; t)$ be the length of the dynamic shortest path at time t from the source node 1 to the node i and $d(i, n; t)$ 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 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 $V_0 = \{i_t | i \in N, t \in H_i\}$, $E_0 = \{(i_t, j_\theta) | (i, j) \in A, t \in H_{i,j}\}$, $u_0(i_t, j_\theta) = u_1(i, j; t), (i_t, j_\theta) \in E_0$. The static reduced expanded network $R_1 = (V_1, E_1, l_1, u_1)$ is constructed from network R_0 as follows: $V_1 = V_0 \cup \{0, n+1\}, E_1 = E_0 \cup \{(0, 1_t) | 1_t \in V_0\} \cup \{(n_t, n+1) | n_t \in V_0\}, u_1(0, 1_t) = u_1(n_t, n+1) = \infty$, $1_t, n_t \in V_0$ and $u_1(i_t, j_\theta) = u_0(i_t, j_\theta)$, $(i_t, j_\theta) \in E_0$.

We remark the fact that the static reduced expanded network R_1 is always a partial subnetwork of static super expanded network R_2. In references [4], [5] it is shown that a dynamic flow for T periods in the dynamic network D is equivalent with a static flow in a static reduced expanded network R_1. Since an item released from a node at a specific time does not return to the location at the same or an earlier time, the static networks R, R_2, R_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 $q(i; t)$ is 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 (1), (2), (3) is equivalent with the maximum flow problem in static reduced expanded network R_1 as follows:

$$f_1(i_t, V_1) - f_1(V_1, i_t) = \begin{cases} v_1, & \text{if } i_t = 0 \quad (9a) \\ 0, & \text{if } i_t \neq 0, n+1 \quad (9b) \\ -v_1, & \text{if } i_t = n+1 \quad (9c) \end{cases}$$

$$0 \leq f_1(i_t, j_\theta) \leq u_1(i_t, j_\theta), \quad (i_t, j_\theta) \in E_1 \quad (10)$$

$$max \quad v_1, \quad (11)$$

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

If T is very large, then the static reduced expanded network R_1 becomes very large and the number of calculations required to find a maximum flow in network R_1 becomes prohibitively large. Happily, Ford and Fulkerson [5] have devised an algorithm that generates a maximum flow in dynamic network D. This algorithm works only when h and q are constant over time. If h and q are constant over time, then a dynamic network D is said to be stationary.

The algorithm for maximum dynamic flow in stationary dynamic network $D = (N, A, h, q)$ is presented in Figure 1.

```
1: MDFSDN;
2: BEGIN
3:     AMVMCSF(G, f*);
4:     ADSFEF(f*, r(P_1), ..., r(P_k));
5:     ARPF(r(P_1), ..., r(P_k));
6: END.
```

Figure 1: Algorithm for maximum dynamic flow in stationary dynamic network.

The procedure AMVMCSF performs the algorithm for maximum value and minimum cost flow $\overset{*}{f}$ in static network $G = (N, A, c, u)$, where $c(i,j) = h(i,j)$, $u(i,j) = q(i,j), (i,j) \in A$. Let $O(n, m, \bar{h}, \bar{q})$ be the complexity of procedure AMVMCSF with $\bar{h} = max\{h(i,j)|(i,j) \in A\}$, $\bar{q} = max\{q(i,j)|(i,j) \in A\}$. The procedure ADSFEF performs the algorithm for decomposition of static flow $\overset{*}{f}$ in elementary flows (path flows) with $r(P_s)$ the flow along the path $P_s, s = 1, \ldots, k$, from source node 1 to sink node n. This algorithm has the complexity $O(m^2)$. We remark the fact that it is necessary that $c(P_s) \leq T, s = 1, \ldots, k$. The procedure ARPF performs the algorithm to repeat each path flow, starting out from source node 1 at time periods 0 and repeat it after each time period as long as there is enough time left in the horizon for the flow along the path to arrive at the sink node n. This algorithm has complexity $O(nT)$. Hence, the algorithm MDFSDN has complexity $O(\max\{O(n, m, \bar{h}, \bar{u}), nT\})$. The dynamic flow obtained with the algorithm MDFSDN is called temporally repeated flow and has the value:

$$\overset{*}{w} = (T+1)\overset{*}{v^*} - \sum_A h(i,j)\overset{*}{f^*}(i,j) \tag{12}$$

where $\overset{*}{v^*}$ is value of maximum flow and minimum cost static flow $\overset{*}{f^*}$.

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

4 Flows in bipartite static networks

In this section we consider that static network $G = (N, A, u)$ 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 $n_2 = |N_2|$. 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 $u(1', 1) = \infty$. A bipartite network is called unbalanced if $n_1 << n_2$ and balanced otherwise.

The observation of Gusfield, Martel, and Fernandez-Baca [6] that the time bounds for several maximum 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 $L \leq 2n_1 + 1$. Hence for unbalanced bipartite network $L << n$. Column 3 of Figure 2 summarizes these improvements for several network flow algorithms.

Ahuja, Orlin, Stein, and Tarjan [2] obtain further running time improvements by modifying the algorithms. This modification applies only to preflow push algorithms [4]. 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 Figure 2 summarizes the improvements obtained using this approach.

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

Figure 2: Several maximum flows algorithms and minimum cost flows algorithm.

These ideas are also extended to minimum cost flows. The cost scaling algorithm of Goldberg and Tarjan relies on the concept of approximate optimality [1]. The running time for general network or bipartite network and running time for modified version are presented in Figure 2. We denote $\bar{u} = max\{u(i,j)|(i,j) \in A\}$ and $\bar{c} = max\{c(i,j)|(i,j) \in A\}$.

The reader interested in further details is urged to consult the book [1] and the paper [2].

5 Maximum flows in bipartite dynamic networks

In this section we consider the maximum flows in bipartite dynamic networks in the stationary case i.e. $h(i,j;t) = h(i,j)$, $q(i,j;t) = q(i,j)$, $(i,j) \in A, t \in H$. We use the algorithm MDFSDN which has been presented in Section 3. In this Section the dynamic network $D = (N, A, h, q)$ is bipartite.

We consider the bipartite static network $G = (N, A, c, u)$ where $c(i,j) = h(i,j)$, $u(i,j) = q(i,j), (i,j) \in A$. The procedure AMVMCSF from algorithm MDFSDN performs the algorithm for maximum value and minimum cost

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

Theorem 1. *The algorithm MDFSDN correctly computes the maximum flow in bipartite stationary dynamic network.*

Proof. The algorithm MDFSDN correctly computes the maximum flow in general stationary dynamic network. Obviously that the algorithm is correct for bipartite stationary dynamic network too. □

Theorem 2. *The algorithm MDFSDN applied on bipartite stationary dynamic network has the complexity* $O(max\{n_1 m + n_1^3 \log(n_1 \bar{h}), nT\})$.

Proof. The algorithm MDFSDN applied on general stationary dynamic network has the complexity $O(max\{O(n, m, \bar{h}, \bar{u}), nT\})$. In the bipartite stationary dynamic network we have $O(n, m, \bar{h}, \bar{u}) = O(n_1 m + n_1^3 \log(n_1, \bar{h}))$. Hence the algorithm MDFSDN applied on bipartite stationary dynamic network has the complexity $O(max\{n_1 m + n_1^3 \log(n_1 \bar{h}), nT\})$. □

6 Example

The support digraph of bipartite stationary 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 $h(i, j)$ and the upper bounds (capacities) $q(i, j)$ for all arcs are indicated in Figure 4.

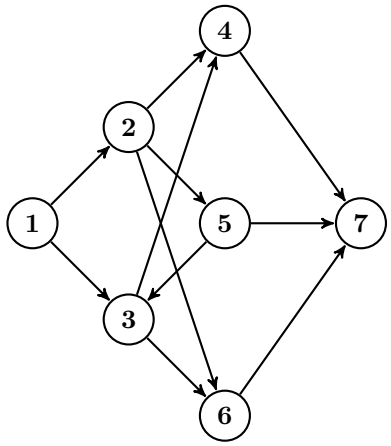

Figure 3: The suport digraph of network $D = (N, A, h, q)$

(i,j)	$(1,2)$	$(1,3)$	$(2,4)$	$(2,5)$	$(2,6)$	$(3,4)$	$(3,6)$	$(4,7)$	$(5,3)$	$(5,7)$	$(6,7)$
$h(i,j)$	1	1	3	1	2	3	1	1	1	1	1
$q(i,j)$	12	10	8	3	3	4	5	12	3	4	10
$\overset{*}{f}(i,j)$	12	9	8	1	3	4	5	12	0	1	8
$\overset{*}{f^*}(i,j)$	12	9	6	3	3	4	5	10	0	3	8

Figure 4: The functions $h, q, \overset{*}{f}, \overset{*}{f^*}$

The maximum flow $\overset{*}{f}$ and the maximum flow of minimum cost $\overset{*}{f^*}$ obtained with the procedure AMVMCSF in bipartite static network $G = (N, A, c, u)$ are presented in Figure 4.

Applying the procedure ADSFEF we obtain the results which are indicated in Figure 5.

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

Figure 5: The results of procedure ADSFEF

The procedure ARPF generates the flow f_0 in network $R_0 = (V_0, E_0, u_0)$. The network R_0 with the flow f_0 is presented in Figure 6. With formula (12) we obtain $\overset{*}{w}_0 = (5+1) \cdot 21 - (1 \cdot 12 + 1 \cdot 9 + 3 \cdot 6 + 1 \cdot 3 + 2 \cdot 3 + 3 \cdot 4 + 1 \cdot 5 + 1 \cdot 10 + 1 \cdot 0 + 1 \cdot 3 + 1 \cdot 8) = 126 - 86 = 40$. A minimum $(1_0, 1_1, 1_2) - (7_3, 7_4, 7_5)$ cut in static network R_0 is $\left[Y_0, \bar{Y_0}\right] = (Y_0, \bar{Y_0}) \bigcup (\bar{Y_0}, Y_0)$ with $Y_0 = \{1_0, 1_1, 1_2, 2_2, 2_3, 3_1, 3_2, 3_3\}$ and $\bar{Y_0} = \{2_1, 4_4, 5_2, 5_3, 5_4, 6_2, 6_3, 6_4, 7_3, 7_4, 7_5\}$. Hence $\left[Y_0, \bar{Y_0}\right] = \{(1_0, 2_1), (2_2, 5_3), (2_2, 6_4), (2_3, 5_4), (3_1, 6_2), (3_1, 4_4), (3_2, 6_4)\} \cup \{(5_2, 3_3)\}$. We have $w_0 = f_0(Y_0, \bar{Y_0}) - f_0(Y_0, \bar{Y_0}) = 40 - 0 = 40 = u_0(Y_0, \bar{Y_0})$. Hence f_0 is a maximum flow, i.e. $\overset{*}{f}_0 = f_0$ and $\overset{*}{w}_0 = 40 = w_0$.

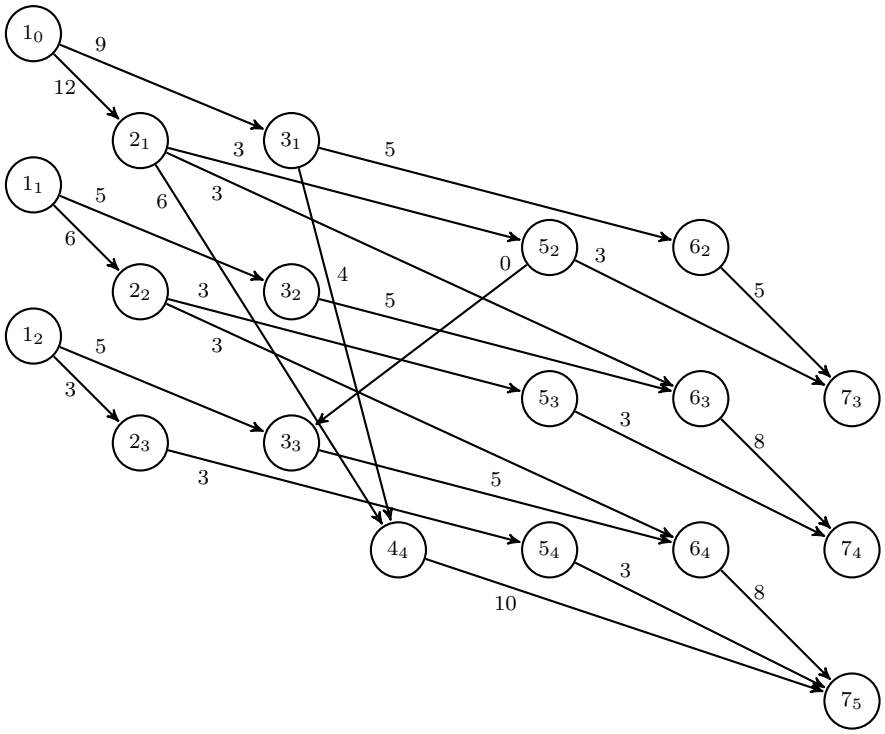

Figure 6: The network $R_0 = (V_0, E_0, u_0)$ with flow $\overset{*}{f}_0 = f_0$.

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

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

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

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

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