About Flow Problems in Networks with Node Capacities

LAURA CIUPALĂ
Department of Computer Science
University Transilvania of Braşov
Iuliu Maniu Street 50, Braşov
ROMANIA
laura_ciupala@yahoo.com

Abstract: - In this paper, we study different network flow problems in networks with node capacity. The literature on network flow problems is extensive, but these problems are described and studied in networks in which only arcs could have finite capacities. There are several applications that arise in practice and that can be reduced to a specific network flow problem in a network in which also nodes have limited capacity. This is the reason for focusing on maximum flow problem, minimum flow problem and minimum cost flow problem in a network with node capacities.

Key-Words: - Network flow; Network algorithms; Minimum cost flow problem Network algorithms; Maximum flow; Minimum flow; Minimum cost flow; Minimum cut

1 Introduction

Network flow problems are a group of network optimization problems with widespread and diverse applications. The literature on network flow problems is extensive. Over the past 50 years researchers have made continuous improvements to algorithms for solving several classes of problems. From the late 1940s through the 1950s, researchers designed many of the fundamental algorithms for network flow, including methods for maximum flow and minimum cost flow problems. In the next decades, there are many research contributions concerning improving the computational complexity of network flow algorithms by using enhanced data structures, techniques of scaling the problem data etc.

One of the reasons for which the maximum flow problem and that minimum cost flow problem were studied so intensively is the fact that they arise in a wide variety of situations and in several forms.

Although it has its own applications, the minimum flow problem was not dealt so often as the maximum flow ([1], [2], [3], [6], [18], [19], [20], [21], [23]) and the minimum cost flow problem ([1], [3], [5], [9], [10], [25]).

These network flow problems are intensively studied in networks in which only arcs could have finite capacities, but there are several applications that arise in practice and that can be reduced to a specific network flow problem in a network in which also nodes have capacity. For instance, the nodes might be airports with limited runaway capacities for takeoff and landings or might be switches in a communication network with a limited number of ports. In such networks, we are interested to determine (minimum, maximum or minimum cost) flows that satisfy both the arc and node capacities. In the next three sections, we will show how each of these three network flow problem can be solved in those particular network which allow nodes to have limited capacities.

2 Maximum Flow Problem

The maximum flow problem is one of the most fundamental problems in network flow theory and it was studied extensively. The importance of the maximum flow problem is due to the fact that it arises in a wide variety of situations and in several forms. Sometimes the maximum flow problem occurs as a subproblem in the solution of more difficult network problems, such as the minimum cost flow problem or the generalized flow problem. The maximum flow problem also arises in a number of combinatorial applications that on the surface might not appear to be maximum flow problems at all. The problem also arises directly in problems as far reaching as machine scheduling, the assignment of program modules to computer processors, the rounding of census data in order to retain the confidentiality of individual households, tanker scheduling and several others.

2.1. Standard Maximum Flow Problem

Without any loss of generality, we can consider a network with zero lower bounds, because any maximum flow problem in a network with positive lower bounds can be transformed in an equivalent maximum flow problem in a network with zero lower bounds (for details see [1]).

Let $G = (N, A, c, s, t)$ be a capacitated network with a nonnegative capacity $c(i, j)$ associated with each arc $(i, j) \in A$. We distinguish two special nodes in the network G: a source node s and a sink node t.

Let $n=|N|$, $m = |A|$ and $C = \max \{c(i, j) \mid (i, j) \in A\}$.

A *flow* is a function $f : A \rightarrow \mathbf{R}_+$ satisfying the next conditions:

$$f(s, N) - f(N, s) = v \quad (1)$$
$$f(i, N) - f(N, i) = 0, \; i \neq s,t \quad (2)$$
$$f(t, N) - f(N, t) = -v \quad (3)$$
$$0 \leq f(i, j) \leq c(i, j), \; (i, j) \in A \quad (4)$$

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 maximized.

For the maximum flow problem, a *preflow* is a function $f : A \rightarrow \mathbf{R}_+$ satisfying the next conditions:

$$f(i, N) - f(N, i) \geq 0, \; i \neq s,t \quad (5)$$
$$0 \leq f(i, j) \leq c(i, j), \; (i, j) \in A \quad (6)$$

Let f be a preflow. We define the *excess* of a node $i \in N$ in the following manner:

$$e(i) = f(i, N) - f(N, i) \quad (7)$$

Thus, for the maximum flow problem, for any preflow f, we have:

$$e(i) \geq 0, \; i \in N \setminus \{s, t\}.$$

We say that a node $i \in N \setminus \{s, t\}$ is *active* if $e(i) > 0$ and *balanced* if $e(i) = 0$.

A preflow f for which

$$e(i) = 0, \; i \in N \setminus \{s, t\}$$

is a flow. Consequently, a flow is a particular case of preflow.

For the maximum flow problem, the *residual capacity* $r(i, j)$ of any arc $(i, j) \in A$, with respect to a given preflow f, is given by

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

By convention, if $(i, j) \in A$ and $(j, i) \notin A$, then we add the arc (j, i) to the set of arcs A and we set $c(j, i) = 0$. The residual capacity $r(i, j)$ of the arc (i, j) represents the maximum amount of additional flow that can be sent from the node i to node j using both of the arcs (i, j) and (j, i).

The network $G_f = (N, A_f)$ consisting only of those arcs with strictly positive residual capacity is referred to as the *residual network* (with respect to the given preflow f).

In the residual network $G_f = (N, A_f)$ the *distance function* $d : N \rightarrow \mathbf{N}$ with respect to a given preflow f is a function from the set of nodes to the nonnegative integers.

We say that a distance function is *valid* if it satisfies the following validity conditions:

$$d(t) = 0$$
$$d(i) \leq d(j) + 1, \text{ for every arc } (i, j) \in A_f.$$

We refer to $d(i)$ as the distance label of node i.

Theorem 1.([1])*(a) If the distance labels are valid, the distance label $d(i)$ is a lower bound on the length of the shortest directed path from node i to sink node t in the residual network.*

(b) If $d(s) \geq n$, the residual network contains no directed path from the source node s to the sink node t.

A preflow is *blocking* if it saturates an arc on every path from s to t.

The maximum flow problem was first formulated and solved using the well known augmenting path algorithm by Ford and Fulkerson in 1956. Since then, two types of maximum flow algorithms have been developed: augmenting path algorithms and preflow algorithms:

1) The augmenting path algorithms maintain mass balance constraints at every node of the network other than the source node and the sink node. These algorithms incrementally augment flow along paths from the source node to the sink node. By determining the augmenting paths with respect to different selection rules, different algorithms were developed.
2) The preflow algorithms flood the network so that some nodes have excesses. These algorithms incrementally relieve flow from nodes with excesses by sending flow from the node forward toward the sink node or backward toward the source node. By imposing different rules for selecting nodes with excesses, different preflow algorithms

were obtained. These algorithms are more versatile and more efficient than the augmenting path algorithms.

2.2 Maximum Flow Problem in Networks with Node Capacities

Now we can formulate the maximum flow problem in networks with node capacities. Let $G = (N, A, c, s, t)$ be a network on which we define a new function: $nc: A \rightarrow R_+$. The node capacity function, nc, associates to each node i a positive value $nc(i)$ that represents the maximum amount of flow that can pass through node i.

In a network with node capacities $G = (N, A, c, nc, s, t)$, a feasible flow is a function $f: A \rightarrow R_+$ satisfying conditions (1) - (4) and:

$$f(s, N) \leq nc(s) \quad (8)$$
$$f(N, i) \leq nc(i) \quad \forall i \in N \setminus \{s\} \quad (9)$$

The maximum flow problem in networks with node capacities consists in determining a flow of which value is maximized.

We will solve the maximum flow problem in the network with node capacities $G = (N, A, c, nc, s, t)$ by determining a standard maximum flow in a standard network $G' = (N', A', c', s', t)$ that is equivalent to the original network G. Network G' is determined in the following manner:

$$N' = N_1 \cup N_2$$
$$N_1 = \{i' \mid i \in N\}$$
$$N_2 = \{i'' \mid i \in N\}$$
$$A' = A_1 \cup A_2$$
$$A_1 = \{(i'', j') \mid (i, j) \in A\}$$
$$A_2 = \{(i', i'') \mid i \in N\}$$
$$l'(i'', j') = l(i, j), \forall (i'', j') \in A_1$$
$$l'(i', i'') = 0, \forall (i', i'') \in A_2$$
$$c'(i'', j') = c(i, j), \forall (i'', j') \in A_1$$
$$c'(i', i'') = nc(i), \forall (i', i'') \in A_2$$

Theorem 2. There is a feasible flow f' of value v' in network G' if and only if there is a feasible flow f of value $v = v'$ in network G.

Proof. In network G', let f' be a feasible of value v'. We determine the flow f in the network G with node capacities in the following manner:

$$f(i, j) = f'(i'', j'), \forall (i, j) \in A \quad (10)$$

Obviously, f is a feasible flow of value $v = v'$ in network G.

Reciprocally, let f be a feasible of value v in the network G. We establish the flow f' in the network G' in the following manner:

$$f'(i'', j') = f(i, j), \forall (i, j) \in A_1$$
$$f'(i', i'') = f(N, i), \forall (i', i'') \in A_2$$

Obviously, f' is a feasible flow of value $v' = v$ in network G'.

A consequence of the theorem 2 is the following:

Theorem 3. There is a maximum flow f' of value v' in network G' if and only if there is a maximum flow f of value $v = v'$ in network G.

Consequently, the maximum flow problem in a network with node capacities $G = (N, A, c, nc, s, t)$ can be solved by determining a maximum flow f' in a transformed standard network $G' = (N', A', c', s', t)$. From the maximum flow f', we establish easily the maximum flow f in G by using the relations (10). For finding the maximum flow f' in G', we can use any algorithm for the maximum flow problem: any augmenting path algorithm (see [1], [3]) or any preflow algorithm (see [1], [3], [18], [19], [20], [21]).

3 Minimum Flow Problem

Although it has its own applications, the minimum flow problem was not dealt so often as the maximum flow ([1], [2], [3], [6], [18], [19], [20], [21], [23]) and the minimum cost flow problem ([1], [3], [5], [9], [10], [25]).
There are many problems that occur in economy that can be reduced to minimum flow problems. Examples can be found in [15] and [16].

3.1 Standard Minimum Flow Problem

Given a capacitated network $G = (N, A, l, c, s, t)$ with a nonnegative capacity $c(i, j)$ and with a nonnegative lower bound $l(i, j)$ associated with each arc $(i, j) \in A$. We distinguish two special nodes in the network G: a source node s and a sink node t.

Let $n = |N|$, $m = |A|$ and $C = \max \{ c(i, j) \mid (i, j) \in A \}$.

A *flow* is a function $f : A \rightarrow \mathbf{R}_+$ satisfying the next conditions:

$$f(s, N) - f(N, s) = v \quad (11)$$
$$f(i, N) - f(N, i) = 0, i \neq s,t \quad (12)$$
$$f(t, N) - f(N, t) = -v \quad (13)$$
$$l(i, j) \leq f(i, j) \leq c(i, j), (i, j) \in A \quad (14)$$

for some $v \geq 0$, where

$$f(i, N) = \Sigma_j f(i, j), i \in N$$

and

$$f(N, i) = \Sigma_j f(j, i), i \in N.$$

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 minimized. So, the objective is to send as little flow as possible through the network G from the source node s to the sink node t.

For the minimum flow problem, a *preflow* is a function $f : A \rightarrow \mathbf{R}_+$ satisfying the next conditions:

$$f(i, N) - f(N, i) \leq 0, i \neq s,t \quad (15)$$
$$l(i, j) \leq f(i, j) \leq c(i, j), (i, j) \in A \quad (16)$$

Let f be a preflow. We define the *deficit* of a node $i \in N$ in the following manner:

$$e(i) = f(i, N) - f(N, i) \quad (17)$$

Thus, for the minimum flow problem, for any preflow f, we have:

$$e(i) \leq 0, i \in N \setminus \{s, t\}.$$

We say that a node $i \in N \setminus \{s, t\}$ is *active* if $e(i) < 0$ and *balanced* if $e(i) = 0$.

A preflow f for which

$$e(i) = 0, i \in N \setminus \{s, t\}$$

is a flow. Consequently, a flow is a particular case of preflow.

For the minimum flow problem, the *residual capacity* $r(i, j)$ of any arc $(i, j) \in A$, with respect to a given preflow f, is given by

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

By convention, if $(i, j) \in A$ and $(j, i) \notin A$, then we add the arc (j, i) to the set of arcs A and we set $l(j, i) = 0$ and $c(j, i) = 0$. The residual capacity $r(i, j)$ of the arc (i, j) represents the maximum amount of flow from the node i to node j that can be cancelled by modifying the flow on both of the arcs (i, j) and (j, i).

The network $G_f = (N, A_f)$ consisting only of those arcs with strictly positive residual capacity is referred to as the *residual network* (with respect to the given preflow f).

In the residual network $G_f = (N, A_f)$ the *distance function* $d : N \rightarrow \mathbf{N}$ with respect to a given preflow f is a function from the set of nodes to the nonnegative integers.

We say that a distance function is *valid* if it satisfies the following validity conditions:

$$d(s) = 0$$
$$d(j) \leq d(i) + 1, \text{ for every arc } (i, j) \in A_f.$$

We refer to $d(i)$ as the distance label of node i.

Theorem 4. *(a) If the distance labels are valid, the distance label d(i) is a lower bound on the length of the shortest directed path from the source node s to node i in the residual network.*

(b) If d(t) ≥ n, the residual network contains no directed path from the source node s to the sink node t.

Proof. (a) Let $P = (s=i_1, i_2, \ldots, i_k, i_{k+1}=i)$ be any path of length k from node s to node i in the residual network. The validity conditions imply that:

$$d(i_2) \leq d(i_1) + 1 = d(s)+1 = 1$$
$$d(i_3) \leq d(i_2) + 1 \leq 2$$
$$d(i_4) \leq d(i_3) + 1 \leq 3$$
$$\ldots.$$
$$d(i_{k+1}) \leq d(i_k) + 1 \leq k.$$

(b) We proved that $d(t)$ is a lower bound on the length of the shortest path from the source node s to the sink node t in the residual network and we know that no directed path can contain more than $(n-1)$ arcs. Consequently, if $d(t) \geq n$, then the residual network contains no directed path from s to t.

We say that the distance labels are *exact* if for each node i, $d(i)$ equals the length of the shortest path from node s to node i in the residual network.

We refer to an arc (i, j) from the residual network as an *admissible arc* if $d(j) = d(i) + 1$; otherwise it is *inadmissible*.

We refer to a node i with $e(i) < 0$ as an *active* node. We adopt the convention that the source node and the sink node are never active.

3.2.Minimum Flow Problem in Networks with Node Capacities

Now we can formulate the minimum flow problem in networks with node capacities. Let $G = (N, A, l, c, s, t)$ be a network on which we define a new function: $nc{:}A \rightarrow R_+$. The node capacity function, nc, associates to each node i a positive value $nc(i)$ that represents the maximum amount of flow that can pass through node i.

In a network with node capacities $G = (N, A, l, c, nc, s, t)$, a feasible flow is a function $f{:}A \rightarrow R_+$ satisfying conditions (11) - (14) and:

$$f(s, N) \leq nc(s) \tag{18}$$
$$f(N, i) \leq nc(i) \quad \forall i \in N \setminus \{s\} \tag{19}$$

The minimum flow problem in networks with node capacities consists in determining a flow of which value is minimized.

We will solve the minimum flow problem in the network with node capacities $G = (N, A, l, c, nc, s, t)$ by determining a standard minimum flow in a standard network $G' = (N', A', l', c', s', t)$ that is equivalent to the original network G. Network G' is determined in the following manner:

$$N' = N_1 \cup N_2$$
$$N_1 = \{i' \mid i \in N\}$$
$$N_2 = \{i'' \mid i \in N\}$$
$$A' = A_1 \cup A_2$$
$$A_1 = \{(i'', j') \mid (i, j) \in A\}$$
$$A_2 = \{(i', i'') \mid i \in N\}$$
$$l'(i'', j') = l(i, j), \ \forall (i'', j') \in A_1$$
$$l'(i', i'') = 0, \ \forall (i', i'') \in A_2$$
$$c'(i'', j') = c(i, j), \ \forall (i'', j') \in A_1$$
$$c'(i', i'') = nc(i), \ \forall (i', i'') \in A_2$$

Theorem 5. There is a feasible flow f' of value v' in network G' if and only if there is a feasible flow f of value $v = v'$ in network G.

Proof. In network G', let f' be a feasible of value v'. We determine the flow f in the network G with node capacities in the following manner:

$$f(i, j) = f'(i'', j'), \ \forall (i, j) \in A \tag{20}$$

Obviously, f is a feasible flow of value $v = v'$ in network G.

Reciprocally, let f be a feasible of value v in the network G. We establish the flow f' in the network G' in the following manner:

$$f'(i'', j') = f(i, j), \ \forall (i, j) \in A_1$$
$$f'(i', i'') = f(N, i), \ \forall (i', i'') \in A_2$$

Obviously, f' is a feasible flow of value $v' = v$ in network G'.

A consequence of the theorem 5 is the following:

Theorem 6. There is a maximum flow f' of value v' in network G' if and only if there is a maximum flow f of value $v = v'$ in network G.

Consequently, the minimum flow problem in a network with node capacities $G = (N, A, l, c, nc, s, t)$ can be solved by determining a minimum flow f' in a transformed standard network $G' = (N', A', l', c', s', t'')$. From the minimum flow f', we establish easily the minimum flow f in G by using the relations (20). For finding the minimum flow f' in G', we can use any algorithm for the minimum flow problem: any decreasing path algorithm (see [15], [16]), any preflow algorithm (see [11], [12], [13], [15], [16]) or even the minimax algorithm (see [3], [14]).

4 Minimum Cost Flow Problem

The minimum cost flow problem, as well as one of its special cases which is the maximum flow problem, is one of the most fundamental problems in network flow theory and it was studied extensively. The importance of the minimum cost flow problem is also due to the fact that it arises in almost all industries, including agriculture, communications, defence, education, energy, health care, medicine, manufacturing, retailing and transportation. Indeed, minimum cost flow problem are pervasive in practice.

4.1.Standard Minimum Cost Flow Problem

Let $G = (N, A)$ be a directed graph, defined by a set N of n nodes and a set A of m arcs. Each arc $(i, j) \in A$ has a capacity $c(i, j)$ and a cost $b(i, j)$. We associate with each node $i \in N$ a number $v(i)$ which indicates its supply or demand depending on whether $v(i) > 0$ or $v(i) < 0$. In the directed network $G = (N, A, c, b, v)$, the minimum cost flow problem is to determine the flow $f(i, j)$ on each arc $(i, j) \in A$ which

$$\text{minimize} \quad \sum_{(i, j) \in A} b(i, j) f(i, j) \tag{21}$$

subject to

$$\sum_{j|(i,j)\in A} f(i,j) - \sum_{j|(j,i)\in A} f(j,i) = v(i), \forall i \in N \quad (22)$$

$$0 \le f(i,j) \le c(i,j), \forall (i,j) \in A. \quad (23)$$

where $\sum_{i\in N} v(i) = 0$.

A flow f satisfying the conditions (22) and (23) is referred to as a *feasible flow*.

Let C denote the largest magnitude of any supply/demand or finite arc capacity, that is

$$C = \max(\max\{v(i) \mid i\in N\}, \max\{c(i,j) \mid (i,j)\in A, c(i,j)<\infty\})$$

and let B denote the largest magnitude of any arc cost, that is

$$B = \max\{b(i,j) \mid (i,j)\in A\}.$$

For solving a minimum cost flow problem, several algorithms were developed from the primal-dual algorithm proposed by Ford and Fulkerson in 1962 to the polynomial-time cycle-cancelling algorithms described by Sokkalingam, Ahuja and Orlin in 2001. Most of these algorithms work on the residual network. So, before describing them, we have to introduce some notions and some assumptions.

The residual network $G(f) = (N, A(f))$ corresponding to a flow f is defined as follows. We replace each arc $(i, j)\in A$ by two arcs (i, j) and (j, i). The arc (i, j) has the cost $b(i, j)$ and the residual capacity $r(i, j) = c(i, j) - f(i, j)$ and the arc (j, i) has the cost $b(j, i) = -b(i, j)$ and the residual capacity $r(j, i) = f(i, j)$. The residual network consists only of arcs with positive residual capacity.

We shall assume that the minimum cost flow problem satisfies the following assumptions:

Assumption 1. The network is directed.

This assumption can be made without any loss of generality. In [1] it is shown that we can always fulfil this assumption by transforming any undirected network into a directed network.

Assumption 2. All data (cost, supply/demand and capacity) are integral.

This assumption is not really restrictive in practice because computers work with rational numbers which we can convert into integer numbers by multiplying by a suitably large number.

Assumption 3. The network contains no directed negative cost cycle of infinite capacity.

If the network contains any such cycles, there are flows with arbitrarily small costs.

Assumption 4. All arc costs are nonnegative.

This assumption imposes no loss of generality since the arc reversal transformation described in [1] converts a minimum cost flow problem with negative arc costs to one with nonnegative arc costs. This transformation can be done if the network contains no directed negative cost cycle of infinite capacity.

Assumption 5. The supplies/demands at the nodes satisfy the condition $\sum_{i\in N} v(i) = 0$ and the minimum cost flow problem has a feasible solution.

Assumption 6. The network contains an uncapacitated directed path (i.e. each arc in the path has infinite capacity) between every pair of nodes.

We impose this condition by adding artificial arcs $(1, i)$ and $(i, 1)$ for each $i\in N$ and assigning a large cost and infinite capacity to each of these arcs. No such arc would appear in a minimum cost solution unless the problem contains no feasible solution without artificial arcs.

We associate a real number $\pi(i)$ with each node $i\in N$. We refer to $\pi(i)$ as the *potential* of node i. These node potentials are generalizations of the concept of distance labels.

For a given set of node potentials π, we define the reduced cost of an arc (i, j) as

$$b^{\pi}(i,j) = b(i,j) - \pi(i) + \pi(j).$$

The reduced costs are applicable to the residual network as well as to the original network.

Theorem 6. ([1]) (a) For any directed path P from node h to node k we have

$$\sum_{(i,j)\in P} b^{\pi}(i,j) = \sum_{(i,j)\in P} b(i,j) - \pi(h) + \pi(k)$$

(b) For any directed cycle W we have

$$\sum_{(i,j)\in W} b^{\pi}(i,j) = \sum_{(i,j)\in W} b(i,j).$$

Theorem 7. (**Negative Cycle Optimality Conditions)** ([1]) A feasible solution f is an optimal solution of the minimum cost flow problem if and

only if there is no negative cycle in the residual network $G(f)$.

Theorem 8. (**Reduced Costs Optimality Conditions)** ([1]) A feasible solution f is an optimal solution of the minimum cost flow problem if and only if some set of node potentials π satisfy the following reduced cost optimality conditions:

$b^{\pi}(i, j) \geq 0$ for every arc (i, j) in the residual network $G(f)$.

Theorem 9. (Complementary Slackness Optimality Conditions) ([1]) A feasible solution f is an optimal solution of the minimum cost flow problem if and only if for some set of node potentials π, the reduced cost and flow values satisfy the following complementary slackness optimality conditions for every arc $(i, j) \in A$:

$$\text{If } b^{\pi}(i, j) > 0, \text{ then } f(i, j) = 0 \quad (24)$$
$$\text{If } 0 < f(i, j) < c(i, j), \text{ then } b^{\pi}(i, j) = 0 \quad (25)$$
$$\text{If } b^{\pi}(i, j) < 0, \text{ then } f(i, j) = c(i, j) \quad (26)$$

A *pseudoflow* is a function $f : A \rightarrow \mathbf{R}_+$ satisfying the only conditions (23).

For any pseudoflow f, we define the *imbalance* of node i as

$$e(i) = v(i) + f(N, i) - f(i, N), \quad \text{for all } i \in N.$$

If $e(i) > 0$ for some node i, we refer to $e(i)$ as the *excess* of node i; if $e(i) < 0$, we refer to $-e(i)$ as the *deficit* of node i. If $e(i) = 0$ for some node i, we refer to node i as the *balanced*.

The residual network corresponding to a pseudoflow is defined in the same way that we define the residual network for a flow.

The optimality conditions can be extended for pseudoflows. A pseudoflow f^* is optimal if there are some set of node potentials π such that the following reduced cost optimality conditions are satisfied:

$b^{\pi}(i, j) \geq 0$ for every arc (i, j) in the residual network $G(f^*)$.

We refer to a flow or a pseudoflow f as ε-*optimal* for some $\varepsilon > 0$ if for some node potentials π, the pair (f, π) satisfies the following ε-*optimality conditions*:

$$\text{If } b^{\pi}(i, j) > \varepsilon, \text{ then } f(i, j) = 0 \quad (27)$$
$$\text{If } -\varepsilon \leq b^{\pi}(i, j) \leq \varepsilon, \text{ then}$$
$$0 \leq f(i, j) \leq c(i, j) \quad (28)$$
$$\text{If } b^{\pi}(i, j) < -\varepsilon, \text{ then } f(i, j) = c(i, j) \quad (29)$$

These conditions are relaxations of the (exact) complementary slackness optimality conditions (24) - (26) and they reduce to complementary slackness optimality conditions when $\varepsilon = 0$.

The algorithms for determining a minimum cost flow rely upon the optimality conditions stated by Theorems 7, 8 and 9.

The basic algorithms for minimum cost flow can be divided into two classes: those that maintain feasible solutions and strive toward optimality and those that maintain infeasible solutions that satisfy optimality conditions and strive toward feasibility (for details see [1]).

Algorithms from the first class are: the cycle-canceling algorithm and the out-of-kilter algorithm. The cycle-canceling algorithm maintains a feasible flow at every iteration, augments flow along negative cycle in the residual network and terminates when there is no more negative cycle in the residual network, which means (from Theorem 7) that the flow is a minimum cost flow. The out-of-kilter algorithm maintains a feasible flow at every iteration and augments flow along shortest path in order to satisfy the optimality conditions.

Algorithms from the second class are: the successive shortest path algorithm and primal-dual algorithm. The successive shortest path algorithm maintains a pseudoflow that satisfies the optimality conditions and augments flow along shortest path from excess nodes to deficit nodes in the residual network in order to convert the pseudoflow into an optimal flow. The primal-dual algorithm also maintains a pseudoflow that satisfies the optimality conditions and solves maximum flow problems in order to convert the pseudoflow into an optimal flow.

Starting from the basic algorithms for minimum cost flow, several polynomial-time algorithms were developed. Most of them were obtained by using the scaling technique. By capacity scaling, by cost scaling or by capacity and cost scaling, the following polynomial-time algorithms were developed: capacity scaling algorithm, cost scaling algorithm, double scaling algorithm, repeated capacity scaling algorithm and enhanced capacity scaling algorithm (for details see [1], [10]).

Another approach for obtaining polynomial-time algorithms is to select carefully the negative cycles in the cycle-canceling algorithm (for details see [24]).

4.2.Minimum Cost Flow Problem in Networks with Node Capacities

Now we can formulate the minimum cost flow problem in networks with node capacities. Let $G=(N, A, c, b, v)$ be a network on which we define a new function: $nc:A\rightarrow R_+$. The node capacity function, nc, associates to each node i a positive value $nc(i)$ that represents the maximum amount of flow that can pass through node i.

In a network with node capacities $G = (N, A, c, nc, b, v)$, a feasible flow is a function f: $A\rightarrow R_+$ satisfying the following conditions:

$$\sum_{j|(i,j)\in A} f(i,j) - \sum_{j|(j,i)\in A} f(j,i) = v(i), \forall i \in N \quad (30)$$

$$0 \le f(i,j) \le c(i,j), \forall (i,j) \in A. \quad (31)$$

$$f(s, N) \le nc(s) \quad (32)$$

$$f(N, i) \le nc(i) \quad \forall i \in N\setminus\{s\} \quad (33)$$

where $\sum_{i\in N} v(i) = 0$.

The minimum cost flow problem in networks with node capacities consists in determining a feasible flow of which cost is minimized, which means finding a function f: $A\rightarrow R_+$ that satisfies conditions (30) - (33) and:

$$\text{minimize} \sum_{(i,j)\in A} b(i,j) f(i,j)$$

We will solve the minimum cost flow problem in a network with node capacities $G=(N, A, c, nc, b, v)$ by determining a standard minimum cost flow in a standard network $G'= (N', A', c', b', v')$ that is equivalent to the original network G. Network G' is determined in the following manner:

$N'=N_1\cup N_2$
$N_1=\{i' \mid i\in N\}$
$N_2=\{i'' \mid i\in N\}$
$A'=A_1\cup A_2$
$A_1=\{(i'', j') \mid (i,j)\in A\}$
$A_2=\{(i', i'') \mid i\in N\}$
$c'(i'', j') = c(i, j), \forall (i'', j')\in A_1$
$c'(i', i'') = nc(i), \forall (i', i'')\in A_2$
$b'(i'', j') = b(i, j), \forall (i'', j')\in A_1$
$b'(i',i'') = 0, \forall (i',i'')\in A_2$
$v'(i') = 0$, for any $i'\in N_1$ such that $v(i) \le 0$
$v'(i') = v(i)$, for any $i'\in N_1$ such that $v(i) > 0$
$v'(i'') = 0$, for any $i''\in N_2$ such that $v(i) \ge 0$
$v'(i'') = v(i)$, for any $i''\in N_2$ such that $v(i) < 0$

Obviously, $\sum_{i\in N'} v'(i) = 0$.

Theorem 10. There is a feasible flow f', with the cost $z(f') = \sum_{(i,j)\in A} b(i,j) f'(i,j)$ in network $G'= (N', A', c', b', v')$ if and only if there is a feasible flow f of cost $z(f) = \sum_{(i,j)\in A} b(i,j) f(i,j)$ in network $G = (N, A, c, nc, b, v)$, with $z(f) = z(f')$.

Proof. In network G', let f' be a feasible of cost $z(f')$. We determine the flow f in the network G with node capacities in the following manner:

$$f(i,j)=f'(i'',j'), \forall (i,j)\in A \quad (34)$$

Obviously, f is a feasible flow of cost $z(f) = z(f')$ in network G.

Reciprocally, let f be a feasible of cost $z(f)$ in the network G. We establish the flow f' in the network G' in the following manner:

$$f'(i'',j')= f(i,j), \forall (i,j)\in A_1$$
$$f'(i',i'') =f(N,i), \forall (i',i'')\in A_2$$

Obviously, f' is a feasible flow of cost $z(f') = z(f)$ in network G'.

A consequence of the theorem 10 is the following:

Theorem 11. There is a minimum cost flow f', with the cost $z(f')= \sum_{(i,j)\in A} b(i,j) f'(i,j)$ in network $G'= (N', A', c', b', v')$ if and only if there is a minimum cost flow f with cost $z(f) = \sum_{(i,j)\in A} b(i,j) f(i,j)$ in network $G=(N, A, c, nc, b, v)$, with $z(f) = z(f')$.

Consequently, the minimum cost flow problem in a network with node capacities $G = (N, A, c, nc, b, v)$, can be solved by determining a minimum cost flow f' in a transformed standard network $G'= (N', A', c', b', v')$. From the minimum cost flow f', we establish easily the minimum cost flow f in G by using the relations (34). For finding the minimum cost flow f' in G', we can use any algorithm for the minimum cost flow problem.

5 Conclusion

In this paper, we studied three different network flow problems in networks with node capacity: minimum flow problem, maximum flow problem and minimum cost flow problem. The literature on network flow problems is extensive, but these problems are described and studied in networks in which only arcs could have finite capacities. There are several applications that arise in practice and that can be reduced to a specific network flow problem in a network in which also nodes have limited capacity. For instance, the nodes might be airports with limited runaway capacities for takeoff and landings or might be switches in a communication network with a limited number of ports. In such networks, we are interested to determine (minimum, maximum or minimum cost) flows that satisfy both the arc and node capacities. This is the reason for focusing on maximum flow problem, minimum flow problem and minimum cost flow problem in a network with node capacities. We solved all these problems by transforming them into corresponding standard flow problems. For a given network with node capacities, we determined a transformed network without node capacities in different ways for different flow problems. But all these transformed networks have $2n$ nodes and $n + m$ arcs if the initial network has n nodes and m arcs. This means that any of these three flow problems in a network with node capacities can be solved by applying any algorithm for the corresponding standard flow problem in the transformed network and the time complexity is not changed.

References:

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

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

[3] J. Bang-Jensen and G. Gutin, *Digraphs: Theory, Algorithms and Applications*, Springer-Verlag, London, 2001.

[4] J. Barros, S.D. Servetto, *Network Information Flow with Correlated Sources*, IEEE Transactions on Information Theory, 52(1), 155-170, 2006.

[5] L. Ciupală, About Minimum Cost Flow Problem in Networks with Node Capacities, *Proceedings of the 13th WSEAS International Conference on Computers*, 2009, pp. 67-70.

[6] L. Ciupală, About wave algorithms for network flow problem, *International Journal of Computers* vol. 2 nr.3, 2008, pp. 291-300.

[7] L. Ciupală, The wave preflow algorithm for the minimum flow problem, *Proceedings of the 10th WSEAS International Conference on Mathematical and Computational Methods in Science and Engineering*, 2008, pp. 473-476.

[8] L. Ciupală, About Minimum Flow Problem in Networks With Node Capacities, BALCOR 2007, Belgrad, Serbia, 2007, *Proceedings of the 8th Balkan Conference on Operation Research*, 2007, pp. 151-154.

[9] L. Ciupală, A deficit scaling algorithm for the minimum flow problem, *Sadhana* Vol.31, No. 3, 2006, pp.1169-1174.

[10] L. Ciupală, A scaling out-of-kilter algorithm for minimum cost flow, *Control and Cybernetics* Vol.34, No.4, 2005, pp. 1169-1174.

[11] L. Ciupală and E. Ciurea, A highest-label preflow algorithm for the minimum flow problem, *Proceedings of the 11th WSEAS International Conference on Computers*, 2007, pp. 565-569.

[12] L. Ciupală and E. Ciurea, About preflow algorithms for the minimum flow problem, *WSEAS Transactions on Computer Research* vol. 3 nr.1, January 2008, pp. 35-41.

[13] L. Ciupală and E. Ciurea, An algorithm for the minimum flow problem, *The Sixth International Conference of Economic Informatics*, 2003, pp. 167-170.

[14] L. Ciupală and E. Ciurea, An approach of the minimum flow problem, *The Fifth International Symposium of Economic Informatics*, 2001, pp. 786-790.

[15] E. Ciurea and L. Ciupală, Sequential and parallel algorithms for minimum flows, *Journal of Applied Mathematics and Computing* Vol.15, No.1-2, 2004, pp. 53-78.

[16] E. Ciurea and L. Ciupală, Algorithms for minimum flows, *Computer Science Journal of Moldova* Vol.9, No.3(27), 2001, pp. 275-290.

[17] A. Deshpande, S. Patkar and H. Narayanan: Submodular Theory Based Approaches For Hypergraph Partitioning *WSEAS Transactions on Circuit and Systems*, Issue 6, Volume 4, 2005, pp. 647-655.

[18] V. Goldberg and R. E. Tarjan, A New Approach to the Maximum Flow Problem, *Journal of ACM* Vol.35, 1988, pp. 921-940.

[19] S. Fujishige, A maximum flow algorithm using MA ordering, *Operation Research Letters* 31, No. 3, 2003, pp. 176-178.

[20] S. Fujishige, S. Isotani, New maximum flow algorithms by MA orderings and scaling, *Journal of the Operational Research Society of Japan* 46, No. 3, 2003, pp. 243-250.

[21] S. Kumar, P. Gupta, An incremental algorithm for the maximum flow problem, *Journal of Mathematical Modelling and Algorithms* 2, No.1, 2003, pp. 1-16.

[22] S. Patkar, H. Sharma and H. Narayanan: Efficient Network Flow based Ratio-cut Netlist Hypergraph Partitioning, *WSEAS Transactions on Circuits and Systems* vol. 3, no. 1, January 2004, pp. 47-53

[23] A. Schrijver, On the history of the transportation and maximum flow problems, *Mathematical Programming* 91, No.3, 2002, pp. 437-445.

[24] P.T. Sokkalingam, R. Ahuja, J. Orlin, J., New Poynomial-Time Cycle-Canceling Algorithms for Minimum Cost Flows, http://web.mit.edu/jorlin/www/papers.html, 2001.

[25] K.D. Wayne, *A polynomial Combinatorial Algorithm for Generalized Minimum Cost Flow*, Mathematics of Operations Research, 2002, pp. 445-459.