Article

A bit-capacity scaling algorithm for the constrained minimal cost network flow problem

Muhammad Tlas
Scientific Services Department, Atomic Energy Commission, P. O. Box 6091, Damascus, Syria
E-mail: pscientific31@aec.org.sy

Received 11 January 2021; *Accepted 20 February 2021*; *Published 1 June 2021*

Abstract
A polynomial time algorithm for solving the minimum-cost network flow problem has been proposed in this paper. This algorithm is mainly based on the binary representation of capacities; it solves the minimum-cost flow problem in directed graph of n nodes and m directed arcs as a sequence of $O(n^2)$ shortest path problems on residual networks. The algorithm runs in $O(n^2mr)$ time, where r is the smallest integer greater than or equal to $\mathrm{Log}_2 B$, and B is the largest arc capacity of the network. A generalization of this proposed algorithm has been also performed in order to solve the minimum-cost flow problem in a directed network subject to non-negative lower bound on the flow vector. A formulation of both the transportation and the assignment problems, as a minimal cost network flow problem has been also performed. A numerical example has been inserted to illustrate the use of the proposed method.

Keywords minimal cost flow problem; bit-scaling algorithm; polynomial time algorithm; augmenting path method; transportation problem; assignment problem.

Network Biology
ISSN 2220-8879
URL: http://www.iaees.org/publications/journals/nb/online-version.asp
RSS: http://www.iaees.org/publications/journals/nb/rss.xml
E-mail: networkbiology@iaees.org
Editor-in-Chief: WenJun Zhang
Publisher: International Academy of Ecology and Environmental Sciences

1 Introduction

The minimum-cost network flow problem is a generalization of the maximum flow problem. It is one of the most fundamental network flow problems. The problem has applications to a remarkably wide range of fields, including chemistry, physics, computer networking, most branches of engineering, manufacturing, public policy and social systems, scheduling and routing, telecommunications, and transportation (Ahuja et al., 1993). Different approaches have been proposed to solve the minimum- cost flow problem. The classical algorithm for minimum-cost flow problems is the Fulkerson's Out of Kilter algorithm (Ford and Fulkerson, 1962; Zhang, 2017) which is essentially a primal method and runs in exponential time in the worst case.

There are a number of different polynomial time algorithms for the minimum-cost flow problem as: the capacity-scaling approach of Edmonds and Karp (1972) with running time $O\left(\left(m \log B\right)\left(m + n \log n\right)\right)$ on

networks with n nodes, m arcs and maximum arc capacity B, the cost-scaling approach of Goldberg and Tarjan (1990) which runs in $O\left(nm\log\frac{n^2}{m}\log nC\right)$ time on networks with maximum arc cost magnitude C, the double scaling of both costs and capacities of Ahuja et al. (1992) which solves the problem in $O\left(nm\log\log B\log nC\right)$ time in the worst case, the strongly polynomial time of Orlin (1991) with running time bounded by $O\left(\left(m\log n\right)\left(m+n\log n\right)\right)$ and the primal network simplex polynomial time of Orlin (1996) which runs in $O\left(\min\left(n^2m\log nC, n^2m^2\log n\right)\right)$ time at most.

One important idea is common to all first four of these algorithms, that of scaling or successive approximation. Scaling algorithms work by solving a sequence of sub-problems whose numeric parameters more and more closely approximate those of the original problem. A solution for one sub-problem helps to solve the next sub-problem in the sequence.

In this paper, an efficient polynomial algorithm is presented for determining the minimum-cost flow in a network with an upper bound $O(n^2m\ r)$ on the number of arithmetic operations, where n, m are the numbers of nodes and arcs of the network respectively and r is the smallest integer greater than or equal to $\log B$, where B is the largest arc capacity of the network. The algorithm is basically based on the binary representation of capacities; it solves the minimum- cost flow problem as a sequence of $O(n^2)$ shortest path problems on residual networks (Zhang, 2016).

A generalization of this proposed algorithm has been also performed, in this paper, in order to solve a minimum-cost flow problem in a network subject to nonnegative lower bound on the flow vector.

A formulation of the transportation and the assignment problems as a minimal cost network flow problem has been also achieved in this paper.

2 Preliminary Problem

In this section we define the minimum-cost flow problem and introduce the terminology and notation used throughout the paper.

2.1 Minimum-cost flow problem statement

Consider a directed graph (digraph) $G=(V,E)$ consisting of a set V of nodes and a set E of arcs whose elements are ordered pairs of distinct nodes. A directed network is a directed graph with numerical values attached to its nodes and arcs. Let $n=|V|$ and $m=|E|$, we associate with each arc $k=(i,j)\in E$ a nonnegative integral unit cost c_k and a nonnegative integral capacity b_k. Frequently, we distinguish two special nodes in a graph, the source s and the sink t. An arc $k=(i,j)\in E$ has two end points i and j, the node i is called the tail and node j is called the head of arc k. The arc $k=(i,j)$ is said to emanate from node i, the arc $k=(i,j)$ is an outgoing arc of node i and an incoming arc of node j. The arc adjacency list of node i, $E(i)$, is defined as the set of arcs emanating from node i, i.e., $E(i)=\{k=(i,j)\in E: j\in V\}$. The degree of a node is the number of incoming and outgoing arcs at that node.

The total cost of a flow x from source node s to sink node t is $\sum_{k\in E} c_k x_k$ and let its value be z. The problem is to find a maximum flow of minimum-cost among the source node s and the sink node t with value z^*.

A flow is a value x on arcs satisfying the following constraints:

$x_{ij} \le b_{ij} \quad \forall (i,j) \in E$ (capacity constraint),

$x_{ij} = -x_{ji} \quad \forall (i,j) \in E$ (flow anti-symmetry constraint) and

$\sum_{j\in V} x_{ij} = 0 \quad \forall i \in V \setminus \{s,t\}$ (flow conservation constraint).

Call a flow x extreme if it is of minimum-cost among flows with value x_{st}.

2.2 Labeling function

A labeling function (potential function) u is defined as a function from nodes to the real numbers, i.e., $u: V \to \mathbf{R}$, where $\mathbf{R}$ denotes real numbers. The x and u are called compatible if the flow x and labeling function u together satisfy the following conditions: for arc $(i,j) \in E$

If $\bar{c}_{ij} = c_{ij} - (u_j - u_i) > 0$, then $x_{ij} = 0$,

If $\bar{c}_{ij} = c_{ij} - (u_j - u_i) < 0$, then $x_{ij} = b_{ij}$,

If $\bar{c}_{ij} = c_{ij} - (u_j - u_i) = 0$, then $0 \le x_{ij} \le b_{ij}$,

$\bar{c}_{ij} = c_{ij} - (u_j - u_i)$ is called the reduced cost of the arc (i,j).

2.3 Residual network

A residual network $G(x)$ corresponding to a feasible flow x is defined as follows: for arc $(i,j) \in E$

If $x_{ij} < b_{ij}$, then there is a forward arc (direct arc) (i,j) has flow $x_{ij} \ge 0$ and reduced cost $\bar{c}_{ij} = c_{ij} - (u_j - u_i) \ge 0$,

If $x_{ij} = b_{ij}$, then the arc (i,j) is ignored,

If $x_{ij} > 0$, then there is a backward arc (reverse arc) (j,i) has flow $x_{ji} = -x_{ij} \le 0$ and reduced cost $\bar{c}_{ji} = -c_{ij} + (u_j - u_i) \ge 0$,

If $x_{ij} = 0$, then the arc (j,i) is ignored.

2.4 Artificial arc

We introduced on network an additional arc (t,s) has cost $c_{ts} = 0$ and capacity $b_{ts} = \infty$.

3 Minimum-Cost Flow Algorithm with Zero Lower Bound on The Flow Vector

This algorithm solves the minimum-cost flow problem in polynomial time with zero lower bound and b upper bound on the flow vector x i.e. $0 \le x_k \le b_k$ for all arcs $k = 1,...,m$ on the network $G = (V, E)$, and also it is considered that $b_k < \infty$ for all $k = 1,...,m$.

Initialization

Set $b_k = \sum_{a=0}^{q} b_k^a 2^a$ for all arcs $k = 1,...,m$ /binary system where $b_k^a = 0$ or 1/

Set $r := q + 1$

Set $u_i := 0$ for all nodes $i = 1,...,n$ $/ s = 1, t = n /$

Set $x_k := 0$ and $b_k := 0$ for all arcs $k = 1,...,m$

Set $x_{ts} := 0$ /total flow/ and $z := 0$ /total cost/

Iterations

While(1) $(r \ge 1)$, then do

Set $r := r - 1$ and $z := 2z$

Set $x_k := 2x_k$ and $b_k := 2b_k$ for all arcs $k = 1,...,m$

Set $x_{ts} := 2x_{ts}$

Set $k := 1$

While(2) $(k \le m)$, then do /scan arcs $k = (i, j)$ /

If(1) $b_k^r = 1$, then do

Let $b_k := b_k + 1$

If(2) $\bar{c}_k = c_k - (u_j - u_i) < 0$, then do / $k = (i, j)$, $c_{ts} = 0$ /

Do procedure $D(j, i)$ from j to i on the residual network $G(x)$

If(3) $i \in p$, then do

Set $x_k := x_k + 1$

Set $x_l := x_l + 1$ for all forward arcs l on the shortest path μ of reduced costs from j to i in $G(x)$

Set $x_{gf} := x_{gf} - 1$ for all backward arcs $l = (f, g)$ on the shortest path μ

Set $z := z + \bar{c}_k$ / $\bar{c}_k = c_k - (u_j - u_i)$ with new node potentials u_i and u_j /

End If(3)

If(4) $\bar{c}_k = c_k - (u_j - u_i) < 0$ / with new node potentials u_i and u_j /

Set $u_e := u_e - \bar{c}_{ij}$ for all nodes $e = 1,...,n$ and $e \ne j$

End If(4)

End If(2)

Do procedure $D(s, t)$ from s to t on the new residual network $G(x)$

If(5) $t \in p$, then do

Set $x_{ts} := x_{ts} + 1$

Set $x_l := x_l + 1$ for all forward arcs l on the shortest path μ of reduced costs from s to t in $G(x)$

Set $x_{gf} := x_{gf} - 1$ for all backward arcs $l = (f, g)$ on the shortest path μ

Set $z := z + (u_t - u_s)$ /with new node potentials u_s and u_t /

End If(5)

End If(1)

Set $k := k + 1$

End While(2)

End While(1)

Set $u_i := u_i - u_s$ for all $i = 1,...,n$

End the algorithm

3.1 Procedure $D(j^*, i^*)$ (A variant of Dijkstra's algorithm)

This procedure gives the shortest path of reduced costs between j^* and i^* on the defined residual network $G(x)$ based on Dijkstra's algorithm

Initialization

Set $p := \phi$,

Set $I := \{1, 2, ..., n\}$,

Set $g = 0$

Set $d_j = \begin{cases} 0 & if \ \ j = j^* \\ \infty & if \ \ j \neq j^* \end{cases}$ for all $j = 1, ..., n$

Iterations

While $(I \neq \phi)$ do

Let $h := \inf \{d_i \setminus i \in I\}$

If ($h = \infty$ or $d_{i*} = g$) do

Set $d_i := g$ for all $i \in I$

Set $I := \phi$

Else do

Set $g := h$

Find $i \in I$ such that $d_i = g$

Set $I := I \setminus \{i\}$ and $p := p \cup \{i\}$

For all $j \in I$,such that (i, j) is an arc in the residual network, do

If $(d_j > g + \bar{c}_{ij})$ do / $\bar{c}_{ij} \geq 0$ reduced cost/

Set $d_j := g + \bar{c}_{ij}$

Set $pred(j) := i$

End If

End For all

End If

End While

Set $u_i := u_i + d_i$ for all $i = 1, ..., n$

End the procedure

Notes

1. After the application of the procedure $D(j^*, i^*)$ on the defined residual network, new potential function u will be re-determined.
2. After the application of the procedure $D(j^*, i^*)$ on the defined residual network, it is found that the set $p \neq \phi$ because $j^* \in p$ at least.
3. After the application of the procedure $D(j^*, i^*)$ on the defined residual network, if $i^* \in p$, then there is a path between j^* and i^* on the defined residual network else there is not any path between j^* and i^* on the defined residual network.

The following procedure determines the shortest path of reduced costs μ defined by nodes on the defined residual network from j^* to i^* in the case when there is a path between them i.e., $i^* \in p$.

3.2 Identification of the shortest path from j^* to i^* on the defined residual network

Initialization

Set $i := i^*$

Set $\mu := \{i\}$

Iterations

While $(i \neq j^*)$ do

Set $j := pred(i)$

Set $i := j$

Set $\mu := \{i\} \cup \mu$

End While

Comments

1. It is to notice that, in pending the execution of the algorithm the compatibility conditions between the current flow x and the current potential function u must be satisfied. In the contrary case it is necessarily to change the current potential function as follows: for arc $(i, j) \in E$
   - If $\bar{c}_{ij} < 0$ and $x_{ij} < b_{ij}$, then we will change the node potentials to be: $u_e := u_e - \bar{c}_{ij}$ for all $e = 1,...,n$ and $e \neq j$
   - If $\bar{c}_{ij} > 0$ and $x_{ij} > 0$, then we will change the node potentials to be: $u_e := u_e + \bar{c}_{ij}$ for all $e = 1,...,n$ and $e \neq i$
2. The procedure $D(j,i)$ is applied when $\bar{c}_{ij} < 0$, in this case, there are two possibilities, the first one is $i \in p$ (there is a path μ from j to i) in this case, it is found that $z_{new} < z_{old}$ when the arc $(t,s) \notin \mu$ and $z_{new} > z_{old}$ when the arc $(t,s) \in \mu$. The second one is $i \notin p$ in this case, z dos not change but the node potentials will be changed.
3. When we apply the procedure $D(s,t)$, there are also two possibilities, the first one is $t \in p$ (there is a path from s to t) in this case, it is found that $z_{new} > z_{old}$. The second one is $t \notin p$, in this case, it is found that, z dos not change but the node potentials will be changed.
4. The new reduced cost $\bar{c}_{ij}(new)$ is equal to the old reduced cost $\bar{c}_{ij}(old)$ minus the deference between d_j and d_i because: for arc $(i, j) \in E$

$$\begin{aligned}\bar{c}_{ij}(new) &= c_{ij} - (u_j - u_i) \\ &= c_{ij} - (u_j(old) + d_j - u_i(old) - d_i) \\ &= c_{ij} - (u_j(old) - u_i(old)) - (d_j - d_i)\end{aligned}$$

$$\bar{c}_{ij}(new) = \bar{c}_{ij}(old) - (d_j - d_i)$$

5. The reduced cost $\bar{c}_{ji}$ of the arc (j,i) is equal to the inverse reduced cost $\bar{c}_{ij}$ of the arc (i, j) and verse versa because: for arc $(i, j) \in E$

$$\begin{aligned}\bar{c}_{ji} &= c_{ji} - (u_i - u_j) \\ &= -c_{ij} - (u_i - u_j) \\ &= -(c_{ij} + (u_i - u_j)) \\ &= -(c_{ij} - (u_j - u_i)) \\ &= -\bar{c}_{ij}\end{aligned}$$

3.3 Complexity of the algorithm with zero lower bound on the flow vector

The time taken by the procedure $D(j^*, i^*)$, which is based on Dijkstra's algorithm is $O(n^2)$ arithmetic operations, where n is the number of nodes in the network $G = (V, E)$ .The maximum number of iterations of the algorithm is $m \times r$,where m is the number of arcs in the network $G = (V, E)$ and r is the smallest integer greater than or equal to $\log B$, where B is the largest arc capacity of the network. The procedure $D(j^*, i^*)$ is applied twice times in each iteration, then the time taken by the algorithm is at most $O(n^2 mr)$ arithmetic operations.

4 Minimum-Cost Flow Algorithm with Nonnegative Lower Bound on The Flow Vector

This algorithm solves the minimum-cost flow problem in polynomial time with $a \geq 0$ nonnegative lower bound and b upper bound on the flow vector x i.e. $0 \leq a_k \leq x_k \leq b_k$ for all arcs $k = 1, ..., m$ on the network $G = (V, E)$, and also it is considered that $b_k < \infty$ for all $k = 1, ..., m$.

It is supposed that there is a nonnegative lower bound $a \geq 0$ on the flow x in the network $G = (V, E)$ i.e. $0 \leq a_k \leq x_k \leq b_k$ this implies that

$0 \leq x_k - a_k \leq b_k - a_k$ for all arcs $k = 1, ..., m$.

Let $y_k = x_k - a_k$ and $b_k^* = b_k - a_k$ for all arcs $k = 1, ..., m$, which implies that $x_k = y_k + a_k$, $b_k = b_k^* + a_k$ and $0 \leq y_k \leq b_k^*$ for all arcs $k = 1, ..., m$.

Using the conservation constraint, it can be see that

$$\sum_{i=1}^{n} x_{ij} = \sum_{s=1}^{n} x_{js} \text{ for all nodes } j = 1, ..., n \quad (1)$$

From another hand, we have

$$\sum_{i=1}^{n} x_{ij} = \sum_{i=1}^{n} y_{ij} + \sum_{i=1}^{n} a_{ij} \text{ for all nodes } j = 1, ..., n \quad (2)$$

$$\sum_{s=1}^{n} x_{js} = \sum_{s=1}^{n} y_{js} + \sum_{s=1}^{n} a_{js} \text{ for all nodes } j = 1, ..., n \quad (3)$$

Using (1), (2) and (3), it can be found that

$$\sum_{i=1}^{n} y_{ij} = \sum_{s=1}^{n} y_{js} + w_j \text{ for all nodes } j = 1, ..., n$$

where $w_j = \sum_{s=1}^{n} a_{js} - \sum_{i=1}^{n} a_{ij}$ for all nodes $j = 1, ..., n$

An arc of capacity w_j and zero cost is added in the node j where, $j = 1, ..., n$, we define also a new

source (super source) called s^* and a new sink (super sink) called t^*.

In the case of $w_j > 0$, then an outgoing arc in the node j of the form (j, t^*) is added where, its capacity is $b^*_{jt^*} = w_j$ and its cost is $c_{jt^*} = 0$, in the case of $w_j < 0$, then an incoming arc in the node j of the form (s^*, j) is added where, its capacity is $b^*_{s^*j} = -w_j$ and its cost is $c_{s^*j} = 0$, in the case of $w_j = 0$, then there is not any arc added in the node j. These added arcs are at most n arcs called auxiliary arcs. A special arc of the form (t^*, s^*) is also added where, its capacity is $b^*_{t^*s^*} = \infty$ and its cost is $c_{t^*s^*} = 0$.

This new defined digraph will be denoted by $G^*(V^*, E^*)$, where it is consisting of the same set of nodes V added to it the super source s^* and the super sink t^* with $|V^*| = n^* = n + 2$, the same set of arcs E added to it all auxiliary arcs with $|E^*| = m^*$, where $m \le m^* \le m + n$ and the two special arcs (t, s) and (t^*, s^*).

Let w is the sum of capacities of auxiliary arcs which have strictly positive capacities i.e. $w = \sum_{\{j \in V : w_j > 0\}} w_j$, and let z_x denote the total cost of the flow x with nonnegative lower bound on x and let z_y denote the total cost of the flow y with zero lower bound on y, then it can see that:

$$z_x = \sum_{k=1}^{m} c_k x_k = \sum_{k=1}^{m} c_k (y_k + a_k) = \sum_{k=1}^{m} c_k y_k + \sum_{k=1}^{m} c_k a_k$$

$$z_x = z_y + \sum_{k=1}^{m} c_k a_k$$

Initialization

Set $b^*_k = \sum_{a=0}^{q} b^a_k 2^a$ for all arcs $k = 1, ..., m^*$ /binary system where $b^a_k = 0$ or 1/

Set $r^* := q + 1$

Set $u_i := 0$ for all nodes $i = 1, ..., n^*$ / $s = 1, t = n, s^* = n^* - 1, t^* = n^*, n^* = n + 2$ /

Set $y_k := 0$ and $b^*_k := 0$ for all arcs $k = 1, ..., m^*$

Set $y_{ts} := 0$ /total flow/ and $z_y := 0$ /total cost/

Set $y_{t^*s^*} := 0$

Set $w = \sum_{\{j \in V : w_j > 0\}} w_j$

Iterations

While(1) $(r^* \ge 1)$, then do

Set $r^* := r^* - 1$ and $z_y := 2z_y$

Set $y_k := 2y_k$ and $b^*_k := 2b^*_k$ for all arcs $k = 1, ..., m^*$

Set $y_{ts} := 2y_{ts}$ and $y_{t^*s^*} := 2y_{t^*s^*}$

Set $k := 1$

While(2) $(k \le m^*)$, then do /scan arcs $k = (i, j)$ /

If(1) $b^r_k = 1$, then do

Let $b_k^* := b_k^* + 1$

If(2) $\bar{c}_k = c_k - (u_j - u_i) < 0$, then do / $k = (i, j)$, $c_{ts} = 0, c_{t^*s^*} = 0$ /

Do procedure $D(j, i)$ from j to i on the residual network $G^*(y)$

If(3) $i \in p$, then do

Set $y_k := y_k + 1$

Set $y_l := y_l + 1$ for all forward arcs l on the shortest path μ of reduced costs from j to i in $G^*(y)$

Set $y_{gf} := y_{gf} - 1$ for all backward arcs $l = (f, g)$ on the shortest path μ

Set $z_y := z_y + \bar{c}_k$ / $\bar{c}_k = c_k - (u_j - u_i)$ with new node potentials u_i and u_j /

End If(3)

If(4) $\bar{c}_k = c_k - (u_j - u_i) < 0$ / with new node potentials u_i and u_j /

Set $u_e := u_e - \bar{c}_{ij}$ for all nodes $e = 1, ..., n^*$ and $e \neq j$

End If(4)

End If(2)

Do procedure $D(s^*, t^*)$ from s^* to t^* on the new residual network $G^*(y)$

If(5) $t^* \in p$, then do

Set $y_{t^*s^*} := y_{t^*s^*} + 1$

Set $y_l := y_l + 1$ for all forward arcs l on the shortest path μ of reduced costs from s^* to t^* in $G^*(y)$

Set $y_{gf} := y_{gf} - 1$ for all backward arcs $l = (f, g)$ on the shortest path μ

Set $z_y := z_y + \left(u_{t^*} - u_{s^*}\right)$ /with new node potentials u_{s^*} and u_{t^*} /

End If(5)

Do procedure $D(s, t)$ from s to t on the new residual network $G^*(y)$

If(6) $t \in p$, then do

Set $y_{ts} := y_{ts} + 1$

Set $y_l := y_l + 1$ for all forward arcs l on the shortest path μ of reduced costs from s to t in $G^*(y)$

Set $y_{gf} := y_{gf} - 1$ for all backward arcs $l = (f, g)$ on the shortest path μ

Set $z_y := z_y + \left(u_t - u_s\right)$ /with new node potentials u_s and u_t /

End If(6)

End If(1)

Set $k := k + 1$

End While(2)

End While(1)

Set $u_i := u_i - u_s$ for all $i = 1, ..., n$

If (7) $(y_{t^*s^*} < w)$, then, the network $G(V, E)$ has no feasible flow

Else Set $x_k = y_k + a_k$ for all arcs $k = 1, ..., m$

Set $b_k = b_k^* + a_k$ for all arcs $k = 1,...,m$

Set $z_x = z_y + \sum_{k=1}^{m} a_k c_k$

End If(7)

End the algorithm

Notes

1. The quantity $w = \sum_{\{j \in V : w_j > 0\}} w_j$ is the maximum flow in the network $G^*(V^*, E^*)$, then, we always have $(y_{t^*s^*} \le w)$, where $y_{t^*s^*}$ is the flow in $G^*(V^*, E^*)$.
2. In the case when all auxiliary arcs in $G^*(V^*, E^*)$ are saturated, i.e. $y_{t^*s^*} = w$, then the flow y is optimal in $G^*(V^*, E^*)$ and consequently the flow $x = y + a$ is optimal in $G(V, E)$.
3. In the case when there are some auxiliary arcs in $G^*(V^*, E^*)$ are not saturated, i.e. $y_{t^*s^*} < w$, then the flow y is optimal in $G^*(V^*, E^*)$ and consequently there is not any feasible flow x in $G(V, E)$.

Complexity of the algorithm with nonnegative lower bound on the flow vector: The time taken by the procedure $D(j^*, i^*)$, which is based on Dijkstra's algorithm is $O((n^*)^2)$ arithmetic operations, where n^* is the number of nodes in the network $G^* = (V^*, E^*)$.The maximum number of iterations of the algorithm is $m^* \times r^*$,where m^* is the number of arcs in the network $G^* = (V^*, E^*)$ and r^* is the smallest integer greater than or equal to $\log B$, where B is the largest arc capacity of the network $G^* = (V^*, E^*)$. The procedure $D(j^*, i^*)$ is applied triple times in each iteration, then the time taken by the algorithm is at most $O((n^*)^2 m^* r^*)$ arithmetic operations.

5 Minimum-Cost Flow Problem with Infinite Upper Bound on The Flow Vector

Two cases have been treated before in this paper, the first one is when there are a zero lower bound and a finite upper bound on the flow vector x i.e. $0 \le x_k \le b_k < \infty$ for all $k = 1,...,m$ and the second case is when there are a nonnegative lower bound and a finite upper bound on the flow x i.e. $0 \le a_k \le x_k \le b_k < \infty$ for all $k = 1,...,m$.

Now, two additional cases will be treated, the first one is when there are a zero lower bound and an infinite upper bound on the flow x i.e. $0 \le x_k \le b_k \le \infty$ for all $k = 1,...,m$. The second case is when there are a nonnegative lower bound and an infinite upper bound on the flow x i.e. $0 \le a_k \le x_k \le b_k \le \infty$ for all $k = 1,...,m$.

In the case of $0 \le x_k \le b_k \le \infty$ for all $k = 1,...,m$, we will do the following procedure:

Procedure

This procedure constructs an auxiliary network derived from the original network $G = (V, E)$ and also tests if the original minimum-cost flow problem has a feasible solution or not.

Initialization */auxiliary network/*

For each arc $(i, j) \in E$, then do

If $b_{ij} = \infty$ then, there is a forward arc (i, j) has a reduced cost $\bar{c}_{ij} = 1$

If $b_{ij} < \infty$ then the arc (i, j) is ignored

Iteration

Do procedure $D(s,t)$ from s to t on this auxiliary network.

If there is a path goes from s to t, i.e. $t \in p$, then the maximum flow is infinite and the minimum-cost flow problem does not have any finite feasible solution, else the maximum flow is upper bounded by the value of $\beta = \sum_{\{(i,j): i \in p \& j \in V \setminus p\}} b_{ij}$. In this case we will change the infinity ∞ in the original network $G = (V, E)$ by the value of β and solve it anew by the proposed algorithm.

Now, in the case of $0 \le a_k \le x_k \le b_k \le \infty$ for all $k = 1,...,m$, we will change it to the case of $0 \le y_k \le b_k^* \le \infty$ for all $k = 1,...,m$, where $y_k = x_k - a_k$ and $b_k^* = b_k - a_k$ for all $k = 1,...,m$, and we repeat the same procedure used before in the first case.

6 Formulating The Transportation and The Assignment Problems As A Minimal Cost Network Flow Problem

The transportation and the assignment problems can be formulated as a minimum-cost network flow problem on a bipartite digraph $G = (V = V_1 \cup V_2, E)$, where $V_1 = \{1,...,n_1\}$ is the set of sources, $V_2 = \{n_1 + 1,...,n\}$ is the set of sinks, with $|V| = n$, and $E = \{(i, j) : i \in V_1, j \in V_2\}$ is the set of arcs with $|E| = m$. The unit shipping cost from $i \in V_1$ to $j \in V_2$ is c_{ij}. Node $i \in V_1$ has a positive integral supply o_i, and node $j \in V_2$ has a positive integral demand of h_j. The transportation problem is to find a flow $x \in R_+^m$, that satisfies the supply-or-demand at minimum cost.

An artificial source (s) has been added and artificial arcs of the form (s, i) $(i = 1,...,n_1)$ have been also defined, they have capacities $b_{si} = o_i$ $(i = 1,...,n_1)$ and costs $c_{si} = 0$ $(i = 1,...,n_1)$.

An artificial sink (t) has been added and artificial arcs of the form (j, t) $(j = n_1 + 1,...,n)$ have been also defined, they have capacities $b_{jt} = h_j$ $(j = n_1 + 1,...,n)$ and costs $c_{jt} = 0$ $(j = n_1 + 1,...,n)$.

The arcs of the form (i, j) $(i = 1,...,n_1, j = n_1 + 1,...,n)$ have capacities $b_{ij} = \beta$, where $\beta = \min\left\{\sum_{i=1}^{n_1} o_i, \sum_{j=n_1+1}^{n} h_j\right\}$ and costs c_{ij}.

when $o_i = h_j = 1$ for all $i = 1,...,n_1$ and $j = n_1 + 1,...,n$, then the problem is called the assignment problem.

The both, the transportation problem and the assignment problem can be easily solved by the proposed algorithm as a minimum-cost network flow problem, in polynomial time.

7 Conclusion

Using the binary representation technique of capacities, a polynomial time algorithm for the minimum-cost flow problem has been developed in this paper. The algorithm runs in $O(n^2 m\ r)$ time, where n, m are the numbers of nodes and arcs of the network $G(V, E)$, respectively and r is the smallest integer greater than or equal to $\log B$, where B is the largest arc capacity of the network. The algorithm solves the minimum-cost flow problem as a sequence of $O(n^2)$ shortest path problems on residual networks.

A generalization of this algorithm has been also performed in order to solve a minimum-cost flow problem in a network with nonnegative lower bound on the flow vector.

8 Illustrative Example

The demonstration of the proposed algorithm for solving the minimum-cost flow problem will be done though the following numerical example presented analytically in Table 1 and graphically in Fig 1.

Table 1 Arcs notations, arcs capacities, arcs costs and lower bounds on the flow vector.

Arc $k=(i,j)$	Arc tail i	Arc head j	Arc capacity b_k	Arc cost c_k	Flow lower bound a_k^1	Flow lower bound a_k^2
1	1	2	50	3	15	45
2	1	3	30	6	7	25
3	1	4	15	8	5	15
4	2	3	50	2	5	20
5	2	5	25	2	3	25
6	3	4	15	2	5	15
7	3	5	45	1	4	40
8	3	6	10	3	2	10
9	3	8	15	8	4	14
10	4	6	10	1	5	9
11	4	9	20	3	4	19
12	5	7	90	9	1	33
13	5	8	10	8	5	8
14	6	8	60	5	1	40
15	7	8	10	1	2	10
16	7	11	10	2	4	5
17	8	10	10	1	3	3
18	8	11	80	4	50	60
19	9	8	20	2	4	16
20	9	10	10	3	4	5
21	10	11	10	3	3	7

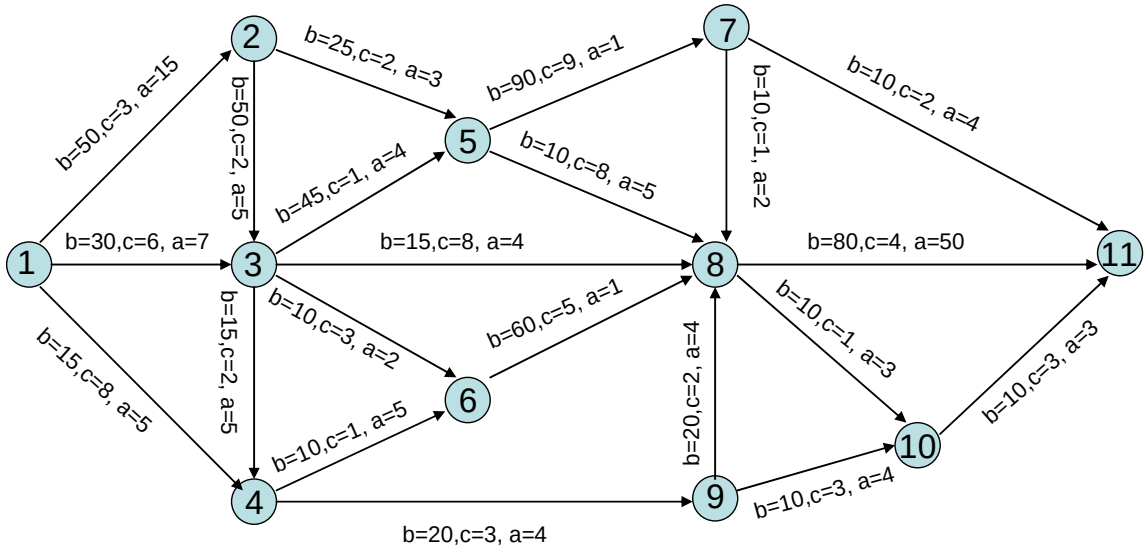

Fig. 1 Diagram of the example with a^1 lower bound on the flow vector.

The solution found by the proposed algorithm is given analytically in Table 2 and graphically in Fig 2.

Table 2 The solution of the proposed minimum-cost flow problem.

Arc $k=(i,j)$	Arc tail i	Arc head j	Arc capacity b_k	Arc cost c_k	Arc flow with a_k^1	Arc flow with a_k^2
1	1	2	50	3	50	The minimum-cost flow problem has no feasible flow
2	1	3	30	6	20	
3	1	4	15	8	15	
4	2	3	50	2	25	
5	2	5	25	2	25	
6	3	4	15	2	15	
7	3	5	45	1	5	
8	3	6	10	3	10	
9	3	8	15	8	15	
10	4	6	10	1	10	
11	4	9	20	3	20	
12	5	7	90	9	20	
13	5	8	10	8	10	
14	6	8	60	5	20	
15	7	8	10	1	10	

16	7	11	10	2	10	
17	8	10	10	1	3	
18	8	11	80	4	68	
19	9	8	20	2	16	
20	9	10	10	3	4	
21	10	11	10	3	7	
					Total flow is $x_{ts} = 85$ **Total cost is** $z = 1475$	

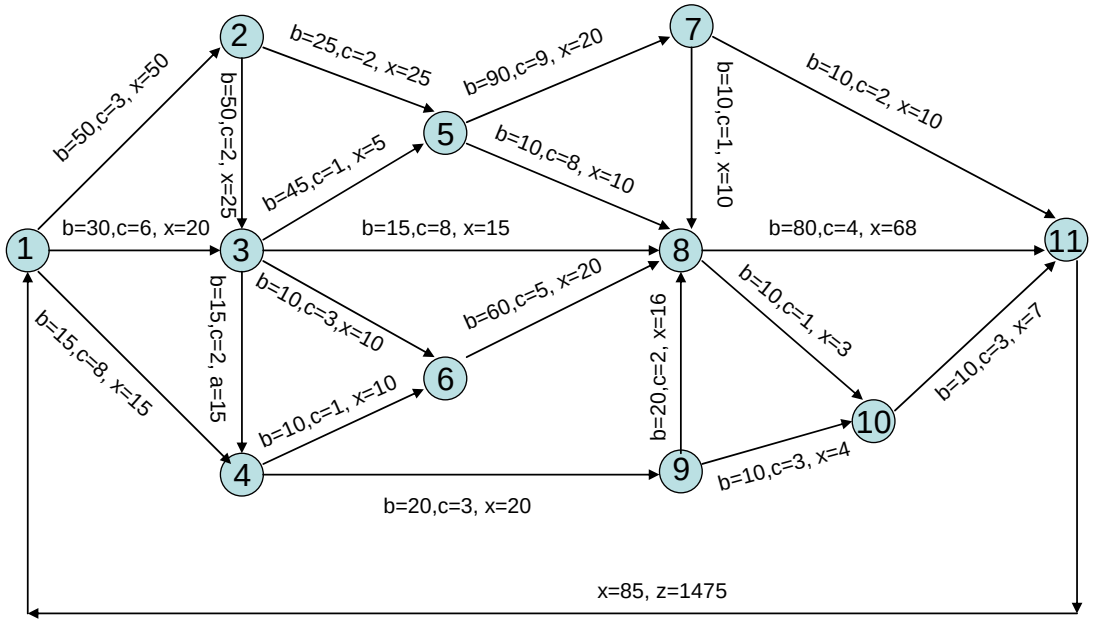

Fig. 2 Diagram of the solution with a^1 lower bound on the flow vector.

Acknowledgments

Author wishes to thank Prof. I. Othman, the DG of the AECS for his valuable support and encouragement throughout this work. The anonymous reviewers are cordially thanked for their critics, remarks and suggestions that considerably improved the final version of this paper.

References

Ahuja RK, Goldberg AV, Orlin J.B, Tarjan RE. 1992. Finding minimum-cost flows by double scaling. Mathematical Programming, 53: 243-266

Ahuja RK, Magnanti TL, Orlin JB. 1993. Network Flows: Theory, Algorithms and Applications. Prentice Hall, Englewood Cliffs, NJ, USA

Edmonds J, Karp R. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. Journal of ACM, 248-264

Ford LR, Fulkerson DR. 1962. Flows in Networks, Princeton University Press, Princeton, NJ, USA

Goldberg AV, Tarjan RE. 1990. Solving minimum-cost flow problem by successive approximation. Mathematics of Operations Research, 15: 430-466

Orlin JB. 1993. A faster strongly polynomial minimum-cost flow algorithm. Operations Research, 41(2): 338-350

Orlin JB. 1997. A polynomial time primal network simplex algorithm for minimum-cost flows. Mathematical Programming, 78: 109-129

Zhang WJ. 2016. A Matlab program for finding shortest paths in the network: Application in the tumor pathway. Network Pharmacology, 1(1): 42-53

Zhang WJ. 2017. Finding minimum cost flow in the network: A Matlab program and Application. Selforganizology, 4(2): 30-34