Article

A bit-arc capacity scaling algorithm for the maximum flow problem subjected to box constraints on the flow vector in digraph

Muhammad Tlas

Scientific Services Department, Atomic Energy Commission, P. O. Box 6091, Damascus, Syria

Email: pscientific31@aec.org.sy

Received 31 October 2024; *Accepted 10 December 2024*; *Published online 23 December 2024*; *Published 1 June 2025*

Abstract

A bit-arc capacity scaling algorithm to solve the maximal flow problem subjected to box constraints on the flow vector in directed network has been presented. The algorithm is mainly based on successive divisions of capacities by multiples of two. It solves the maximal flow problem as a sequence of $O(n^2)$ Dijkstra's shortest path between two nodes in the defined residual network with n nodes and m arcs. It is proven that, the algorithm's complexity was estimated to be no more than $O(n^2mr)$ arithmetic operations in the worst case to reach the maximum vector flow through the directed network. Where r denotes to the smallest integer greater than or equal to log B, and B denotes to the largest arc capacity of the network. A numerical example has been illustrated using the proposed algorithm.

Keywords maximum flow problem; scaling algorithm; polynomial time algorithm; augmenting path method; network flow; digraph.

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 maximal flow problem is the problem of determining the maximum amount of flow that can be sent from a source node to a sink node through a capacitated network without exceeding the capacity of any arc, in which conservation of flow holds at every node except the source and sink nodes.

The maximum flow problem is widely studied in both applications and theory (Zhang, 2017, 2018). Its applications can be found in diverse fields such as: Telecommunication Wireless Networks (Azar et al., 2011; Caillouet et al., 2010; Hu et al., 2010; Thulasiraman and Shen, 2010; Rushdi and Alsalami, 2020); Image Segmentation(Freedman and Zhang, 2005; Song et al., 2010; Zeng et al., 2008); Extraction of Web Communities (Asano et al., 2006; Horiike et al., 2009; Imafuji and Kitsuregawa, 2004); Transportation (Anderson et al., 2007; Brede and Boschetti, 2009; Çalıskan, 2011; Rebennack et al., 2010); Ecosystem (Rushdi and Alsalami, 2021); Coding Network and Wireless ad hoc Networks (Ahlswede et al., 2000).

The fundamental algorithmic techniques for solving the maximum flow problem are presented in Armstrong et al (1998), Noda et al (2000), pham et al (2006), Ahuja et al (1993), Ahuja and Orlin (1989), Goldfarb and Hao (1990, 1991), Orlin et al (1993), Gabow (1985), Cheriyan and Mehlhorn (1999) and Cherkassky and Goldberg (1997).

Generally, there are two main basic groups of algorithms to solve the maximal flow problem in directed networks: The first group of algorithms is the augmenting path methods which were introduced by Ford and Fulkerson (1962), and Edmonds and Karp (1972). The algorithm of Ford and Fulkerson is known as the augmenting path algorithm. An augmenting path is a directed path from the source to the sink in the residual network. The algorithm proceeds by identifying augmenting paths and sending flows on these paths until the network does not contain such a path. Complexity of the algorithm is $O(nmB)$, where n and m denotes the numbers of nodes and arcs in the network, respectively, and B is the largest arc capacity in the network.

The algorithm of Edmonds and Karp is known as the shortest augmenting path algorithm. This algorithm sends flow along the shortest path from the source to the sink in the residual network. The length of paths is the number of arcs that belongs to it. The complexity of this algorithm is $O(nm^2)$.

Recently Tlas (2022, 2023) has presented efficient algorithms to solve the maximum flow problem in polynomial time. These algorithms are mainly based, first on the binary representation of arcs capacities and second, on the use of both the Breadth-first search technique and Dijkstra's variant to find the shortest path going from the source node to the sink node in the residual networks.

The second group of algorithms is the preflow-push methods which were introduced by Golberg and Tarjan (1988) who takes the original idea of preflow from Karzanov (1974). The idea of the preflow-push algorithms is to select an active node and to push flow to its neighbors. To estimate the active nodes that are closer to the sink, the method keeps the distance label for each node. Thus, it sends flow only on admissible arcs. If the selected active node has no admissible arcs, its distance label is increased. This operation is called relabel. The algorithm terminates when the network does not contain active nodes. The complexity of the algorithm is $O(n^2m)$.

In this paper, a bit-arc capacity scaling algorithm is presented to find the maximum flow in a directed 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 $\log B$. The algorithm is basically based on successive divisions of capacities by multiples of two; it solves the maximum flow problem as a sequence of $O(n^2)$ shortest path problems on residual networks between two any nodes using Dijkstra's variant.

A generalization of this proposed algorithm has been also performed, in this paper, in order to solve the maximum flow problem in directed networks with nonnegative lower bound (box constraints) on the flow vector.

2 Preliminarily

In this section, we define the maximum flow problem and introduce the terminology and notation used throughout the paper.

2.1 Maximum flow problem statement

We consider a directed graph (digraph) $G=(V,E)$ consists of a set V of nodes and a set E of arcs. A directed network is a directed graph with numerical values attached to its arcs. Let $n=|V|$ and $m=|E|$. We associate with each arc $k=(i,j)\in E$ a nonnegative integral capacity b_k. Frequently, two special nodes are distinguished 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.

We introduced into network an additional arc (artificial arc) (t,s) has a capacity $b_{ts}=\infty$.

The total flow x from source node s to sink node t is x_{ts}.

The problem is to find a maximum flow x among the source node s and the sink node t with value x_{ts}.

A flow is a value x on arcs satisfies 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).

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. x and u are called compatible if the flow x and labeling function u together satisfy the following conditions: (for each 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 $0\le x_{ij}\le b_{ij}$,

Where $c_{ij}=1\ \forall(i,j)\in E$ is called the cost of the arc (i,j) and $\bar{c}_{ij}=c_{ij}-(u_j-u_i)$ 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 each arc $(i, j) \in E$)

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

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 reduced cost $\bar{c}_{ji} = -c_{ij} + (u_j - u_i) \geq 0$,

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

With taken into consideration that $c_{ij} = 1 \ \forall (i, j) \in E$.

2.4 Artificial arc

We introduce on network an additional arc (t, s) which has cost $c_{ts} = 0$, capacity $b_{ts} = \infty$ and reduced cost $\bar{c}_{ts} = c_{ts} - (u_s - u_t) = u_t - u_s \geq 0$.

3 Maximum Flow Algorithm With Zero Lower Bound On The Flow Vector

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

Initialization:

Set $r := \min \{q \in \mathbf{Z}^{+=} / 2^q > \max \{b_k, k = 1, ..., m\}\}$

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

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

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

Set $c_k := 1$ for all arcs $k = 1, ..., m$ and $c_{ts} := 0$

Iteration:

While (1) $(r \geq 1)$ then do

Set $r := r - 1$

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

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

Set $b_k := \left\lfloor \frac{B_k}{2^r} \right\rfloor$ for all arcs $k = 1,...,m$ /$\lfloor x \rfloor$ is the greatest integer less than or equal to x /

Set $k := 1$

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

If (1) $x_k < b_k$, then do

If (2) $\bar{c}_k = c_k - (u_j - u_i) < 0$ then do

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

If (3) $t \in P$, then do

Set $x_k := x_k + 1$

Set $x_v := x_v + 1$ for all forward arcs v on the path μ

of minimal reduced costs from j to i in $G(x)$

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

End If (3)

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

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 path μ of

minimal reduced costs from s to t in $G(x)$

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

path μ

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 $D(j^*, i^*)$ (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 $j = 1,...,n$

Iteration:

While $(I \neq \phi)$ do

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

If $h = \infty$ or $d_{i^*} := g$ then 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 $\quad / \bar{c}_{ij} \geq 0$ 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

a. After the application of the procedure $D(j^*, i^*)$ on the defined residual network, new potential function u will be re-determined.

b. 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.

c. 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 the 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 $\mu := \{i\}$

Iteration:

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

Set $j := pred(i)$

Set $i := j$

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

End While

Comments

a. It is noticed that, when the algorithm is executed, 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 $e \neq i$

b. 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 $(i, j) \in E$)

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

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

c. 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. For any iteration, the procedure $D(j^*, i^*)$ is applied two times and then the time taken by the algorithm will be no more than $O(n^2 mr)$ arithmetic operations to reach the maximum vector flow through the directed network.

4 Maximum Flow Algorithm With Non-Negative Lower Bound On The Flow Vector

This algorithm solves the maximum flow problem in polynomial time with $a \geq 0$ nonnegative lower bound and b upper bound (box constraints) 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 the whole 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 seen 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

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

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

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

Initialization:

Set $r^* := \min\left\{q \in \mathbf{Z}^{+=} / 2^q > \max\left\{b^*_k, k = 1,...,m^*\right\}\right\}$

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 $B^*_k := b^*_k$ for all arcs $k = 1,...,m^*$

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

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

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

Set $c_k := 1$ for all arcs $k = 1,...,m$

Iteration:

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

Set $r^* := r^* - 1$

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

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

Set $b^*_k := \left\lfloor \frac{B^*_k}{2^{r^*}} \right\rfloor$ for all arcs $k = 1,...,m^*$

Set $k := 1$

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

If (1) $y_k < b_k^*$, then do

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

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 $e \ne i$

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 of reduced costs

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

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 $k = 1, ..., m$

End If(7)

The total flow from source *s* to sink t on the network $G(V, E)$ is $x_{ts} = y_{ts}$

End the algorithm

Notes

a. The quantity $w = \sum_{\{j \in V : w_j > 0\}} w_j$ is the maximum flow in the network $G^*(V^*, E^*)$, and then we always have $(y_{t^*s^*} \le w)$ where, $y_{t^*s^*}$ is the flow in $G^*(V^*, E^*)$.

b. 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 $G(V, E)$.

c. 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 $G^* = (V^*, E^*)$. For any iteration, the procedure $D(j^*, i^*)$ is applied three times and then the time taken by the algorithm will be at most $O((n^*)^2 m^* r^*)$ arithmetic operations to reach the maximum vector flow through the directed network.

5 Maximum Flow Algorithm With Infinite Upper Bound On The Flow Vector

Two cases have been treated earlier in this paper, the first one is when a zero lower bound and a finite upper bound are 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 a nonnegative lower bound and a finite upper bound are 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 a zero lower bound and an infinite upper bound are 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 a nonnegative lower bound and an infinite upper bound are 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 forms an auxiliary network derived from the original network $G = (V, E)$ and also tests if the original maximum 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 maximum 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 the infinity ∞ in the original network $G = (V, E)$ will be changed by the value of β and resolve it using 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 Conclusion

In this paper, using the successive divisions of capacities by multiples of two, a bit-scaling algorithm to solve the maximal flow problem has been presented. The algorithm runs in no more than $O(n^2 m\ r)$ arithmetic operations to reach the maximum vector flow through the directed network. Where n and m denote 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 maximal flow problem as a sequence of $O(n^2)$ shortest path sub-problems on residual networks.

A generalization of this algorithm has been also performed in order to solve the maximal flow problem in directed networks subjected to box constraints on the flow vector.

7 Illustrative Example

The demonstration of the proposed algorithm for solving the maximum flow problem will be done through the following numerical example.

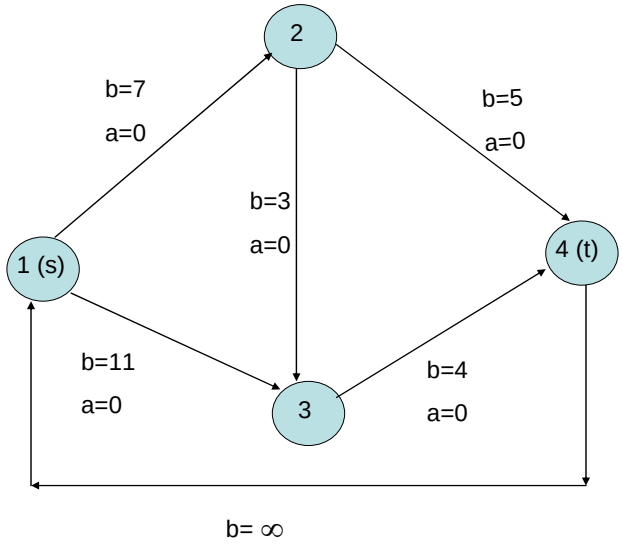

Fig. 1 Diagram of example with zero lower bound on the flow vector (network $G(V, E)$).

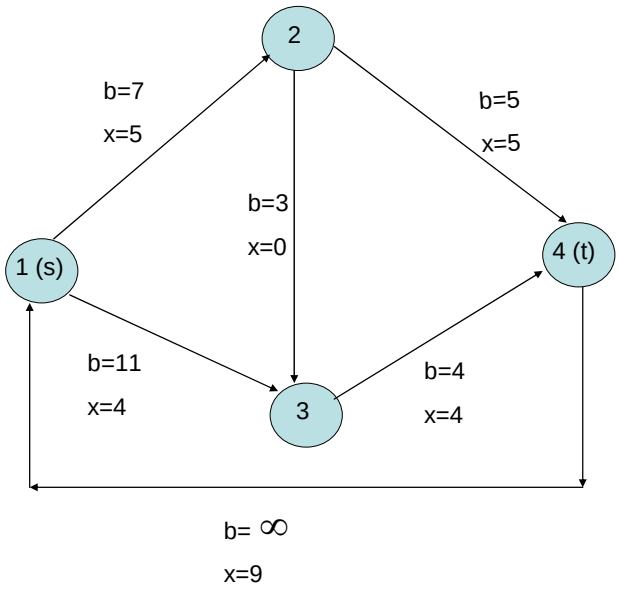

Fig. 2 Diagram of solution with zero lower bound on the flow vector (network $G(V, E)$).

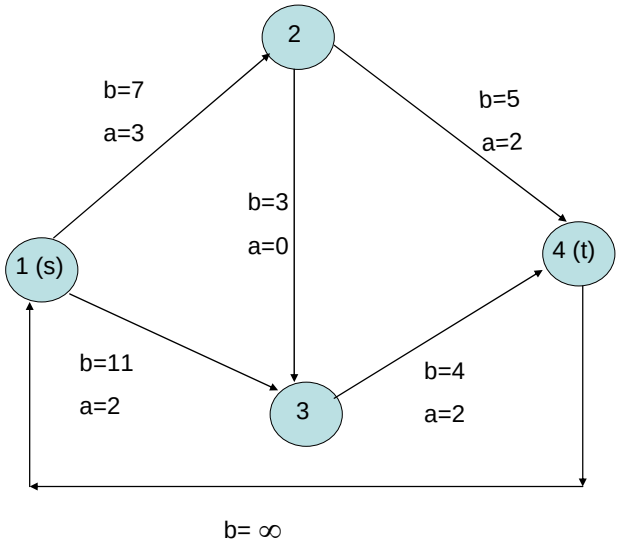

Fig. 3 Diagram of example with nonnegative lower bound on the flow vector (network $G(V, E)$).

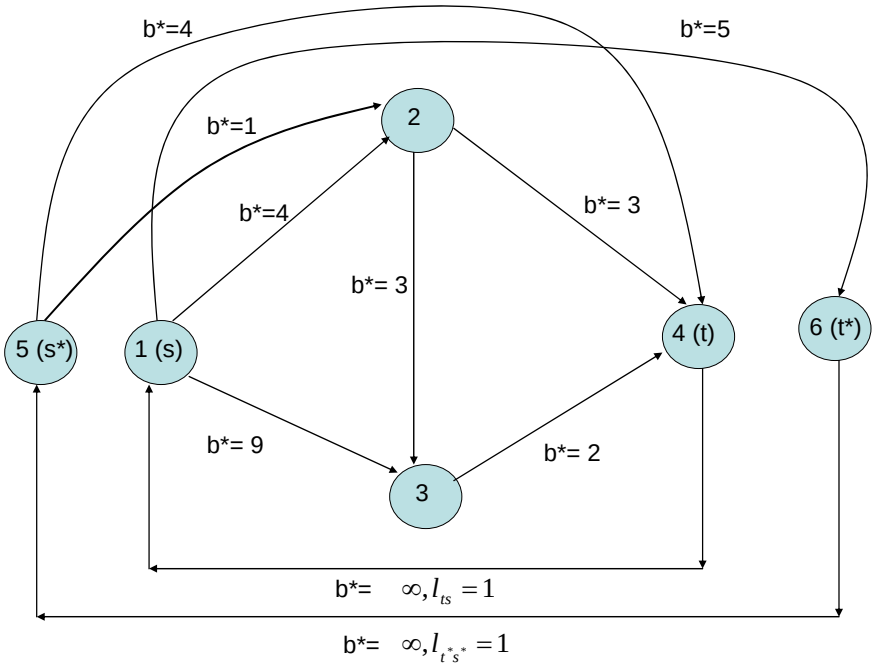

Fig. 4 Diagram of example with added auxiliary arcs (network $G^*(V^*, E^*)$).

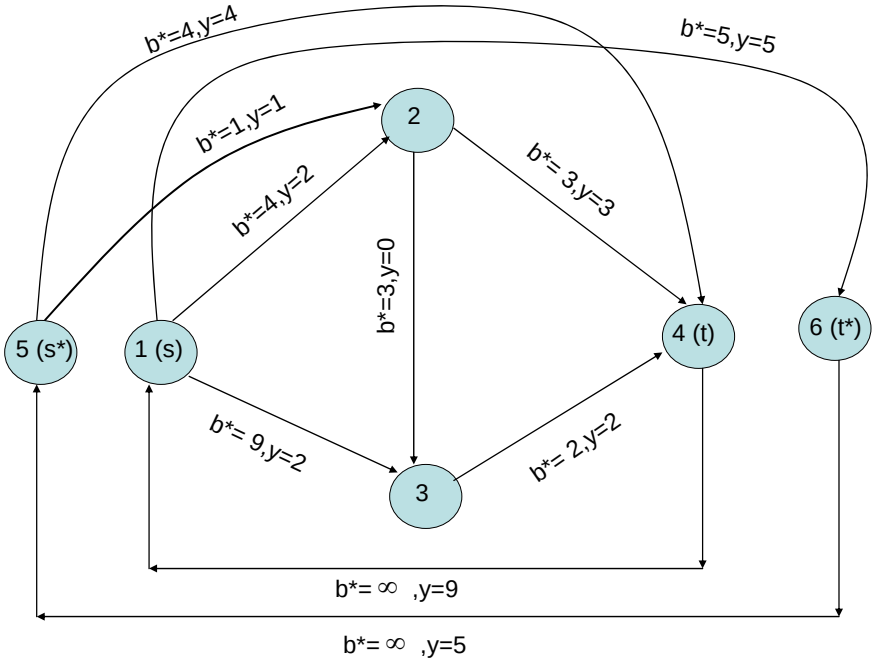

Fig. 5 Diagram of solution with added auxiliary arcs (network $G^*(V^*, E^*)$).

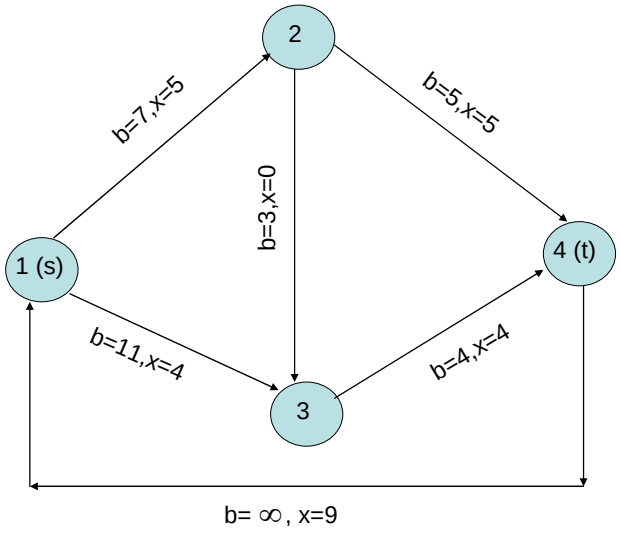

Fig. 6 Diagram of solution with nonnegative lower bound on the flow vector (network $G(V, E)$).

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

Ahlswede R, Cai N, Li SYR, Yeung RW. 2000. Network information flow. IEEE Trans on Information Theory, 46: 1204-1216

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

Ahuja RK, Orlin JB. 1989. A fast and simple algorithm for the maximum flow problem. Operations Research, 37: 748-759

Anderson LB, Atwell RJ, Barnett DS, Bovey RL. 2007. Application of the maximum flow problem to sensor placement on urban road networks for homeland security. Homeland Security Affairs, 3(3): 1-15

Armstrong RD, Chen W, Goldfarb D, Jin Z. 1998.Strongly polynomial dual simplex methods for the maximum flow problem. Mathematical Programming, 80: 17-33

Asano Y, Nishizeki T, Toyoda M, Kitsuregawa M. 2006. Mining communities on the Web using a max-flow and a site oriented framework. IEICE – Transactions on Information and Systems E, 89-D(10): 2606-2615

Azar Y, Mądry A, Moscibroda T, Panigrahi D, Srinivasan A. 2011. Maximum bipartite flow in networks with adaptive channel width. Theoretical Computer Science, 412(24): 2577-2587

Brede M, Boschetti F. 2009. Analysing weighted networks. An approach via maximum flows. In: Complex Sciences (Zhou J, ed). 1093-1104, Springer-Verlag, Berlin, Germany

Caillouet C, Perennes S, Rivano H. 2010. Cross line and column generation for the cut covering problem in wireless networks. Electronic Notes in Discrete Mathematics, 36: 255-262

Çalıskan C. 2011. A specialized network simplex algorithm for the constrained maximum flow problem.

European Journal of Operational Research, 210(2): 137-147

Cherkassky BV, Goldberg AV. 1997. On implementing push-relabel method for the maximum flow problem. Algorithmica, 19: 390-410

Cheriyan J, Mehlhorn K. 1999. An analysis of the highest-level selection rule in the preflow-push max-flow algorithm. Information Processing Letters, 69: 239-242

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

Freedman D, Zhang T. 2005. Interactive graph cut based segmentation with shape priors. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1: 755–762

Gabow HN. 1985. Scaling algorithms for network problems. Journal of Computer and System Sciences, 31(2): 148-168

Goldberg AV, Tarjan RE. 1988. A new approach to the maximum flow problem. Journal of Assoc Comput Mach, 35(4): 921-940

Goldfarb D, Hao J. 1990. A primal simplex algorithm that solves the maximum flow problem in at most nm pivots and $O(n^2m)$ time. Mathematical Programming, 47: 353-365

Goldfarb D, Hao J. 1991. On strongly polynomial variants of the network simplex algorithm for the maximum flow problem. Operations Research Letters, 10: 383-387

Gondran M, Minoux M. 1985. Graphes et algorithmes. Editions Eyrolles, France

Horiike T, Takahashi Y, Kuboyama T, Saka-moto H. 2009. Extracting research communities by improved maximum flow algorithm. In: KES 2009 Proceedings of the 13th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems: Part II: Springer-Verlag, Berlin, Germany, 5712: 472-479

Imafuji N, Kitsuregawa, M. 2004. Finding Web communities by maximum flow algorithm using well-assigned edge capacities. The Institute of Electronics, Information and Communication Engineers E, 87-D(2): 407-415

Hu CC, Kuo YL, Chiu CY, Huang YM. 2010. Maximum bandwidth routing and maximum flow routing in wireless mesh networks. Telecommunication Systems, 44(1-2): 125-134

Karzanov AV. 1974. Determining the maximum flow in a network by the method of preflows. Soviet Mathematics Doklady, 15: 434-437

Noda AS, Gonzalez-Sierra MA, Gonzalez-Martin C. 2000. An algorithmic study of the maximum flow problem: A comparative statistical analysis. Sociedad de Estadistica e Investigacion Operativa, 8(1): 135-162

Orlin JB, Plotkin SA, Tardos E. 1993. Polynomial dual network simplex algorithms. Mathematical Programming, 60: 255-276

Pham TL, Bui M, Lavallee I, Do SH. 2006. A distributed preflow-push for the maximum flow problem. IICS 2005, LNCS 3908: 195-206, Springer-Verlag, Berlin, Heidelberg, Germany

Rebennack S, Arulselvan A, Elefteriadou L, Pardalos PM. 2010. Complexity analysis for maximum flow problems with arc reversals. Journal of Combinatorial Optimization, 19: 200-216

Rushdi AMA, Alsalami OM. 2020. Reliability evaluation of multi-state flow networks via map methods. Journal of Engineering Research and Reports, 13(3):45-59

Rushdi AMA, Alsalami OM. 2021. Reliability analysis of flow networks with an ecological perspective. Network Biology, 11(1): 1-28

Song Q, Liu Y, Liu Y, Saha PK, Sonka M, Wu X. 2010. Graph search with appearance and shape information for 3-D prostate and bladder segmentation. In: MIC- CAI, Part III, LNCS. Medical Image Computing and Computer-Assisted Intervention. Springer-Verlag, Berlin, Germany, 6363: 172-180

Thulasiraman P, Shen X. 2010. Interference aware resource allocation for hybrid hierarchical wireless networks. Computer Networks, 54(13): 2271-2280

Tlas M. 2022. Using the binary representation of arc capacity in a polynomial time algorithm for the constrained maximum flow problem in directed networks. Network Biology, 12(3): 81-96

Tlas M. 2023. A polynomial time algorithm for the maximal constrained network flow problem based on the bit-arc capacity scaling technique. Network Biology, 13(4): 122-136

Zeng Y, Dimitris Samaras D, Chen W, Peng Q. 2008. Topology cuts: A novel mincut/ max- flow algorithm for topology preserving segmentation in N–D images. Computer Vision and Image Understanding, 112(1): 81-90

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

Zhang WJ. 2018. Fundamentals of Network Biology. World Scientific Europe, London, UK