Article

Using the binary representation of arc capacity in a polynomial time algorithm for the constrained maximum flow problem in directed networks

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

Received 25 April 2022; *Accepted 30 May 2022*; *Published online 6 June 2022*; *Published 1 September 2022*

Abstract
In this paper, the binary representation of arc capacity has been used in developing an efficient polynomial time algorithm for the constrained maximum flow problem in directed networks. The algorithm is basically based on solving the maximum flow problem as a sequence of $O(n^2)$ shortest path problems on residual directed networks with n nodes generated during iterations. The complexity of the algorithm is estimated to be no more than $O(n^2mr)$ arithmetic operations, where m denotes the number of arcs in the network, and r is the smallest integer greater than or equal to log B (B denotes the largest arc capacity in the directed network). Generalization of the algorithm has been also performed in order to solve the maximum flow problem in a directed network subject to non-negative lower bound on the flow vector. A formulation of the simple transportation problem, as a maximal network flow problem has been also performed. Numerical example has been inserted to illustrate the use of the proposed algorithm.

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

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 maximum flow problem is one of the most fundamental network problems and has been investigated extensively in the literature. The maximum 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. Its applications can be found in diverse fields such as engineering, traffic management, scheduling, etc. Recently, it has been applied in some new domains, such as 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), Cherkassky and Goldberg (1997), Alsalami and Rushdi (2021), Rushdi and Alsalami (2020,2021), Curry and Smith (2021), Suvak et al. (2020), and Zhang (2018a, b).

In general there are two principal categories of algorithms for solving the maximum flow problem:

The first category of algorithms is the augmenting path methods which are 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. The complexity of the algorithm is $O(nmB)$, where n, m are the numbers of nodes and arcs respectively in the network and B is the largest arc capacity in the directed 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 the paths is the number of arcs that belongs to it. The complexity of this algorithm is $O(nm^2)$.

The second category of algorithms is the preflow-push methods which are introduced by Golberg and Tarjan (1988) who takes the original idea of preflow from Karzanov (1974). The idea of 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, an efficient polynomial time algorithm is presented for determining the maximum flow in a directed network with an upper bound $O(n^2mr)$ on the number of arithmetic operations, where r is the smallest integer greater than or equal to log B, where B denotes the largest arc capacity in the directed network. The algorithm is basically based on the binary representation of arc capacity; it solves the maximum flow problem as a sequence of $O(n^2)$ shortest path problems on residual directed networks generated during iterations. A generalization of this algorithm has been also performed, in this paper, in order to solve a constrained maximum flow problem in a directed network with nonnegative lower bound on the flow vector. A formulation of the simple transportation problem, as a maximal network flow problem has been also performed. A numerical example has been inserted to illustrate the use of the proposed method.

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) $G=(V,E)$ consisting 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, 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.

We introduced on network an additional arc (artificial arc) (t,s) has 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 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).

2.2 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 $x_{ij}\ge 0$, residual flow $r_{ij}=b_{ij}-x_{ij}$ and length or cost $l_{ij}=1$,

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 $x_{ji}=-x_{ij}\le 0$, residual flow $r_{ji}=x_{ij}$ and length or cost $l_{ji}=1$,

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

2.3 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. $0\le x_k\le b_k$ 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 $B_k := b_k$ for all arcs $k = 1,...,m$
        Set $x_k := 0$ and $b_k := 0$ for all arcs $k = 1,...,m$
        Set $x_{ts} := 0$ /total flow/and $b_{ts} := \infty$ /s=1, t=n/
        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 $r := q + 1$
```

Iteration

```
While (1) $(r \geq 1)$, then do
        Set $r := r - 1$
        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) $(k \leq m)$, then do /scan arcs $k = (i, j)$ /
        If (1) $b_k^r = 1$, then do
            Let $b_k := b_k + 1$
            Do procedure $Dijkstra(s,t)$ from $s$ to $t$ on the new residual network
            $G(x)$
          If (2) $t \in p$, then do
                Set $x_{ts} := x_{ts} + 1$
                Set $x_v := x_v + 1$ for all forward arcs $v$ on the shortest
path $\mu$
                        of lengths from $s$ to $t$ in $G(x)$
                Set $x_{ji} := x_{ji} - 1$ for all backward arcs $v = (i, j)$ on
the
                        shortest path $\mu$
          End If (2)
```

```
                    End If (1)
            Set $k : k + 1$
        End While (2)
End While (1)
End the algorithm
```

2.4 Procedure $Dijkstra(s,t)$

This procedure gives the shortest path of lengths between s and t on the defined residual network $G(x)$

```
Initialization
    Set $p := \phi$,
    Set $I := \{1,2,...,n\}$,
    Set $g = 0$,
    Set $d_j = \begin{cases} 0 & if\ j = s \\ \infty & if\ j \neq s \end{cases}$        for all $j = 1,...,n$
Iteration
While $(I \neq \phi)$ do
    Let $h := \inf\{d_i \setminus i \in I\}$
    If $h = \infty$ do
        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 $(d_j > g + l_{ij})$ do
                Set $d_j := g + l_{ij}$
                Set $pred(j) := i$
            End If
```

```
                End For all
            End If
End While
End the procedure
```

Notes

a. After the application of the procedure $Dijkstra(s,t)$ on the defined residual network, it is found that the set $p \neq \phi$ because $s \in p$ at least.

b. After the application of the procedure $Dijkstra(s,t)$ on the defined residual network, if $t \in p$, then there is a path between s and t on the defined residual network else there is not any path between s and t on the defined residual network.

c. Being the lengths of all arcs in the residual network $G(x)$ defined to be equal to positive ones, and then the general shortest path algorithm of Dijkstra with at most $O(n^2)$ arithmetic operations can be used to find an augmenting path in the residual network (Gondran and Minoux, 1985).

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

2.5 Determination of the shortest path from s to t on the defined residual network

```
Initialization
        Set $i := t$
        Set $\mu := \{i\}$
Iteration
While $(i \neq s)$ do
        Set $j := pred(i)$
        Set $i := j$
        Set $\mu := \{i\} \cup \mu$
End While
```

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

The time taken by the procedure $Dijkstra(s,t)$, 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 $Dijkstra(s,t)$ is applied once time, in each iteration, then the time taken by the algorithm is at most $O(n^2mr)$ arithmetic operations.

3 Maximum Flow Algorithm With Nonnegative 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 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

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

From another hand, we have

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

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

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

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

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 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 $b^*_{jt^*} = w_j$, 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 $b^*_{s^*j} = -w_j$, 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 $b^*_{t^*s^*} = \infty$.

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

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

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

$$b_{ts} = b^*_{ts} = \infty, b_{t^*s^*} = b^*_{t^*s^*} = \infty \ /s = 1, t = n, s^* = n^* - 1, t^* = n^*, n^* = n + 2/$$

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

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

Iteration

```
While (1) $(r^* \geq 1)$, then do
        Set $r^* := r^* - 1$
        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) $(k \leq m^*)$, then do /scan arcs $k = (i,j)$ /
        If (1) $b_k^r = 1$, then do
            Let $b_k^* := b_k^* + 1$
            Do procedure $Dijkstra(s^*,t^*)$ from $s^*$ to $t^*$ on the new residual network $G^*(y)$
            If (2) $t^* \in p$, 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 $\mu$ of lengths from $s^*$ to $t^*$ in $G^*(y)$
                Set $y_{ji} := y_{ji} - 1$ for all backward arcs $l = (i,j)$ on the shortest path $\mu$
            End If (2)
            Do procedure $Dijkstra(s,t)$ from $s$ to $t$ on the new residual network $G^*(y)$
            If (3) $t \in p$, then do
                Set $y_{ts} := y_{ts} + 1$
                Set $y_l := y_l + 1$ for all forward arcs $l$ on the shortest
```

```
path $\mu$ of lengths from $s$ to $t$ in $G^*(y)$
            Set $y_{ji} := y_{ji} - 1$ for all backward arcs $l = (i, j)$ on the shortest path $\mu$
            End If (3)
          End If (1)
      Set $k : k + 1$
    End While (2)
End While (1)
If (4) $(y_{t^*s^*} < w)$, 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 (4)
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. It is always taken in consideration that the artificial added arcs (t, s) and (t^*, s^*) have infinite capacities ($b_{ts} = b^*_{ts} = \infty, b_{t^*s^*} = b^*_{t^*s^*} = \infty$) and lengths of one ($l_{ts} = 1, l_{t^*s^*} = 1$), then they are always considered as permanent arcs in the residual network $G^*(y)$.

b. 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 $G^*(V^*, E^*)$.

c. 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 $G(V, E)$.

d. 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 $G(V, E)$.

The time taken by the procedure $Dijkstra(s,t)$, 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 $Dijkstra(s,t)$ is applied twice times in each iteration, then the time taken by the algorithm is at most $O((n^*)^2 m^* r^*)$ arithmetic operations.

4 Maximum 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

This procedure constructs 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 length $l_{ij} = 1$

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

Iteration

Do procedure $Dijkstra(s,t)$ 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 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.

5 Formulating The Simple Transportation Problem As A Maximum Network Flow

The simple transportation problem can be formulated as a maximal 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 $|E| = m$. 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 simple transportation problem is to find a maximum flow $x \in R_+^m$ that satisfies the supply-or-demand at the same time with respecting the maximum capacities of arcs in the directed network.

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 $b_{si} = o_i \;(i = 1,...,n_1)$.

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 $b_{jt} = h_j \;(j = n_1 + 1,...,n)$.

The arcs of the form $(i, j)\;(i = 1,...,n_1, j = n_1 + 1,...,n)$ have non-negative integral capacities $b_{ij} \ge 0$. The simple transportation problem can be easily solved by the proposed algorithm as a maximum network flow problem in polynomial time.

6 Conclusions

Using the binary representation technique of arc capacity, a polynomial time algorithm for the maximum flow problem has been developed in this paper. The algorithm runs in no more than $O(n^2 m\, r)$ arithmetic operations, where n , m are the numbers of nodes and arcs of the directed 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 maximum flow problem as a sequence of $O(n^2)$ shortest path problems on residual networks generated during iterations.

Generalization of this algorithm has been also performed in order to solve a constrained maximum flow problem in a network with nonnegative lower bound on the flow vector.

Formulation of the simple transportation problem, as a maximal network flow problem has been also performed and presented in this work.

7 Illustrative Example

The demonstration of the proposed algorithm for solving the maximum flow problem will be done though the following numerical example (Figs 1-6).

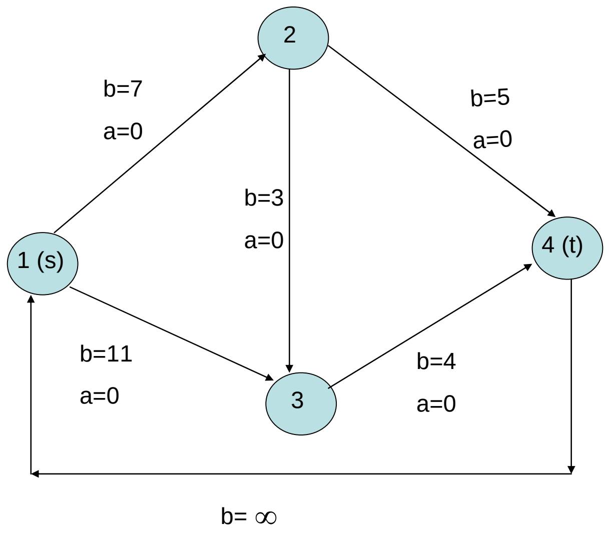

Fig. 1 Diagram of example with zero lower bound on the flow vector (network *G*(*V*, *E*)).

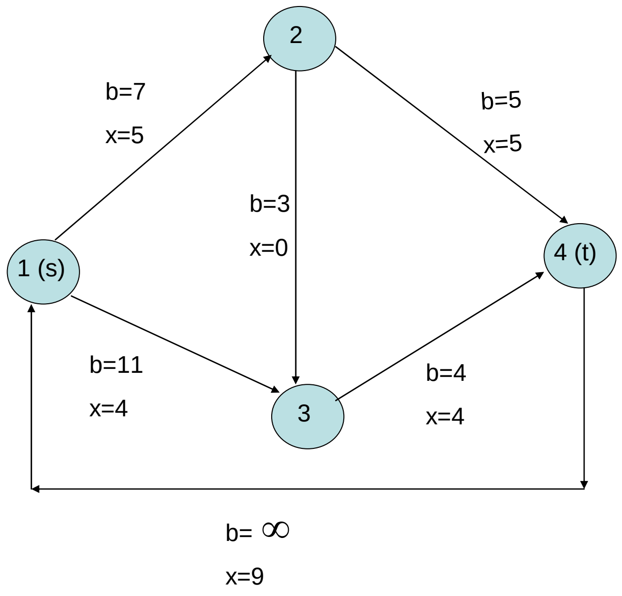

Fig. 2 Diagram of solution with zero lower bound on the flow vector (network *G*(*V*, *E*)).

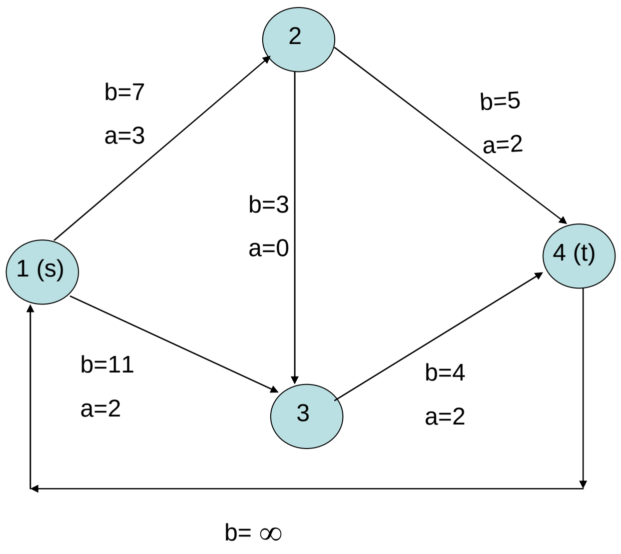

Fig. 3 Diagram of example with nonnegative lower bound on the flow vector (network *G*(*V*, *E*)).

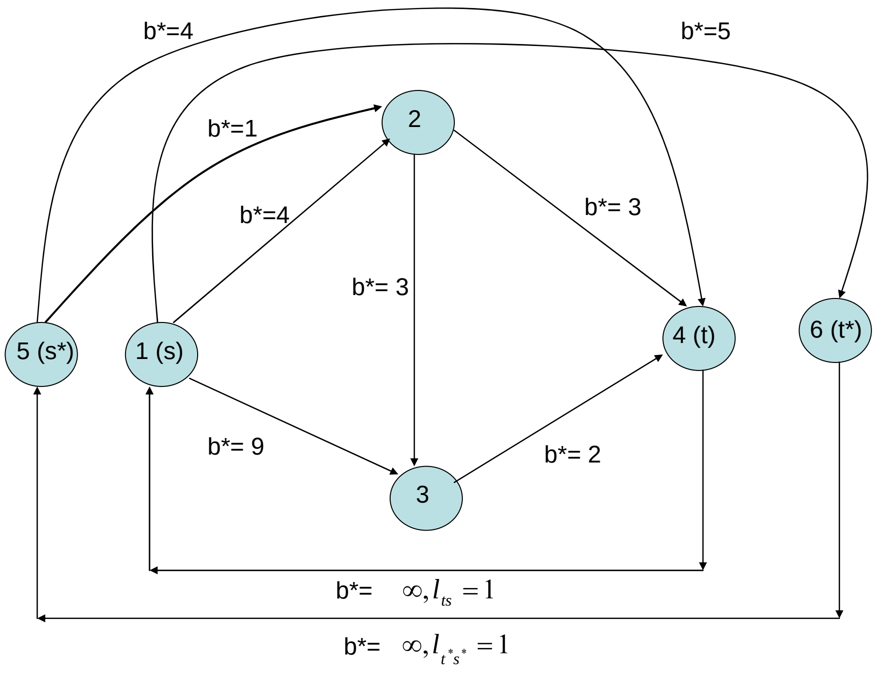

Fig. 4 Diagram of example with added auxiliary arcs (network *G**(*V**, *E**)).

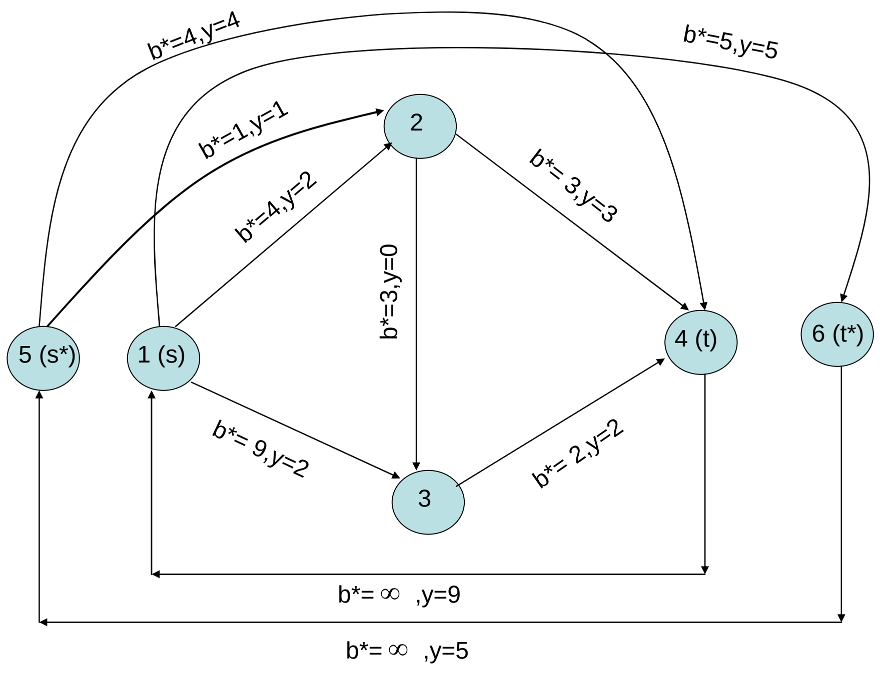

Fig. 5 Diagram of solution with added auxiliary arcs (network $G^*(V^*, E^*)$).

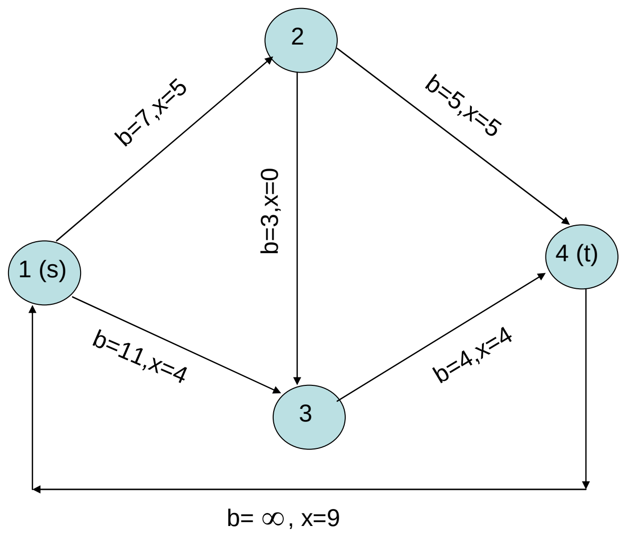

Fig. 6 Diagram of solution with nonnegative lower bound on the flow vector (network $G(V, E)$).

Acknowledgments

The 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

Alsalami OM, Rushdi AAM. 2021. A Review of Flow-Capacitated Networks: Algorithms, Techniques and Applications. Asian Journal of Research in Computer Science, 7(3): 1-33

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

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

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

Curry RM, smith JC. 2021. An augmenting-flow algorithm for a class of node-capacitated maximum flow problems. Networks, https://doi.org/10.1002/net.22082

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 Univ. Press, Princeton, NJ, USA

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 the Association for Computing Machinery, 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

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

Şuvak Z, Altınel IK, Aras N. 2020. Exact solution algorithms for the maximum flow problem with additional conflict constraints. European Journal of Operational Research, 287(2): 410-437

Zhang WJ. 2018a. Finding maximum flow in the network: A Matlab program and application. Computational Ecology and Software, 8(2): 57-61

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