Article

A polynomial time algorithm for the maximal constrained network flow problem based on the bit-arc capacity scaling technique

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

Received 3 April 2023; *Accepted 10 May 2023*; *Published online 19 May 2023*; *Published 1 December 2023*

Abstract
An efficient polynomial time algorithm for solving maximum flow problems in directed networks has been proposed in this paper. The algorithm is basically based on successive divisions of capacities by multiples of two; it solves the maximum flow problem as a sequence of $O(m)$ shortest path problems on residual networks with n nodes and m arcs. It runs in $O(m^2 r)$ time, where r is the smallest integer greater than or equal to $\log B$, and B is the largest arc capacity of the network. A numerical example has been illustrated using this proposed algorithm.

Keywords maximum flow problem; bit-capacity 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 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: 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), Zhang (2018a, b), and Cherkassky and Goldberg (1997).

In general there are two basic categories of algorithms for solving the maximum flow problem: The first category 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. The complexity of the algorithm is $O(nmB)$, where n and m are 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 $O(nm^2)$.

The second category 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 recent work of Tlas (2022), the binary representation of arc capacity has been used in developing an effective and robust polynomial time algorithm for the constrained maximum flow problem in directed networks.

In this paper, an efficient polynomial algorithm is presented for determining the maximum flow in a network with an upper bound of $O(m^2r)$ on the number of arithmetic operations, where m is the number of arcs 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 $O(m)$ shortest path problems on residual networks.

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

2 Preliminarily

In this section I 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 non-negative 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 the node j is called the head of the 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.

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

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

Set $x_{ts}:=0$ /total flow/ and $b_{ts}=\infty$ / *s*=1, *t*=*n*/

Iteration:

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

Set $r := r - 1$

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 $x_k := 2x_k$ 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) $x_k < b_k$, then do

Do procedure $BFS(s,t)$ 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 μ 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 μ

End If(2)

End If(1)

Set $k := k + 1$

End While(2)

End While(1)

End the algorithm

2.4 Procedure *BFS*(*s, t*) (*Breadth-first search*)

This procedure gives the shortest path of lengths between s and t on the defined residual network $G(x)$, where all lengths of arcs are equal to one.

Initialization:

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

Set $V := \{1,2,...,n\}$, $|V| = n$ /Cardinality of *V*/

Set $k := 0$, $P := \phi, P_o := \phi$

Iteration k:

While *(*$|P| \neq |V|$*)* then do /$|P|$ cardinality of P /

Let $P_k := \{j : d(j) = k\}$ and $P := \{j : d(j) \leq k\}$

Let $\Gamma^+(P_k) := \{j : (i,j) \in G(x), i \in P_k, j \in V\}$ /set of successors of P_k elements in $G(x)$ /

Let $\bar{P} := V - P$ /Compliment set of P /

Set $P_{k+1} := \Gamma^+(P_k) \bigcap \bar{P}$

If $P_{k+1} = \phi$, ***then end the algorithm***

Else

Set $d(j) := k + 1$ for all $j \in P_{k+1}$

Set $P := P \bigcup P_{k+1}$

Set $k := k + 1$

End If

End while

End the procedure

Notes

a. After the application of the procedure $BFS(s,t)$ 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 $BFS(s,t)$ on the defined residual network, if $t \in P$ $(t \notin \bar{P})$, 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 one. Note that, we do not need a general shortest path algorithm of Dijkstra with $O(n^2)$ arithmetic operations to find an augmenting path with the fewest number of arcs. We can simply use breadth-first search (BFS) illustrated above with $O(m)$ arithmetic operations (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 Identification of the shortest path μ from s to t on the defined residual network

Initialization:

Set $i := t$

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

Iteration:

While $(i \neq s)$ do

Find j such that $d(j) := d(i) - l_{ji}$

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 $BFS(s,t)$, where all lengths of arcs are equal to one, is $O(m)$ arithmetic operations, where m is the number of arcs in the network $G = (V, E)$ .The maximum number of iterations of the algorithm is $m \times r$,where r is the smallest integer greater than or equal to $\log B$, where B is the largest arc capacity of the network. The procedure $BFS(s,t)$ is applied once time, in each iteration, then the time taken by the algorithm is at most $O(m^2 r)$ 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 $0 \le y_k \le b_k^*$ 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 is added in the node j , where $j = 1,...,n$. We also define 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 second 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 with capacity $b_{t^*s^*}^* = \infty$.

This new defined digraph will be denoted by $G^*(V^*, E^*)$, which 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 $y_k := 0$ and $B_k^* := b_k^*$ 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 $w = \sum_{\{j \in V : w_j > 0\}} w_j$

Iteration:

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

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

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

Set $y_k := 2y_k$ 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) $y_k < b_k^*$, then do

Do procedure $BFS(s^*, t^*)$ 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 μ

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 μ

End If(2)

Do procedure $BFS(s,t)$ 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 μ

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 μ

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 into 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 equal to one ($l_{ts} = 1, l_{t^*s^*} = 1$), therefore, 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 $G^*(V^*, E^*)$, therefore, 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. $y_{t^*s^*}=w$, 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 some auxiliary arcs in $G^*(V^*,E^*)$ are not saturated, i.e. $y_{t^*s^*}<w$, 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 $BFS(s,t)$, where all lengths of arcs are equal to one, is $O(m^*)$ arithmetic operations, where m^* is the number of arcs in the network $G^*=(V^*,E^*)$.The maximum number of iterations of the algorithm is $m^*\times r^*$,where 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 $BFS(s,t)$ is applied twice a time in each iteration, then the time taken by the algorithm is at most $O((m^*)^2 r^*)$ arithmetic operations.

4 Maximum Flow Problem With Infinite Upper Bound On The Flow Vector

Two cases have already been treated in this paper; in first case 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 $k=1,...,m$ while the second case 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 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 $k=1,...,m$. While the second case 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 $BFS(s,t)$ 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 replace the infinity ∞ in the original network $G = (V, E)$ by the value of β and solve it anew by the proposed algorithm.

Now, in 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 Conclusions

A polynomial time algorithm for the maximum flow problem has been developed in this paper, using the successive divisions of capacities by multiples of two. The algorithm runs in $O(m^2 r)$ time, where m is the number of arcs of the network $G(V, E)$ and r is the smallest integer greater than or equal to $\log B$, and B is the largest arc capacity of the network. The algorithm solves the maximum flow problem as a sequence of $O(m)$ shortest path problems on residual networks.

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

6 Illustrative Example

The demonstration of the proposed algorithm for solving the maximum flow problem will be done through the following numerical example (Fig. 1-4).

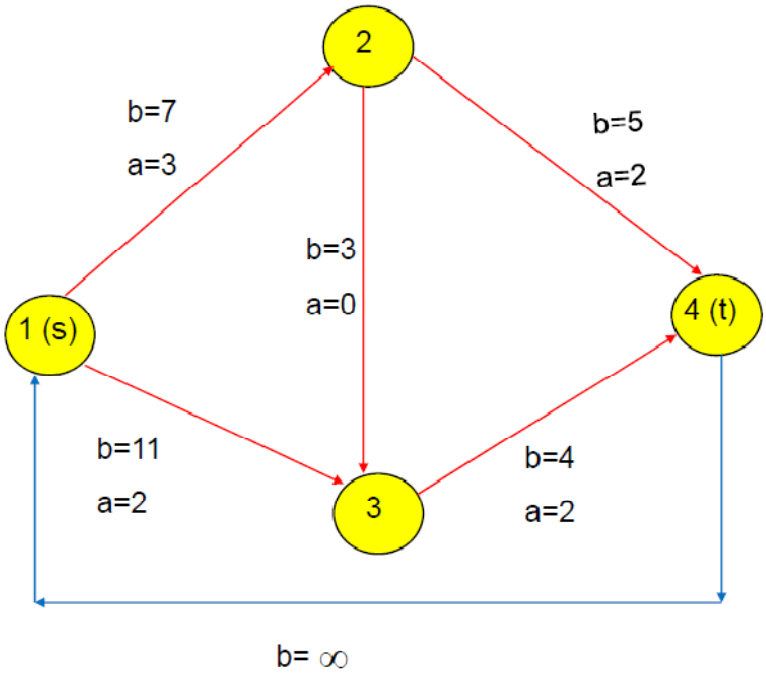

Fig. 1 Diagram of example with nonnegative lower bound on the flow vector (network $G(V, E)$).

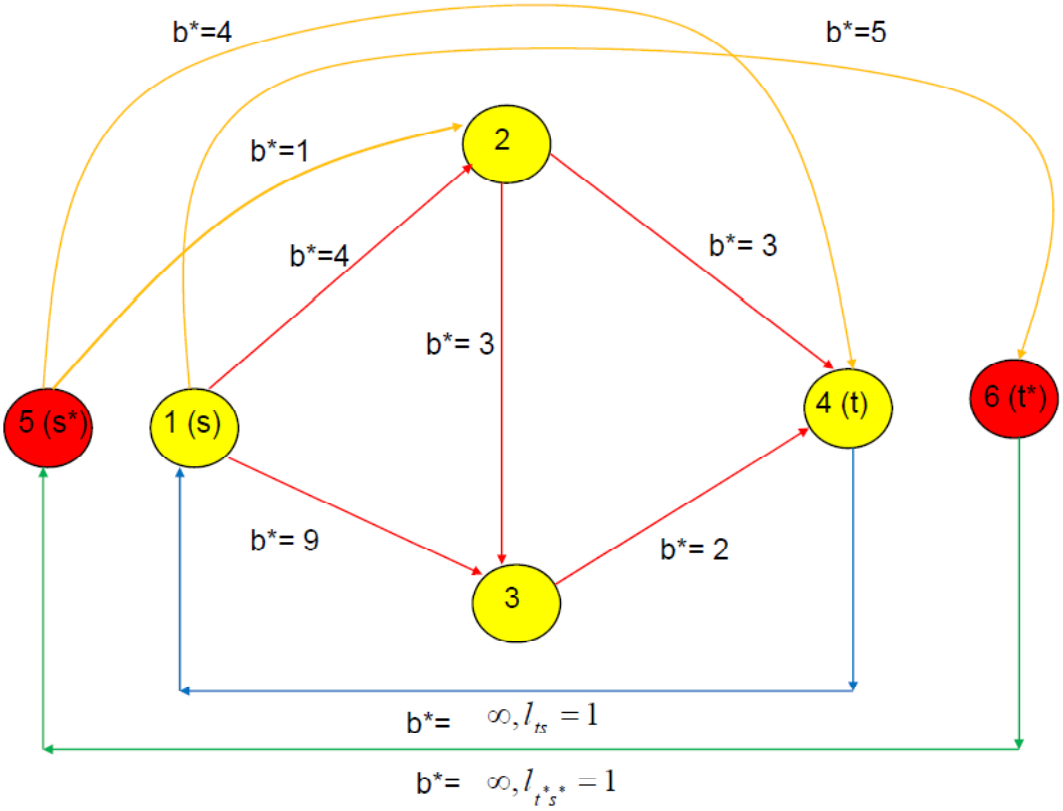

Fig. 2 Diagram of example with added auxiliary arcs (network $G^*(V^*, E^*)$).

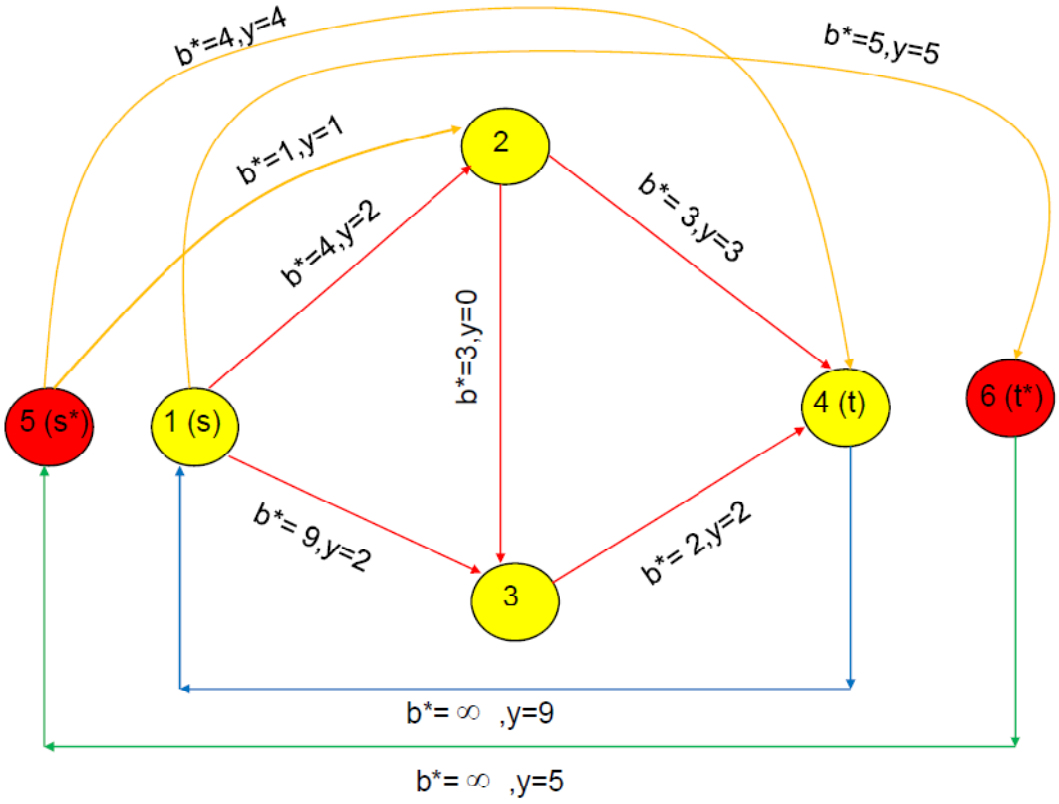

Fig. 3 Diagram of solution with added auxiliary arcs (network $G^*(V^*, E^*)$).

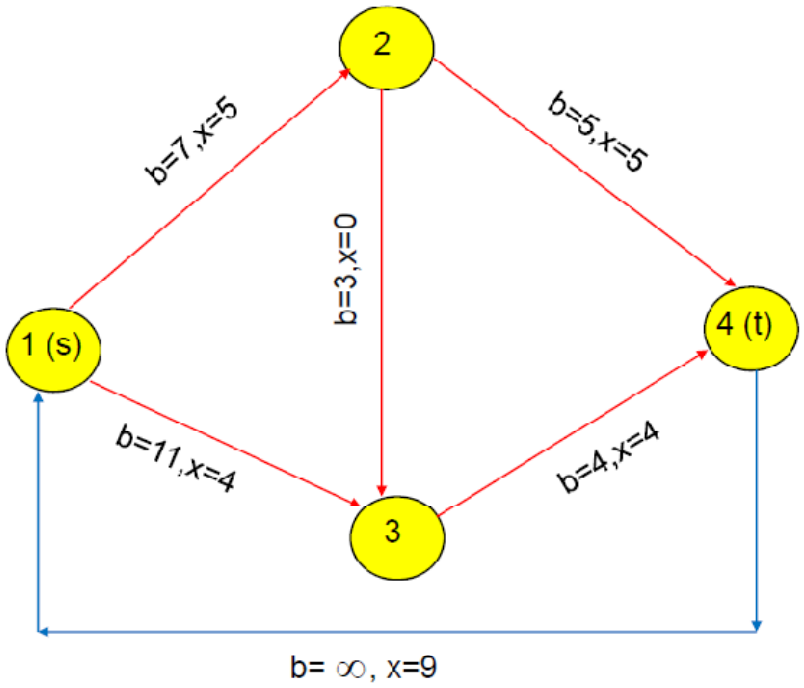

Fig. 4 Diagram of solution with nonnegative lower bound on the flow vector (network $G(V, E)$).

Acknowledgment

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 Transactions 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 (Norwalk, Conn.), 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. Analysing weighted networks. 2009. 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 J. D. Velasquez, et al., (Eds.), KES 2009 Proceedings of the 13th International Conference on Knowledge- Based and Intelligent Information and Engineering Systems: Part II: Berlin, Germany, Springer-Verlag, 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 (Norwalk, Conn.), 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 (Bui A, et al., eds). 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 (Jiang T, ed). 6363: 172-180, Berlin, Springer-Verlag, Germany

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

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. 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