Journal of Physical Sciences, Vol. 19, 2014, 41-50
ISSN: 2350-0352 (print), www.vidyasagar.ac.in/journal
Published on 26 December 2014

An Efficient Algorithm for Finding Maximum Flow in a Network-Flow

Faruque Ahmed, Md. Al-Amin Khan, Aminur Rahman Khan, Syed Sabbir Ahmed and Md. Sharif Uddin

Department of Mathematics, Jahangirnagar University, Bangladesh
E-mail: aminurju@yahoo.com

Received 29 April 2014; accepted 30 August 2014

ABSTRACT
This paper aims to introduce a new efficient algorithmic approach for finding the maximum flow of a maximal flow problem requiring less number of iterations and augmentation than Ford-Fulkerson algorithm.

Keywords: Maximal-Flow Model, Residual network, Residual Capacity, Augmenting Path, Source-Sink cut, Source-Sink cut capacity, Bounded variable simplex method.

1. Introduction

Networking deals a great section of operation research. many problems of our daily life can be presented by network model. There are four types of network model: Shortest-path model, Minimum spanning tree model, Maximal-flow model and Minimum-cost capacitated network model. In this paper we have worked on Maximal-flow model. The objective of the maximal flow problem is to find the maximum flow that can be sent from specified node source (s) to specified node sink (t) through the edges of the network. The maximum flow problem asks for the largest amount of flow that can be transported from one vertex (source) to another (sink). Originally, the maximal flow problem was invented by Fulkerson and Dantzig and solved by specializing the simplex method for the linear programming; and Ford and Fulkerson solved it by augmenting path algorithm.

The literature on network flow problem is extensive. In the mid of 1950, Air Force researchers T. E. Harris and F. S. Ross published a report studying the rail network that linked the Soviet Union to its satellite countries in Eastern Europe. In 1955, Lester R. Ford and Delbert R. Fulkerson created the first known algorithm to obtain the maximum flow in a flow network, the Ford-Fulkerson algorithm. Over the past fifty years researchers have improved several algorithms for solving Maximal-flow problems. In this paper, we have introduced a new algorithm to find the maximum flow in a network and formulated as a linear programming problem (LPP) and then solved it using proposed algorithm.

2. Proposed algorithm

The steps of our proposed algorithm are summarized below.
Step-1: Initialize $f(u, v) = f(v, u) = 0$, for each edge $(u, v) \in E$.
Step-2: Set *flow value* = 0.

Step-3: If there exist any cut $[S, T]$ such that $cap(S, T)$ is equal to the *flow value*, then go to step-8, otherwise go to step-4.
Step-4: Select an augmenting path p from *source* to *sink* in the residual network G_f.
Step-5: Set $c_f(p) = \min_{(u,v)\in p} \{c_f(u, v)\}$.
Step-6: For each $(u, v) \in p$, if $(u, v) \in E$, set $f(u, v) = f(u, v) + c_f(p)$,
else $f(v, u) = f(v, u) - c_f(p)$.
Step-7: Set *flow value* = *flow value* + $c_f(p)$ and Return to step-3.
Step-8: The maximum flow = *flow value.*

3. Illustrative Example

We consider the flow network given by figure 1. Here the source node is denoted by 1 and the sink node is denoted by 8. The capacities are shown on the respective arcs. It is required to find the maximum flow in this network from source 1 to sink 8.

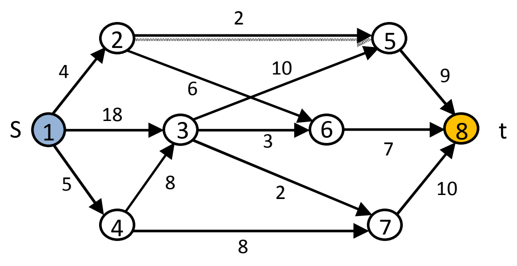


Figure 1:

Now we construct the following source-sink cut $[S, T]$ table from Figure 1.

Source-sink cut $[S,T]$		$cap(S,T)$	Source-sink cut $[S,T]$		$cap(S,T)$
S	T		S	T	
{1}	{2, 3, 4, 5, 6, 7, 8}	27	{1, 2, 3, 6}	{4, 5, 7, 8}	26
{1, 2}	{3, 4, 5, 6, 7, 8}	31	{1, 3, 4, 6}	{2, 5, 7, 8}	31
{1, 3}	{2, 4, 5, 6, 7, 8}	24	{1, 3, 4, 7}	{2, 5, 6, 8}	27
{1, 4}	{2, 3, 5, 6, 7, 8}	38	{1, 2, 3, 4, 5}	{6, 7, 8}	28
{1, 2, 3}	{4, 5, 6, 7, 8}	28	{1, 2, 3, 4, 6}	{5, 7, 8}	29
{1, 2, 4}	{3, 5, 6, 7, 8}	42	{1, 2, 3, 4, 7}	{5, 6, 8}	31
{1, 2, 5}	{3, 4, 6, 7, 8}	38	{1, 2, 3, 5, 6}	{4, 7, 8}	23
{1, 3, 6}	{2, 4, 5, 7, 8}	28	{1, 3, 4, 6, 7}	{2, 5, 8}	31
{1, 4, 7}	{2, 3, 5, 6, 8}	40	{1, 2, 3, 4, 5, 6}	{7, 8}	26
{1, 2, 3, 4}	{5, 6, 7, 8}	31	{1, 2, 3, 4, 6, 7}	{5, 8}	29
{1, 2, 3, 5}	{4, 6, 7, 8}	25	{1, 2, 3, 4, 5, 6, 7}	{8}	26

Table1: Source-sink cut and its capacity

Initialization:

Initialize the value of *f* for each edge to 0 in the flow network *G*.

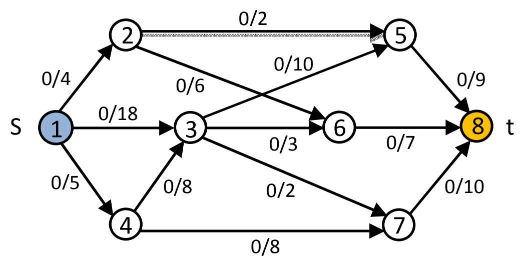


Figure 2:

The value of flow = 0. We see that there does not exist any source-sink cut $[S, T]$ in table-1 such that $cap(S, T) = 0$. Therefore, the flow is not maximum.

Iteration-1:

The residual network G_f (Figure-3) of the initial flow network (Figure-2) and the new resulting flow network (Figure-4) are as follows:

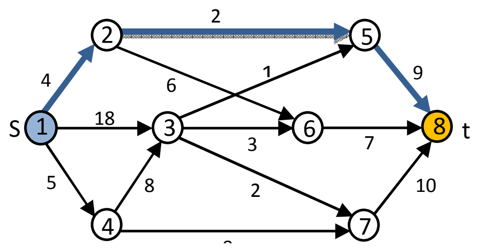


Figure 3:

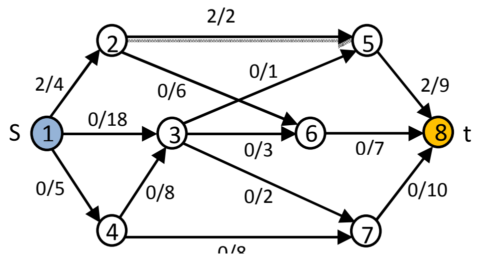


Figure 4:

In this case, $c_f(p)$ = min {4, 2, 9} = 2 and so, flow value = 0 + $c_f(p)$ = 0 + 2 = 2. Thus we see that there does not exist any source-sink cut $[S, T]$ in table-1 such that $cap(S, T) = 2$. Therefore, the flow is not maximum.

Iteration-2:

The residual network G_f (Figure-5) of the resulting flow network (Figure-4) and the new resulting flow network (Figure-6) are as follows:

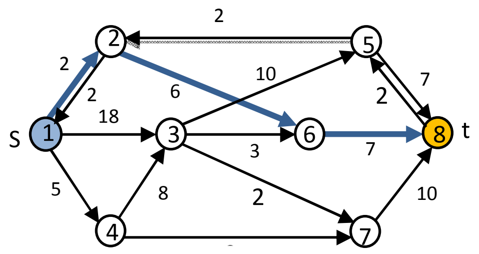


Figure 5:

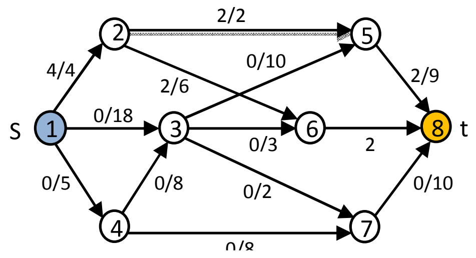


Figure 6:

In this case, $c_f(p)$ = min {2, 6, 7} = 2 and so, flow value = 2 + $c_f(p)$ = 2 + 2 = 4. Thus we see that there does not exist any source-sink cut $[S, T]$ in table-1 such that $cap(S, T) = 4$. Therefore, the flow is not maximum.

Iteration-3:

The residual network G_f (Figure-7) of the resulting flow network (Figure-6) and the new resulting flow network (Figure-8) are as follows:

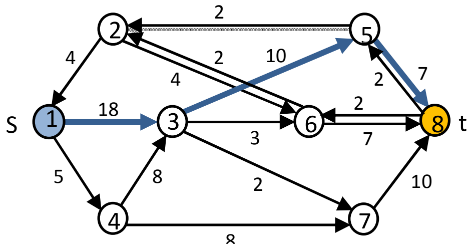


Figure 7:

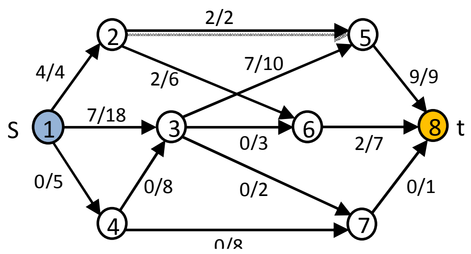


Figure 8:

In this case, $c_f(p)$ = min {18, 10, 7} = 7 and so, flow value = 4 + $c_f(p)$ = 4 + 7 = 11. Thus we see that there does not exist any source-sink cut $[S, T]$ in table-1 such that $cap(S, T) = 11$. Therefore, the flow is not maximum.

Iteration-4:

The residual network G_f (Figure-9) of the resulting flow network (Figure-8) and the new resulting flow network (Figure-10) are as follows:

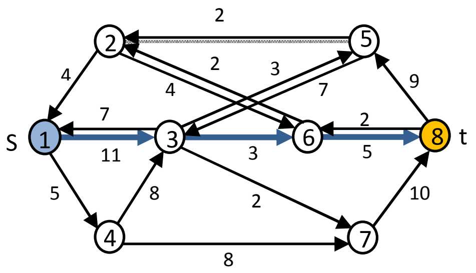


Figure 9:

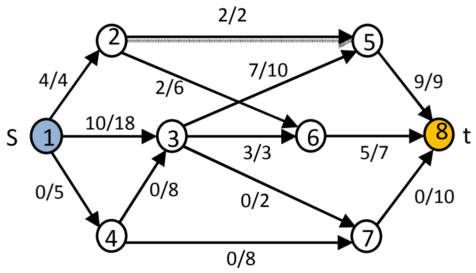


Figure 10:

In this case, $c_f(p)$ = min {11, 3, 5} = 3 and so, flow value = 11 + $c_f(p)$ = 11 + 3 = 14. Thus we see that there does not exist any source-sink cut $[S, T]$ in table-1 such that $cap(S, T) = 14$. Therefore, the flow is not maximum.

Iteration-5:

The residual network G_f (Figure-11) of the resulting flow network (Figure-10) and the new resulting flow network (Figure-12) are as follows:

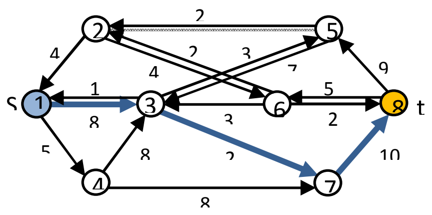


Figure 11:

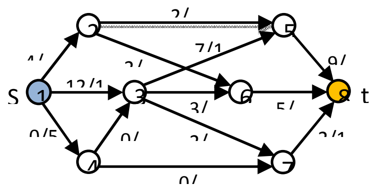
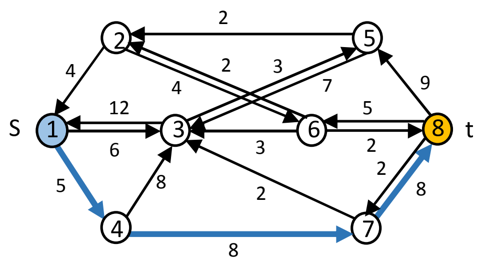


Figure 12:

In this case, $c_f(p)$ = min {8, 2, 10} = 2 and so, flow value = 14 + $c_f(p)$ = 14 + 2 = 16. Thus we see that there does not exist any source-sink cut $[S,T]$ in table-1 such that $cap(S,T) = 16$. Therefore, the flow is not maximum.

Iteration-6:
The residual network G_f (Figure-13) of the resulting flow network (Figure-12) and the new resulting flow network (Figure-14) are as follows:



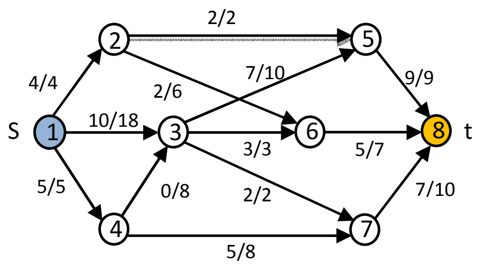
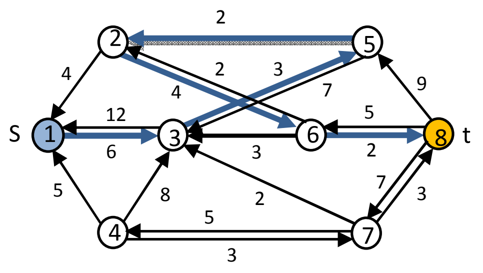


Figure 13: **Figure 14:**

In this case, $c_f(p)$ = min {5, 8, 8} = 5 and so, flow value = 16 + $c_f(p)$ = 16 + 5 = 21. Thus we see that there does not exist any source-sink cut $[S,T]$ in table-1 such that $cap(S,T) = 21$. Therefore, the flow is not maximum.

Iteration-7:
The residual network G_f (Figure-15) of the resulting flow network (Figure-14) and the new resulting flow network (Figure-16) are as follows:



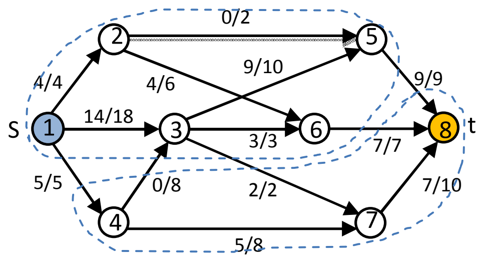


Figure 13: **Figure 14:**

Here, $c_f(p)$ = min {6, 3, 2, 4, 2} = 2 and so, flow value = 21 + $c_f(p)$ = 21 + 2 = 23. In this stage, we see that there exists a source-sink cut $[S,T]$ in table-1, where $S = \{1,2,3,5,6\}$, and $T = \{4,7,8\}$ such that $cap(S,T)$ = 23. So, the algorithm terminates and the flow in iteration-7 is therefore maximum flow. The value of maximum flow through the given network is 23.

4. Solution using Ford-Fulkerson algorithm

Now we are going to solve the same network flow problem by using Ford-Fulkerson algorithm. The procedure in each iteration is briefly summarized below:

Iteration-1 : The augmenting path is: $1 \to 2 \to 5 \to 8$ with residual capacity 2.
Flow value $= 2 + 0 + 0 = 2$.

Iteration-2 : The augmenting path is: $1 \to 2 \to 6 \to 8$ with residual capacity 2.
Flow value $= 4 + 0 + 0 = 4$.

Iteration-3 : The augmenting path is: $1 \to 3 \to 5 \to 8$ with residual capacity 7.
Flow value $= 4 + 7 + 0 = 11$.

Iteration-4 : The augmenting path is: $1 \to 3 \to 6 \to 8$ with residual capacity 3.
Flow value $= 4 + 10 + 0 = 14$.

Iteration-5 : The augmenting path is: $1 \to 4 \to 3 \to 7 \to 8$ with residual capacity 2.
Flow value $= 4 + 10 + 2 = 16$.

Iteration-6 : The augmenting path is: $1 \to 4 \to 7 \to 8$ with residual capacity 3.
Flow value $= 4 + 10 + 5 = 19$.

Iteration-7 : The augmenting path is: $1 \to 3 \to 4 \to 7 \to 8$ with residual capacity 2.
Flow value $= 4 + 12 + 5 = 16$.

Iteration-8 : The augmenting path is: $1 \to 3 \to 5 \to 2 \to 6 \to 8$ with residual capacity 2.
Flow value $= 4 + 14 + 5 = 23$.

Iteration-9 : There does not exist any augmenting path from source to sink. So, the algorithm terminates and the flow in iteration-8 is therefore maximum flow.
The maximum flow value $= 23$.

5. Comparison

In solving the same network flow problem, we see that the Ford-Fullkerson algorithm takes 9 iterations selecting 8 augmenting paths and after the end of 9th iteration we obtain the maximum amount of flow, whereas this proposed algorithm takes 7 iterations selecting 7 augmenting paths and after the end of 7th iteration we obtain the maximum flow.

6. Bounded Variable Simplex method

In a linear programming problem some or all the variables may have lower and upper bounds, that is, constraints of the type $l_j \leq x_j \leq u_j$, where x_j is the variable of the given problem and l_j and u_j are its lower and upper bounds respectively.

The lower bound constraint of the type $l_j \leq x_j$ can be handled directly by substituting $x_j = l_j + x_j'$, where $x_j' \geq 0$. But for an upper bound constraint of the type $x_j \leq u_j$, the substitution $x_j = u_j - x_j''$, $x_j'' \geq 0$ does not guarantee that x_j will remain non-negative. This difficulty is overcome by using a special type simplex method called *bounded variable simplex method.*

The algorithm for the computational procedure to obtain an optimum solution of any LPP by bounded variable simplex method consists of the following steps:

Step-1 : Check whether the objective function of the given LPP is to maximized or minimized. If the given LPP is to be minimized, then we convert it into a maximization problem using the following formula:

$$\text{Minimize } z = -\text{ Maximize } (-z).$$

Step-2 : Check whether the right hand sides of all constraints are non-negative. If any one of them is negative, then multiply the corresponding inequality of the constraint by -1.

Step-3 : Convert all the inequality of the constraints into equations by introducing slack and / or surplus variables in the constraints.

Step-4 : Reformulate the given LPP as a standard maximization LPP.

Step-5 : Obtain an initial basic feasible solution x_B^* to the problem and put it in the appropriate column of the table.

Step-6 : Compute z_j using the following formula:

z_j = Sum of the products of the corresponding elements in the columns of c_B and x_j.

Step-7 : Compute the net evaluation $z_j - c_j$.

Step-8 : Check the sign of $z_j - c_j$.

(i) If all $z_j - c_j \geq 0$, then the basic feasible solution x_B^* is an optimum basic feasible solution. Goto step-14.

(ii) If at least one $z_j - c_j < 0$, then continue with the next step.

Step-9 : If there are more than one negative of $z_j - c_j$, then choose the most negative of them. The tie can be made broken arbitrarily. Let one of them be $z_r - c_r$, for some $j = r$.

Step-10 : The entering variable (non-basic) is then x_r corresponding to most negative $z_r - c_r$. Suppose that a_k^r be the coefficients of the corresponding basic variables x_k in the column of the entering variable x_r; where the index *k* is associated with the basic variables and the index *r* is associated with the entering variable.

Step-11 : Check the sign of a_k^r. Then compute the following quantities:

(i) For those k such that $a_k^r > 0$, compute $\theta_1 = \min\limits_k \left\{\frac{(x_B^*)_k}{a_k^r} > 0\right\}$ and note the basic variable that corresponds to it. If $a_k^r > 0$ does not exist for any k, then $\theta_1 = \infty$.

(ii) For those k such that $a_k^r < 0$, compute $\theta_2 = \min\limits_k \left\{\frac{u_k - (x_B^*)_k}{-a_k^r} > 0\right\}$ and note the basic variable that corresponds to it. If $a_k^r < 0$ does not exist for any k, then $\theta_2 = \infty$.

Step-12 : Compute $\theta = min\,\{\theta_1,\ \theta_2,\ u_r\}$ and note the variable that corresponds to it. Then we must follow any one of the following three tasks:

(i) If $\theta = \theta_1$be obtained corresponding to the basic variable x_k, then x_k leaves the solution and x_r enters into the basis by using the procedure of pivoting of simplex method and then goto step-13.

(ii) If $\theta = \theta_2$be obtained corresponding to the basic variable x_k, then $x_k$leaves the solution and x_r enters into the basis by using the procedure of pivoting of simplex method.
After then x_k being non-basic at its upper bound must be substituted out by using $x_k = u_k - x_k'$ (where $0 \le x_k' \le u_k$), and then goto

(iii) If $\theta = u_r$be obtained corresponding to the entering variable x_r, then x_r cannot be entered into the basis and therefore x_r is substituted by using $x_r = u_r - x_r'$ (where $0 \le x_r' \le u_r$) while it remains non-basic.

Step-13 : Repeat step-6 for computational procedure until an optimum solution is obtained.

Step-14 : End computational procedure.

7. LP Formulation of maximal-flow model

Let f be the amount of flow from source node s to sink node t and x_{ij} be the flow from node i to node j over the arc $(i,\ j)$ in a flow network $G = (V, E)$. Then the LP formulation of the flow network is as follows:

Maximize: f

Subject to: $\sum_{j\in V} x_{sj} - \sum_{k\in V} x_{ks} = f,$

$\sum_{j\in V} x_{tj} - \sum_{k\in V} x_{kt} = -f,$

$\sum_{j\in V} x_{ij} - \sum_{k\in V} x_{ki} = 0,$ for all $i \in V - \{s,\ t\},$

$0 \le x_{ij} \le u_{ij}$, for all $(i,\ j) \in E,$

where u_{ij} is the upper bound of the flow over the arc $(i,\ j)$.

8. Solution using bounded variable simplex method

Now we are going to find the maximum flow in the flow network given in the figure-1 by using Bounded Variable Simplex Method. The associated LP is summarized by the following table with two different, but equivalent, objective functions depending on whether maximize the output from source node 1 (= z_1) or the input to sink node 8 (= z_2).

	x_{12}	x_{13}	x_{14}	x_{25}	x_{26}	x_{35}	x_{36}	x_{37}	x_{43}	x_{47}	x_{58}	x_{68}	x_{78}	
Max z_1	1	1	1											
Max z_2											1	1	1	
Node 2	1			-1	-1									= 0
Node 3		1				-1	-1	-1	1					= 0
Node 4			1						-1	-1				= 0
Node 5				1		1					-1			= 0
Node 6					1		1					-1		= 0
Node 7								1		1			-1	= 0
Capacity	**4**	**18**	**5**	**2**	**6**	**10**	**3**	**2**	**8**	**8**	**9**	**7**	**10**	

Thus the Linear programming problem (LPP) becomes

Maximize: $z = x_{12} + x_{13} + x_{14}$ (‘OR’ Maximize: $z = x_{58} + x_{68} + x_{78}$)

Subject to:

$$x_{12} - x_{25} - x_{26} = 0$$
$$x_{13} - x_{35} - x_{36} - x_{37} + x_{43} = 0$$
$$x_{14} - x_{43} - x_{47} = 0$$
$$x_{25} + x_{35} - x_{58} = 0$$
$$x_{26} + x_{36} - x_{68} = 0$$
$$x_{37} + x_{47} - x_{78} = 0$$

$0 \leq x_{12} \leq 4$, $0 \leq x_{13} \leq 18$, $0 \leq x_{14} \leq 5$, $0 \leq x_{25} \leq 2$, $0 \leq x_{26} \leq 6$, $0 \leq x_{35} \leq 10$, $0 \leq x_{36} \leq 3$, $0 \leq x_{37} \leq 2$, $0 \leq x_{43} \leq 8$, $0 \leq x_{47} \leq 8$, $0 \leq x_{58} \leq 9$, $0 \leq x_{68} \leq 7$, $0 \leq x_{78} \leq 10$.

Solving the above linear programming problem (LPP) by using Bounded Variable Simplex Method (Section-7), the optimal solution is:

$x_{12} = 4$, $x_{13} = 14$, $x_{14} = 5$, $x_{25} = 0$, $x_{26} = 4$, $x_{35} = 9$, $x_{36} = 3$,
$x_{37} = 2$, $x_{43} = 0$, $x_{47} = 5$, $x_{58} = 9$, $x_{68} = 7$ and $x_{78} = 7$.

The associated maximum amount of flow is: $z = x_{12} + x_{13} + x_{14} = 4 + 14 + 5 = 23$.

9. Conclusion

We have provided a new algorithm for finding the amount of maximum flow from source (s) to sink (t) in a flow network. A numerical example is solved to illustrate the proposed algorithm and for the same problem the procedure at each iteration by using Ford-Fulkerson algorithm is briefly summarized. We have also solved the same flow network problem by using bounded variable simplex method. Our proposed algorithm returns the maximum flow that takes less number of iterations and augmentations than the Ford-Fulkerson Algorithm.

REFERENCES

1 L.R.Ford and D.R.Fulkerson, Maximal flow through a network, *Canadian Journal of Mathematics*, *8* (1956) 399-404.

2 D.R.Fulkerson and G.B.Dantzig, Computation of maximum flow in network, *Naval Research Logistics Quarterly*, 2 (1955) 277-283.

3. V.A.Goldberg and R.E.Tarjan, A new approach to the maximum-flow problem, *Journal of the ACM*, 35(4) (1988) 921-940.

4. C.Jain and D.Garg, Improved Edmond-Karps algorithm for network flow problem, *International Journal of Computer Applications*, 37(1), (2012) 48-53.
5. R.K.Ahuja and J.B.Orlin, A fast and simple algorithm for the maximum flow problem, *Operations Research*, 35(5) (1989) 748-759.
6. Md. Al-Amin Khan, A.Rashid, A.R.Khan and Md. Sharif Uddin, An Innovative Approach for Solving Maximal-Flow Problems, *Journal of Physical Sciences*, 17 (2013) 143-154.
7. S.Mandal and M.Pal, A sequential algorithm to solve next-to-shortest path problem on circular-arc graphs, *Journal of Physical Sciences*, 10 (2006) 201-217.
8. M.Bazarra and J.Jarvis, Linear Programming and Network Flows, John Wiley & Sons, 1977.
9. J.Edmonds and R.M.Karp, Theoretical improvements in algorithmic efficiency for network flow problems, *Journal of the ACM*, 19(2) (1972) 248-264.