Journal of Physical Sciences, Vol. 17, 2013, 143-154
ISSN: 0972-8791, www.vidyasagar.ac.in/journal
Published on 26 December 2013

An Innovative Approach for Solving Maximal-Flow Problems

Md. Al-Amin Khan, Abdur Rashid, Aminur Rahman Khan and Md. Sharif Uddin

Department of Mathematics, Jahangirnagar University, Bangladesh.
Email: aminur@juniv.edu

Received 11 November 2013; accepted 11 December 2013

ABSTRACT

This paper aims at introducing a new approach for finding the maximum flow of a maximal- flow problem requiring less number of iterations and less augmentation than Ford-Fulkerson algorithm. To illustrate the proposed method, a numerical example is presented. We have also formulated the maximal-flow problem as a linear programming problem (LPP) and solved it by using Bounded Variable Simplex Method.

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

1. Introduction

Network flow problems have always been among the best studied combinatorial optimization problems. Maximal-flow problem is the classical network flow problem in weighted graphs. The objective of the maximal flow problem is to find the maximum flow that can be sent through the arc of the network from some specified node source (s) to specified node sink (t). Maximal flow problems play an important role in a number of practical contexts including design and operation of telecommunication networks, oil-pipeline systems, water through a system of aqueducts etc [2]. Maximal flow problem can be formulated as an LPP and hence could be solved by usual simplex method. In literature, a good amount of research [5,6,7] is available for solving such kind of problems. Originally the maximal flow problem was invented by Fulkerson and Dantzig [1] and solved by specializing the simplex method for the linear programming, and Ford and Fulkerson [3] solved it by augmenting path algorithm. The improvement of the Ford-Fulkerson method is Edmonds-Karp algorithm [4] which performs better than the previous one. C. Jain and D. Garg [8] proposed an improved version of Edmonds-Karp algorithm to solve the maximum flow problem, which requires less number of iterations and less augmentation to calculate the maximum flow. The algorithm [9,10] is based on finding breakthrough paths with net positive flow between the source and sink nodes. In this paper we have proposed an effective algorithm to find maximum flow in network and formulated as an LPP and solved it by using Bounded Variable Simplex Method.

2. Preliminaries

In this section some basic definitions and notations are reviewed related to maximal-flow problem.

2.1. Flow network

Let $G = (V, E)$ be a directed graph with vertex set V and edge set E. A *flow network* $G=(V,E)$ is a directed graph in which each edge $(u, v) \in E$ has a nonnegative capacity $c(u, v) \geq 0$ and a distinguished source vertex s and sink vertex t [11]. If $(u, v) \notin E$, then for convenience we define $c(u, v) = 0$. A *flow* in G is a real-valued function $f : V \times V \to IR$, that satisfies these constraints:

$f(u, v) \leq c(u, v)$ for all $(u,v) \in V \times V$ (capacity constraint),

$f(u,v) = -f(v,u)$ for all $(u,v) \in V \times V$ (antisymmetry constraint),

$\sum_{u \in V} f(u,w) = \sum_{v \in V} f(w,v)$ for all $w \in V - \{s, t\}$ (flow conservation constraint).

The *value* of a flow f is denoted by $|f|$ and defined as $|f| = \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s)$.

2.2. Residual network and residual capacity

For a given flow network G and a flow f, the residual network G_f consists of edges with capacities that represent how we can change the flow on edges of G. An edge of the flow network can admit an amount of additional flow equal to the edge's capacity minus the flow on that edge. If that value is positive, we place that into G_f with a "residual capacity" of $c_f(u,v) = c(u,v) - f(u,v)$. The only edges of G that are in G_f are those that can admit more flow; those edges (u,v) whose flow equals their capacity $c_f(u,v) = 0$, and they are not in G_f.

2.3. Augmenting path

An *augmenting path* p in a network $G = (V,E)$ with a flow f is a path from s to t in which every edge has positive capacity in the residual network G_f . We can put more flow from s to t through p. We call the maximum capacity by which we can increase the flow on p the residual capacity of p, given by $c_f(p) = \min \{c_f(u, v) : (u, v) \text{ is on } p\}$.

2.4. Source-Sink cut and its capacity

A *source-sink cut* $[S,T]$ of flow network $G=(V,E)$ consists of the edges from a source set S to a sink set T, where S and T partition the set of nodes, with $s \in S$ and $t \in T$.

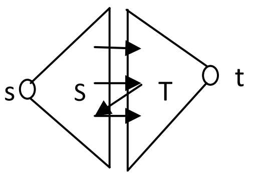

Figure 1:

The *capacity* of the cut $[S,T]$, written cap(S,T), is the total of the capacities on the edges of $[S,T]$, that is,

$$\text{cap}(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v).$$

Note that in a directed network $[S,T]$ denotes the set of edge with tail in S and head in T. Thus the capacity of a cut $[S,T]$ is completely unaffected by edges from T to S.

3. A proposed algorithm

The major steps of the algorithms are given below:

Step 1	:	For each edge $(u, v) \in E$, initialize $f(u, v) = f(v, u) = 0$.
Step 2	:	Calculate lower capacity (L_c) and upper capacity (U_c) in the flow network and then calculate $D = U_c - L_c$.
Step 3	:	If $D = 0$, then set $D = U_c$ *or* L_c.
Step 4	:	If there exists an augmenting path p from s to t in the residual network G_f with capacity at least D then select it; otherwise go to step 9.
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	:	Calculate the flow value.
Step 8	:	If there exist any source-sink cut $[S, T]$ such that cap(S, T) is equal to the flow value, then go to step 10; otherwise go to step 4.
Step 9	:	Set $D = \left\lfloor \frac{D}{2} \right\rfloor$. If $D \geq 1$ go to step 4; otherwise go to step10.
Step 10	:	The flow is maximum.

4. Numerical illustration

We consider the flow network given by Figure 2. Here the source node is denoted by 1 and the sink node is denoted by 6. The capacities are shown on the respective arcs. It is required to find the maximum flow in this network between source 1 to sink 6.

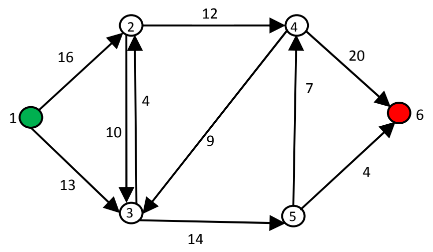

Figure 2:

Now we construct the following source-sink cut $[S, T]$ table from Figure 2.

Source-sink cut $[S, T]$		cap(S, T)
S	T	
{1}	{2, 3, 4, 5, 6}	16+13=29
{1, 2}	{3, 4, 5, 6}	12+13+10=35
{1, 3}	{2, 4, 5, 6}	16+4+14=34
{1, 2, 3}	{4, 5, 6}	12+14=26
{1, 2, 4}	{3, 5, 6}	13+10+9+20=52
{1, 3, 5}	{2, 4, 6}	16+4+7+4=31
{1, 2, 3, 4}	{5, 6}	14+20=34
{1, 2, 3, 5}	{4, 6}	12+7+4=23
{1, 2, 3, 4, 5}	{6}	20+4=24

Table 1: Source-sink cut and its capacity

Initialization: Initialize the value of f for each edge to 0. Here the flow network G is shown with each edge (u, v) labeled as $f(u, v) / c(u, v)$.

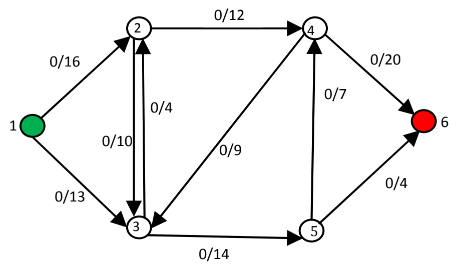

Figure 3:

Now the upper capacity in the flow network, $U_c = 20$ and the lower capacity in the flow network, $L_c = 4$. So, $D = U_c - L_c = 20 - 4 = 16$.

Iteration 1:

The residual network G_f of the initial flow network (Figure 3) is

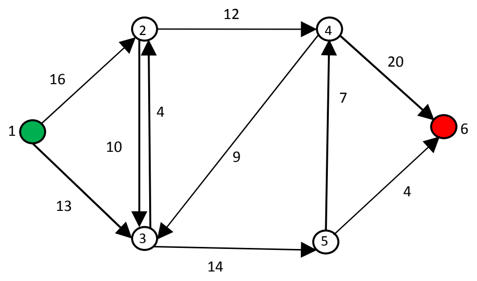

Figure 4:

Since $D = 16$, we have to choose an augmenting path with capacity at least 16. But there is no augmenting path with capacity at least 16.

Iteration 2:

$D = \left[\frac{D}{2}\right] = \left[\frac{16}{2}\right] = 8$. So, we have to choose an augmenting path with capacity at least 8 in Figure 4.

1st augmentation: An augmenting path found in 2nd iteration is $1 - 2 - 4 - 6$ with $c_f(p) = \min\{16, 12, 20\} = 12$.

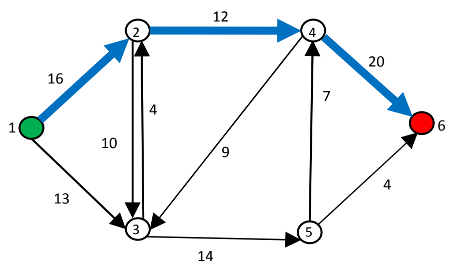

Figure 5:

Update the values of f for each edge along the path.

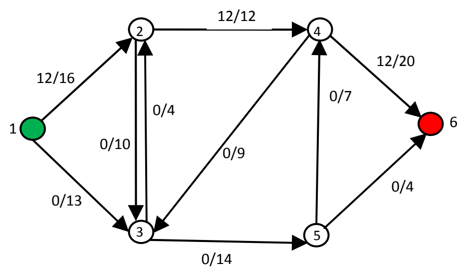

Figure 6:

Now there is no augmenting path with capacity at least 8.
The flow value $f = 12 + 0 = 12$.
We see that there does not exist any source-sink cut $[S, T]$ in table-1such that cap(S, T) = 12. Therefore, the flow is not maximum.

Iteration 3:

$D = \left[\frac{D}{2}\right] = \left[\frac{8}{2}\right] = 4$. Now the augmenting path with capacity at least 4 will be searched.

2nd augmentation:

Since $D = 4$, we select an augmenting path with capacity 4 in the residual network given by figure-7.

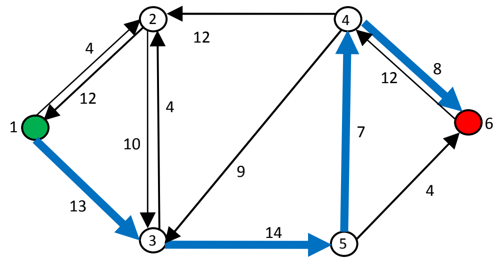

Figure 7:

The augmenting path found in third iteration is 1 – 3 – 5 – 4 – 6 with $c_f(p)$ = min {13, 14, 7, 8}= 7. Update the values of f for each edge along the path.

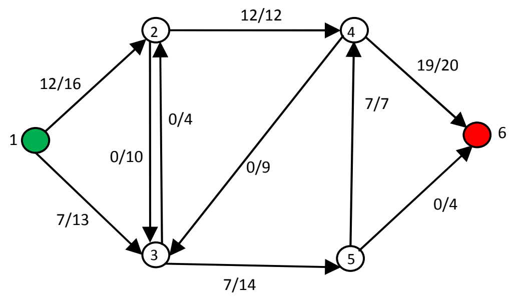

Figure 8:

The flow value f = 12 + 7 = 19.
We see that there does not exist any source-sink cut [S, T] in table-1such that cap(S, T) = 19. Therefore, the flow is not maximum.
The residual network after the 2nd augmentation is shown below

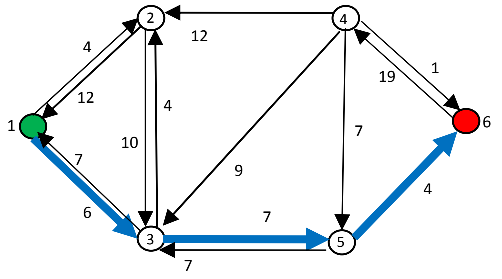

Figure 9:

3rd augmentation:

Now again there is a path with capacity at least 4 and the path found in the same 3rd iteration is 1 – 3 – 5 – 6 with $c_f(p)$ = min {6, 7, 4}= 4. Update the values of f for each edge along the path.

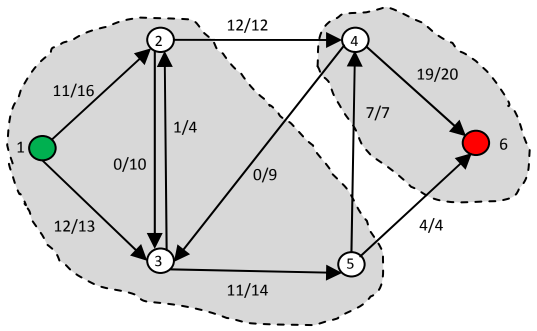

Figure 10:

Therefore, the resulting flow f =11 + 12 = 23.We see that there exists a source-sink cut $[S,T]$ in table-1, where S = {1, 2, 3, 5} and T = {4, 6}, such that cap(S, T) = 23. So the algorithm terminates and the flow in iteration 4 is therefore maximum flow. The value of the maximum flow through the network is 23.

5. Solution using Ford-Fulkerson algorithm

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

Iteration 1	:	Select the augmenting path 1 – 2 – 4 – 3 – 5 – 6 with capacity 4. Maximum flow value f = 4.
Iteration 2	:	Select the augmenting path 1 – 2 – 3 – 5 – 4 – 6 with capacity 7. Maximum flow value f = 4 + 7 = 11.
Iteration 3	:	Select the augmenting path 1 – 3 – 2 – 4 – 6 with capacity 8. Maximum flow value f = 11 + 8 = 19.
Iteration 4	:	Select the augmenting path 1 – 3 – 4 – 6 with capacity 4. Maximum flow value f = 19 + 4 = 23.

After 4th iteration there is no augmenting path with capacity at least 1. Thus, the algorithm terminates and the resulting flow in network returns the maximum flow. Therefore, Maximum flow value f = 23.

6. Comparison

In the Ford-Fulkerson algorithm only one augmenting path is possible to choose in each iteration but in our proposed algorithm we can choose zero (0) or more augmenting path in each iteration.

Now we construct the following table to compare between Ford-Fulkerson algorithm and our proposed algorithm.

Iteration No.	Ford-Fulkerson algorithm (No. of augmentation)	Our proposed algorithm (No. of augmentation)
1st	1	0
2nd	1	1
3rd	1	2
4th	1	Terminates in 3rd iteration

From the table we see that to calculate the maximum flow by using Ford-Fulkerson algorithm we need four augmenting paths with four iterations while by using our proposed algorithm we need only three augmenting paths with three iterations.

7. Bounded variable simplex method

In a linear programming problem some or all the variables may have lower or upper bounds *i.e.*, constraints of the type

$$l_j \le x_j \le u_j$$

where x_j is the jth variable of the problem and l_j and u_j are its lower and upper bounds respectively.

The lower bound constraint can be handled directly by substituting

$$x_j = l_j + x_j', \text{ where } x_j' \ge 0.$$

For an upper bound constraint of the type $x_j \le u_j$, the substitution $x_j = u_j - x_j''$, $x_j'' \ge 0$ does not guarantee that x_j will remain non-negative. This difficulty is overcome by using a special technique called *bounded variable simplex method*, which consists of the following steps:

Step 1 : In any constraint if the R.H.S. is negative, make it positive by multiplying the constraint by '– 1'.

Step 2 : If any constraint is in inequality, then convert the inequality into equations by adding suitable slacks or surplus variables and obtain an initial basic feasible solution.

Step 3 : Calculate the net evaluation $z_j - c_j$. For a maximization problem, if $z_j - c_j \ge 0$ for the non-basic variable, optimum basic feasible solution is attained. If $z_j - c_j < 0$ for any non-basic variable, go to step 4. For a minimization problem reverse is true.

Step 4 : Select the most negative of $z_j - c_j$.

Step 5 : Let x_j be a non-basic variable at zero level which is selected to enter the solution. Compute the quantities

$$\theta_1 = \min_i \left\{ \frac{(X_B^*)_i}{a_{ij}} \right\} \quad \text{where } a_{ij} > 0,$$

$$\theta_2 = \min_i \left\{ \frac{(X_B^*)_i - u_i}{a_{ij}} \right\} \quad \text{where } a_{ij} < 0,$$

$$\text{and} \quad \theta = \min \{ \theta_1, \theta_2, u_i \},$$

where u_i is the upper bound for the variable x_i. Let $(X_B)_r$ be the variable corresponding to $\theta = \min \{ \theta_1, \theta_2, u_i \}$. Then

(a) If $\theta = \theta_1$, $(X_B)_r$ leaves the solution and x_j enters by using the regular row operations of the simplex method.

(b) If $\theta = \theta_2$, $(X_B)_r$ leaves the solution and x_j enters; then $(X_B)_r$ being non-basic at its upper bound must be substituted out by using

$$(X_B)_r = u_r - (X_B)_r', \text{ where } 0 \le (X_B)_r' \le u_r.$$

(c) If $\theta = u_j$, x_j is substituted at its upper bound difference $u_j - x_j'$, while remaining non-basic.

8. 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 arc (i, j) in a flow network $G = (V, E)$.Then the LP formulation of the flow network is

Maximize f

Subject to:

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

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

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

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

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

Now we are going to find the maximum flow in the network given in Figure 2 by using Bounded Variable Simplex method. The associated Linear programming problem is

Maximize $z = x_{12} + x_{13}$

Subject to

$$x_{12} - x_{23} - x_{24} + x_{32} = 0$$
$$x_{13} + x_{23} - x_{32} - x_{35} + x_{43} = 0$$
$$x_{24} - x_{43} - x_{46} + x_{54} = 0$$
$$x_{35} - x_{54} - x_{56} = 0$$

$0 \le x_{12} \le 16$, $0 \le x_{13} \le 13$, $0 \le x_{23} \le 10$, $0 \le x_{24} \le 12$, $0 \le x_{32} \le 4$, $0 \le x_{35} \le 14$, $0 \le x_{43} \le 9$, $0 \le x_{46} \le 20$, $0 \le x_{54} \le 7$, $0 \le x_{56} \le 4$.

It will be very difficult when we will try to solve this LPP by the simplex method. Because we have to write the bounded variables as constraints by inserting slack variables and therefore we obtain a large set of constraints.

This problem can be solved by using Bounded Variable Simplex method. The initial table is:

c_j	1	1	0	0	0	0	0	0	0	0		
Basis	x_{12}	x_{13}	x_{23}	x_{24}	x_{32}	x_{35}	x_{43}	x_{46}	x_{54}	x_{56}	X_B^*	c_B
x_{12}	1	0	-1	-1	1	0	0	0	0	0	0	1
x_{13}	0	1	1	0	-1	-1	1	0	0	0	0	1
x_{46}	0	0	0	-1	0	0	1	1	-1	0	0	0
x_{56}	0	0	0	0	0	-1	0	0	1	1	0	0
z_j	1	1	0	-1	0	-1	1	0	0	0		
z_j-c_j	0	0	0	-1	0	-1	1	0	0	0		
u_{ij}	16	13	10	12	4	14	9	20	7	4		

Iteration 1:

Here x_{35} is the entering variable, because the corresponding $z_j - c_j$ is negative. Now

$\theta_1 = \min\{\infty, \infty\}$ (corresponding to x_{12} and x_{46})
$= \infty$,

$\theta_2 = \min\left\{\frac{(0-13)}{(-1)}, \frac{(0-4)}{(-1)}\right\}$ (corresponding to x_{13} and x_{56})

$= 4$, (corresponding to x_{56})

and $u_{35} = 14$

$\therefore\ \theta = \min\{\theta_1, \theta_2, u_{35}\}$
$= \min\{\infty, 4, 14\} = 4\ (= \theta_2)$

Since $\theta = \theta_2$, x_{56} is substituted at its upper bound difference *i.e.,* $x_{56} = 4 - x'_{56}$ but it remains non-basic. Then we obtain the following table

c_j	1	1	0	0	0	0	0	0	0	0		
Basis	x_{12}	x_{13}	x_{23}	x_{24}	x_{32}	x_{35}	x_{43}	x_{46}	x_{54}		X_B^*	c_B
x_{12}	1	0	-1	-1	1	0	0	0	0	0	0	1
x_{13}	0	1	1	0	-1	-1	1	0	0	0	0	1
x_{46}	0	0	0	-1	0	0	1	1	-1	0	0	0
x'_{56}	0	0	0	0	0	-1	0	0	1	-1	-4	0

Now the entering variable x_{35} becomes basic and the leaving variable x'_{56} becomes non-basic at zero level, which yields:

c_j	1	1	0	0	0	0	0	0	0	0		
Basis	x_{12}	x_{13}	x_{23}	x_{24}	x_{32}	x_{35}	x_{43}	x_{46}	x_{54}	x'_{56}	X_B^*	c_B
x_{12}	1	0	-1	-1	1	0	0	0	0	0	0	1
x_{13}	0	1	1	0	-1	0	1	0	-1	1	4	1
x_{46}	0	0	0	-1	0	0	1	1	-1	0	0	0
x_{35}	0	0	0	0	0	1	0	0	-1	1	4	0
z_j	1	1	0	-1	0	0	1	0	-1	1		
z_j-c_j	0	0	0	-1	0	0	1	0	-1	1		
u_{ij}	16	13	10	12	4	14	9	20	7	4		

Iteration 2:

Here x_{54} is the entering variable, because the corresponding $z_j - c_j$ is negative. Now

$\theta_1 = \min \{\infty\} = \infty,$ (corresponding to x_{12})

$\theta_2 = \min \{\frac{(4-13)}{(-1)}, \frac{(0-20)}{(-1)}, \frac{(4-14)}{(-1)}\}$ (corresponding to x_{13}, x_{46} and x_{56})

$= 9,$ (corresponding to x_{13})

and $u_{54} = 7$

$\therefore\ \theta = \min\{\theta_1, \theta_2, u_{54}\} = \min\{\infty, 9, 7\} = 7\ (= u_{54})$

Because x_{54} enters at its upper bound, X_B^* remains unchanged and x_{54} becomes non-basic at its upper bound. By the substitution $x_{54} = 7 - x'_{54}$, we get

c_j	1	1	0	0	0	0	0	0	0	0		
Basis	x_{12}	x_{13}	x_{23}	x_{24}	x_{32}	x_{35}	x_{43}	x_{46}	x'_{54}	x'_{56}	X_B^*	c_B
x_{12}	1	0	-1	-1	1	0	0	0	0	0	0	1
x_{13}	0	1	1	0	-1	0	1	0	1	1	11	1
x_{46}	0	0	0	-1	0	0	1	1	1	0	7	0
x_{35}	0	0	0	0	0	1	0	0	1	1	11	0
z_j	1	1	0	-1	0	0	1	1	1	1		
z_j-c_j	0	0	0	-1	0	0	1	1	1	1		
u_{ij}	16	13	10	12	4	14	9	20	7	4		

Iteration 3:

Here x_{24} is the entering variable, because the corresponding $z_j - c_j$ is negative. Now

$$\theta_1 = \min \{\infty, \infty\} \qquad \text{(corresponding to } x_{13}, x_{35}\text{)}$$
$$= \infty,$$
$$\theta_2 = \min \left\{\frac{(0-16)}{(-1)}, \frac{(7-20)}{(-1)}\right\} \qquad \text{(corresponding to } x_{12} \text{and } x_{46}\text{)}$$
$$= \min \{16, 13\}$$
$$= 13, \qquad \text{(corresponding to } x_{46}\text{)}$$

and $u_{24} = 12$

$\therefore\ \theta = \min \{\theta_1, \theta_2, u_{24}\}$
$= \min \{\infty, 13, 12\}$
$= 12\ (= u_{24})$

Because x_{24} enters at its upper bound, X_B^* remains unchanged and x_{24} becomes non-basic at its upper bound. By the substitution $x_{24} = 12 - x'_{24}$, we get

c_j	1	1	0	0	0	0	0	0	0	0		
Basis	x_{12}	x_{13}	x_{23}	x'_{24}	x_{32}	x_{35}	x_{43}	x_{46}	x'_{54}	x'_{56}	X_B^*	c_B
x_{12}	1	0	-1	1	1	0	0	0	0	0	12	1
x_{13}	0	1	1	0	-1	0	1	0	1	1	11	1
x_{46}	0	0	0	1	0	0	1	1	1	0	19	0
x_{35}	0	0	0	0	0	1	0	0	1	1	11	0
z_j	1	1	0	1	0	0	1	1	1	1		
z_j-c_j	0	0	0	1	0	0	1	1	1	1		
u_{ij}	16	13	10	12	4	14	9	20	7	4		

The last table is feasible and optimal. The optimal values are obtained by back substitution.

$x_{12} = 12,\quad x_{13} = 11,\quad x_{35} = 11,\quad x_{46} = 19,\quad x_{23} = 0,\quad x_{32} = 0,\quad x_{43} = 0,$

$x'_{24} = 0$ gives $x_{24} = 12 - x'_{24} = 20 - 0 = 20;$

$x'_{54} = 0$ gives $x_{54} = 7 - x'_{54} = 7 - 0 = 7;$

$x'_{56} = 0$ gives $x_{56} = 4 - x'_{56} = 4 - 0 = 4.$

Therefore, the optimal solution is

$x_{12} = 12,\quad x_{13} = 11,\quad x_{35} = 11,\quad x_{46} = 19,\quad x_{23} = 0,\quad x_{32} = 0,\quad x_{43} = 0,$
$x_{24} = 20,\quad x_{54} = 7,\quad x_{56} = 4.$

and the associated maximum amount of flow is

$$z = x_{12} + x_{13} = 12 + 11 = 23.$$

9. Conclusion

We have provided a new algorithm for finding the maximum amount of flow from source to sink in a flow network. The proposed algorithm returns a maximum flow and to calculate the maximum flow this algorithm takes less number of iterations and less augmentation. A numerical example is solved to illustrate the proposed algorithm. By using bounded variable simplex method we have also solved the flow network problem, which is very easy than simplex method because it reduces a set of large number of constraints into a small one.

REFERENCES

1. D.R.Fulkerson and G.B. Dantzig, Computation of maximum flow in network, Naval Research Logistics Quarterly, 2 (1955) 277-283.
2. Kanti Swarup, P.K. Gupta, Mon Mohan, Operations Research, Sultan Chand and Sons, 14th Edition, 2008.
3. L.R.Ford and D.R.Fulkerson, Maximal Flow through a Network, Canadian Journal of Mathematics, (1956)399-404.
4. J.Edmonds and R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, Journal of the ACM, 19(2) (1972) 248-264.
5. A.V. Goldberg, E.Tardos and R.E.Tarjan, A new approach to the maximum-flow problem, Journal of the ACM, 35(1988) 921-940.
6. R.K.Ahuja, James B. Orlin, A fast and simple algorithm for the maximum flow problem, Operations Research, 35(5) (1989) 748-759.
7. Dorit S. Hochbaum, The Pseudo-flow algorithm: A new algorithm for the maximum flow problem, Operations Research, 56(4) (2008) 992-1009.
8. Chintan Jain, Deepak Garg, Improved Edmond-Karps algorithm foe network flow problem, International Journal of Computer Applications, 37(1) (2012) 48-53.
9. H.A.Taha, Operation Research- An Introduction, Prentice Hall, 7th Edition, 2007.
10. Bazarra, M. and J. Jarvis, Linear Programming and Network Flows, John Wiley & Sons, 1977.
11. 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.