Proc. Indian Acad. Sci. (Math. Sci.), Vol. 92, No. 3, December 1983, pp. 171–190.

A maximum flow problem with intermediate node requirements

S N NARAHARI PANDIT and B KRISHNA REDDY

Department of Statistics, Osmania University, Hyderabad 500007, India

MS received 17 August 1982

Abstract. The conventional Maximum flow problem is modified to take account of possible requirements at intermediate nodes across which flow takes place. This is achieved by incorporating pseudo or priority arcs to act as thresholds controlling out-flow from the nodes and modifying the Ford and Fulkerson algorithm to take account of these thresholds.

Effect of introducing these threshold–requirements at intermediate nodes on the final flow into the sink in the network is examined by some numerical examples.

Keywords. Maximum flow; intermediate node; threshold requirements; pseudo or priority arcs; conservation of flows.

1. Introduction

The Network flow optimization problem considered here is that of determining a maximal steady state flow from a node designated as "the source" to a node designated as "the sink" in a network under the following situation:

(a) Arc capacities are non-negative,

(b) Intermediate nodes have finite requirements.

However, the requirement at a node without a positive in-flow need not be satisfied while out-flow from a node with positive in-flow is allowed only if in-flow into the node exceeds its requirement.

2. Some definitions and notations

A Network $N(V; \mathscr{A} \subseteq (V \times V); C)$ is a topological structure having a non-empty set V of elements called 'nodes' (vertices, points) and a set $\mathscr{A}$ of elements called 'arcs' (lines, branches, edges). The set $\mathscr{A}$ is a proper sub-set of elemets from the set $(V \times V)$, i.e., elements of $\mathscr{A}$ are ordered pairs of elements from V and are called 'directed arcs'. Every arc in the network has a positive number associated with it; this number is called the capacity of the arc.

Nodes are represented by small circles with the corresponding node names inside this. Nodes are named as $1, 2, \ldots, n$. Node 1 is the source node and node n is the sink node.

$$V = \{v : v = 1, 2, \ldots, n\} \text{ is the set of nodes.}$$

$$\mathscr{A} = \{a_p : p = 1, 2, \ldots, L;\ a_p = (i, j),\ i, j \in V \text{ and } i \neq j\}$$

is the set of ordered pairs of elements from V forming the set of arcs. L is a positive

integer not more than $(n^2 - n)$.

$K = \{C_p\colon\ p = 1, 2, \ldots, L\}$ is the set of capacities of the arcs a_p, $p = 1, 2, \ldots, L$ respectively.

The intermediate nodes j for $j = 2, 3, \ldots, (n-1)$ have the requirements $R_j > 0$. For convenience, we define $R_1 = R_n = 0$.

It is more convenient to give the capacities of the arcs (i, j) by the corresponding matrix entries C_{ij} of an $(n \times n)$ matrix; for "non-existing arcs" i.e., ordered pairs $(i, j) \notin \mathscr{A}$, C_{ij} is defined as zero. Similarly, $A = (a_{ij})$ is defined as the connectivity matrix of the network, where

$$a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \mathscr{A} \\ 0 & \text{if } (i,j) \notin \mathscr{A} \end{cases}$$

3. Maximum flow problem (MFP for short)

The conventional maximum flow problem (MFP), (Ford and Fulkerson [1], Hu [2]) can be stated as follows:

Find real numbers $\{f_{ij}\}$, such that

$$0 \leqslant f_{ij} \leqslant C_{ij}; \tag{1}$$

$$\sum_{i=1}^{n} f_{ij} - \sum_{k=1}^{n} f_{jk} = 0 \quad \text{if} \quad j \neq 1 \text{ and } n \tag{2}$$

$$\sum_{i=1}^{n} f_{ij} - \sum_{k=1}^{n} f_{jk} = T \quad \text{if} \quad j = n; \tag{3}$$

$$\sum_{i=1}^{n} f_{ij} - \sum_{k=1}^{n} f_{jk} = -T \quad \text{if} \quad j = 1 \tag{4}$$

which also maximise $T = \sum_{i=1}^{n} f_{in}$

Equation (2) is called 'Law of Conservation of Flow' or 'Material balance equation' or 'Kirchhoff's Law for Flows', and indicates that 'No material is lost or added during transit'. In the present problem, (2) is relaxed as follows:

Every intermediate node j, with a positive inflow will have a positive outflow, only after meeting its requirement. However, the requirement of a node without positive inflow is ignored.

4. Maximum flow problem with requirements (MFPR)

Find a set of real numbers $\{f_{ij}\}$ to maximise $T_1 = \sum_{i=1}^{n} f_{in}$ such that

$$0 \leqslant f_{ij} \leqslant C_{ij} \tag{5}$$

Let $I_j = \sum_{i=1}^{n} f_{ij}$ be the inflow for j, then

If $\quad I_j \leqslant R_j, \quad f_{jk} = 0$ for all k while, (6)

$$\text{If} \qquad I_j > R_j, \quad \text{then } I_j = R_j + \sum_{k=1}^{n} f_{jk} \tag{7}$$

In what follows an algorithm for solving MFPR is presented. The algorithm is got by suitably modifying the algorithm due to Ford and Fulkerson for MFP.

5. Modified algorithm for MFPR

(i) Read n; $C_{ij}((i = 1, 2, \ldots, n), (j = 1, 2, \ldots, n))$; $R_i(i = 1, 2, \ldots, n)$
(ii) Do step 3 varying i from 1 to n
(iii) $\mathrm{ST}_i \leftarrow 0$; $\mathrm{FL}_i \leftarrow 0$; $\mathrm{AF}_i \leftarrow 0$; $\mathrm{NOD}_i \leftarrow 0$
(iv) Code $\leftarrow 0$; $\mathrm{NOD}_1 \leftarrow n+1$
(v) Do step 6 varying i from 1 to n
(vi) If $C_{1i} \neq 0$; $\mathrm{NOD}_i \leftarrow 1$; $\mathrm{ST}_i \leftarrow 1$; Code $\leftarrow 1$ else go to 5
(vii) If Code $= 0$ go to 36
(viii) If $\mathrm{NOD}_n \neq 0$ go to 20
(ix) $K \leftarrow 1$
(x) Code $\leftarrow 0$
(xi) Do step 12 through 15 varying i from 1 to n
(xii) If $(\mathrm{ST}_i - K) \neq 0$ go to 11
(xiii) Do step 14 through 15 varying j from 1 to n
(xiv) If $C_{ij} \neq 0$; code $\leftarrow 1$ else go to 13
(xv) If $\mathrm{NOD}_j = 0$; $\mathrm{NOD}_j \leftarrow 1$; $\mathrm{ST}_j \leftarrow (K+1)$ else go to 13
(xvi) If Code $= 0$ go to 36
(xvii) If $\mathrm{NOD}_n \neq 0$ go to 23
(xviii) $K \leftarrow K+1$
(xix) If $K \geqslant n$ go to 36 else go to 10
(xx) $C_{n1} = C_{1n}$; $C_{1n} = 0$
(xxi) Print n, C_{n1} (Prints out the direct path and amount of flow between 1 and n)
(xxii) Go to 2
(xxiii) $K \leftarrow K+1$; $M \leftarrow K+1$
(xxiv) $\mathrm{IJ}_M \leftarrow n$
(xxv) Do step 26 varying i from K to 1
(xxvi) $L1 \leftarrow \mathrm{IJ}_{i+1}$; $\mathrm{IJ}_i \leftarrow \mathrm{NOD}_{L1}$
(xxvii) $N2 \leftarrow K+1$
(xxviii) Do step 29 varying i from 1 to K
(xxix) $\mathrm{FL}_i \leftarrow C_{\mathrm{IJ}_i \mathrm{IJ}_{j+1}}$; $\mathrm{IJR}_i \leftarrow R_{\mathrm{IJ}_i}$
(xxx) Calculate flow and store the flow in AFL and store the remaining requirements to be met in IJR (procedure is given below)
(xxxi) Do step 32 through 33 varying i from 1 to $(N2-1)$
(xxxii) $\mathrm{I2} \leftarrow \mathrm{IJ}_i$; $\mathrm{I3} \leftarrow \mathrm{IJ}_{i+1}$
(xxxiii) $C_{I2\,I3} \leftarrow C_{I2\,I3} - \mathrm{AFL}_i$; $C_{I3\,I2} \leftarrow C_{I3\,I2} + \mathrm{AFL}_i$; $R_{I2} \leftarrow \mathrm{IJR}_i$
(xxxiv) Print IJ; FL; AFL
(xxxv) Go to 2
(xxxvi) Print "Path search is over"
(xxxvii) End

6. Calculation of AFL and IJR

To calculate AFL and IJR the following procedure is adopted. AFL consists of flow values in different arcs included in the path. FL contains the remaining capacities of the arcs in the path. $\{F_{jn}\}$ contains the flow values in the "Priority arcs" for nodes 2 to $(n-1)$.

(i) Do step 2 through 7 varying i from 1 to $(n-2)$
(ii) $K \leftarrow$ Minimum of $\{FL_1, FL_2, \ldots, FL_i, IJR_{i+1}\}$
(iii) Do step 4 through 5 varying j from 1 to i
(iv) $FL_j \leftarrow FL_j - K$
(v) $AFL_j \leftarrow AFL_j + K$
(vi) $IJR_{i+1} \leftarrow IJR_{i+1} - K$
(vii) $F_{(i+1)n} \leftarrow F_{(i+1)n} + K$
(viii) $K1 \leftarrow$ minimum of $\{FL_1, \ldots, FL_{n-1}\}$
(ix) Do step 10 through 11 varying i from 1 to $(n-1)$
(x) $FL_i \leftarrow FL_i - K1$
(xi) $AFL_i \leftarrow AFL_i + K1$
(xii) Return

7. Numerical illustration

A numerical example illustrating the working of the algorithm for the MFPR follows:

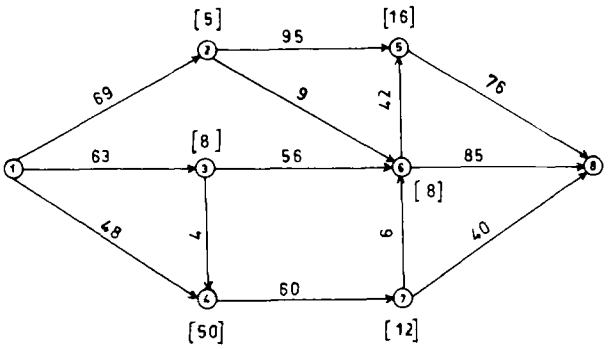


Figure 1.

Figure 1 is a schematic repesentation of a network with capacitated arcs and node requirements. For this network we have,

$$V = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

$$\mathscr{A} = \{(1, 2), (1, 3), (1, 4), (2, 5), (2, 6), (3, 4), (3, 6), (4, 7), (5, 8), (6, 5), (6, 8), (7, 6), (7, 8)\}$$

$$K = \{69, 63, 48, 95, 9, 4, 56, 60, 76, 42, 85, 6, 40\}$$

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}; \quad C = \begin{bmatrix} 0 & 69 & 63 & 48 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 95 & 9 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 56 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 60 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 76 \\ 0 & 0 & 0 & 0 & 42 & 0 & 0 & 85 \\ 0 & 0 & 0 & 0 & 0 & 6 & 0 & 40 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$R = \{0, 5, 8, 50, 16, 8, 12, 0\}$

Numbers present on the arcs are "Capacities". Numbers besides the nodes (represented by number inside the circle) in square brackets are requirements of the nodes.

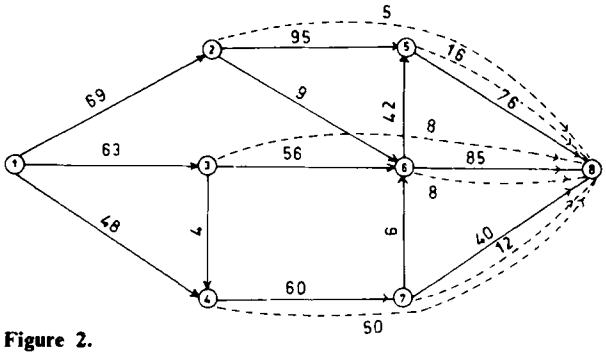


Figure 2.

Figure 2 is obtained from figure 1 by suitable modifications as per the algorithm. Pseudo arcs i.e., Priority arcs are represented by dotted lines. Numbers on the priority arcs are requirements of the corresponding intermediate nodes. The flow matrix (f_{ij}) is augmented by a column (F_{jn}) of the flows corresponding to the flows in priority arcs.

Requirement R_j of node j is sent directly to the sink through a pseudo arc called a 'Priority arc'; capacity of this priority arc is equal to R_j. This arc should be satisfied fully before a positive outflow from node j can occur. In other words, inflow into node j must be greater than R_j in order to have a positive outflow from node j, i.e., $\sum_{i=1}^{n} f_{ij} > R_j$ if, in a flow pattern (f_{ij}), $\sum_{k=1}^{n} f_{jk} > 0$. The flow in priority arc $F_{jn} = \min\left(\sum_{i=1}^{n} f_{ij}, R_j\right)$. The flows (f_{jk}) can be positive only if $F_{jn} = R_j$.

From figure 2 it is clear that the pseudo arcs named priority arcs are different from 'Multiple arcs', where multiple arcs are defined as 'More than one arc exists between the given pair of nodes (i.e., parallel arcs)'.

The vectors {NOD} and {ST} are used to identify a path from source to sink during the different stages of executing the algorithm. At a given stage, if $C_{ij} > 0$, then NOD_j is modified to the values i, ST_j is modified to the value $(\text{ST}_i + 1)$, indicating the step number in which node j is reached. Code is a check counter which takes values 0 or 1 to check the existence of a path.

The capacity matrix C given in §7 is augmented by a row R of requirements and a

column F for 'Flow in priority arcs'. Initially F is set to zero. Augmentations are shown in table 1. Now the iteration starts.

ITERATION 1

Table 1.

		1	2	3	4	5	6	7	8	F	NOD	ST
			(−)									
	1	0	69	63	48	0	0	0	0	0	9	0
		(+)				(−)				(+)		
	2	0	0	0	0	95	9	0	0	0	1	1
	3	0	0	0	4	0	56	0	0	0	1	1
	4	0	0	0	0	0	0	60	0	0	1	1
C =			(+)						(−)	(+)		
	5	0	0	0	0	0	0	0	76	0	2	2
	6	0	0	0	0	42	0	0	85	0	2	2
	7	0	0	0	0	0	6	0	40	0	4	2
						(+)						
	8	0	0	0	0	0	0	0	0	0	5	3
	R	0	(−) 5	8	50	(−) 16	8	12	0			

Initially $\{NOD_i\}$ and $\{ST_i\}$ are set to 'zeroes' for all i. The vectors $\{NOD\}$ and $\{ST\}$ are modified as per the algorithm.

$$\text{Code} = 0;\ NOD_1 = 9$$

$$C_{12} \neq 0;\ NOD_2 = 1;\ ST_2 = 1;\ \text{Code} = 1$$

$$C_{13} \neq 0;\ NOD_3 = 1;\ ST_3 = 1;\ \text{Code} = 1$$

$$C_{14} \neq 0;\ NOD_4 = 1;\ ST_4 = 1;\ \text{Code} = 1$$

Since Code = 1 and $NOD_n = 0$; $K = 1$; Code = 0 $(ST_2 - K) = 0$. For a positive C_{ij} in matrix C with $i = 2$,

$$C_{25} \neq 0;\ NOD_5 = 2;\ ST_5 = 2;\ \text{Code} = 1$$

$$C_{26} \neq 0;\ NOD_6 = 2;\ ST_6 = 2;\ \text{Code} = 1$$

Proceeding in the same way, vectors $\{NOD\}$ and $\{ST\}$ are modified. Results are as shown in table 1.

To identify path, we proceed in vector $\{NOD\}$. $NOD_8 = 5$ with $ST_8 = 3$. Hence node 8 is reached directly from node 5 and is reached in 3 steps from the source.

$NOD_5 = 2$ with $ST_5 = 2$. Hence node 5 is reached directly from node 2 and is reached from source in 2 steps.

$NOD_2 = 1$ with $ST_2 = 1$. Hence node 2 is reached directly from node 1 and is reached from source in 1 step.

Thus we have the path $1 \rightarrow 2 \rightarrow 5 \rightarrow 8$.

Now, we construct table 2 below, where the serial number can go up to 8.

Table 2.

Sl. No.	1	2	3	4
Path	1	2	5	8
FL	69	95	76	
IJR	0	5	16	0

FL_i = Capacity of the arc $(i, i+1)$ in the path.

IJR_i = Requirement of the node i in the path.

Table 3.

FL	0	31	28
IJR	0	0	0
AFL	69	64	48
F	0	5	16

Table 3: constructed from table 2 according to the §6 of the algorithm.

Super-impose the values from table 3 to table 1 such that the corresponding values of AFL are super-imposed for a + ve sign and FL values are super-imposed for a − ve sign of the matrix '*C*'. Similarly, IJR is super-imposed into {R} for a − ve sign and *F* is super-imposed into {F} for a + ve sign. By super-imposition we mean that the corresponding values are added for a + ve sign and substracted for a − ve sign to the existing values. The resultant matrix is shown in table 4. Now, the second iteration starts.

ITERATION 2

Table 4.

		1	2	3	4	5	6	7	8	*F*	NOD	ST
				(−)								
	1	0	0	63	48	0	0	0	0	0	9	0
	2	69	0	0	0	31	9	0	0	5	0	0
		(+)					(−)			(+)		
C =	3	0	0	0	4	0	56	0	0	0	1	1
	4	0	0	0	0	0	0	60	0	0	1	1
	5	0	64	0	0	0	0	0	28	16	6	3
				(+)					(−)	(+)		
	6	0	0	0	0	42	0	0	85	0	3	2
	7	0	0	0	0	0	6	0	40	0	4	2
							(+)					
	8	0	0	0	0	48	0	0	0	0	6	3
				(−)			(−)					
	R	0	0	8	50	0	8	12	0			

We have the path $1 \rightarrow 3 \rightarrow 6 \rightarrow 8$.

Now, we construct the table 5 according to the path and the corresponding capacities alongwith requirements.

Table 5.

Sl. No.	1	2	3	4
Path	1	3	6	8
FL	63	56	85	
IJR	0	8	8	0

Table 6 is obtained from table 5 as per "Calculation of AFL and IJR" of the algorithm.

Table 6.

FL	0	1	38
IJR	0	0	0
AFL	63	55	47
F	0	8	8

The values of table 6 are once again super-imposed on table 4 as explained in iteration 1. Remaining capacities and requirements after the necessary changes are given in table 7. Now, we start with iteration 3.

ITERATION 3

Table 7.

$C =$

	1	2	3	4	5	6	7	8	*F*	NOD	ST
				(−)							
1	0	0	0	48	0	0	0	0	0	9	0
2	69	0	0	0	31	9	0	0	5	0	0
3	63	0	0	4	0	1	0	0	8	0	0
	(+)						(−)		(+)		
4	0	0	0	0	0	0	60	0	0	1	1
5	0	64	0	0	0	0	0	28	16	0	0
6	0	0	55	0	42	0	0	38	8	7	3
				(+)				(−)	(+)		
7	0	0	0	0	0	6	0	40	0	4	2
							(+)				
8	0	0	0	0	48	47	0	0	0	7	3
				(−)			(−)				
R	0	0	0	50	0	0	12	0			

We have the path $1 \rightarrow 4 \rightarrow 7 \rightarrow 8$.

Tables 8 and 9 are obtained as per the algorithm already explained in iterations 1 and 2.

Table 8.

Sl. No.	1	2	3	4
Path	1	4	7	8
FL	48	60	40	
IJR	0	50	12	0

Table 9.

FL	0	60	40
IJR	0	2	12
AFL	48	0	0
F	0	48	0

Similar to the earlier operations of superimpositions, the necessary changes of table 7 are made to obtain table 10 and we proceed further to iteration 4.

ITERATION 4

Table 10.

$C =$

	1	2	3	4	5	6	7	8	*F*	NOD	ST
1	0	0	0	0	0	0	0	0	0	9	0
2	69	0	0	0	31	9	0	0	5	0	0
3	63	0	0	4	0	1	0	0	8	0	0
4	48	0	0	0	0	0	60	0	48	0	0
5	0	64	0	0	0	0	0	28	16	0	0
6	0	0	55	0	42	0	0	38	8	0	0
7	0	0	0	0	0	6	0	40	0	0	0
8	0	0	0	0	48	47	0	0	0	0	0
R	0	0	0	2	0	0	12	0			

From table 10, we find that no feasible path from source (i.e., node 1) to sink (i.e. node 8) exists. Hence, the algorithm is terminated.

It is worth noting the following points of contrast with the usual MFP.

(i) Inflow into node 4, i.e., $\sum_{i=1}^{8} f_{i4} = 48$ is not greater than its requirement $R_4 = 50$. Hence, no outflow takes place from node 4 i.e., $f_{4j} = 0$ for all j.

(ii) Total flow into the sink $= T_1 = 48 + 47 = 95$. However, total outflow from the source is $69 + 63 + 48 = 180 > 95$.

(iii) Requirement of the node 4 is satisfied partially whereas that of node 7 is not met at all. However, flow-out from these nodes is zero.

Table 11.

Capacity matrix	A				B				C				D			
	a	*b*	*c*	*d*	*a*	*b*	*c*	*d*	*a*	*b*	*c*	*d*	*a*	*b*	*c*	*d*
1.	178	178	0	0(4)	205	170	35	35(5)	205	170	35	35(6)	205	178	27	35(7)
2.	158	158	0	0(12)	158	133	25	25(13)	158	133	25	25(14)	158	150	8	25(15)
3.	158	158	0	0(16)	158	97	61	61(17)	158	97	61	61(18)	158	151	7	61(19)
4.	158	158	0	0(20)	158	137	21	70(21)	158	123	35	70(22)	158	138	20	70(23)
5.	158	158	0	0(24)	158	117	41	70(25)	158	103	55	70(26)	158	155	3	70(27)
6.	178	178	0	0(28)	225	160	65	65(29)	225	160	65	65(30)	225	178	47	65(31)
7.	178	178	0	0(32)	241	70	171	212(33)	241	70	171	212(34)	241	125	116	212(35)
8.	114	114	0	0(36)	121	102	19	49(37)	121	102	19	49(38)	121	114	7	49(39)
9.	133	133	0	0(40)	152	121	31	31(41)	152	121	31	31(42)	152	133	19	31(43)
10.	157	157	0	0(44)	179	148	31	31(45)	179	148	31	31(46)	179	157	22	31(47)
11.	150	150	0	0(48)	152	121	31	31(49)	152	121	31	31(50)	152	145	7	31(51)
12.	104	104	0	0(52)	104	73	31	31(53)	104	73	31	31(54)	104	97	7	31(55)

A: Solution by conventional MFP; *B*: Solution by MFPR [i.e., as per case (i)]; *C*: Solution as per case (ii); *D*: Solution as per case (iii).

a: Total out-flow from the source ($\sum_{i=1}^{n} f_{1i}$); *b*: Total in-flow into the sink ($\sum_{i=1}^{n} f_{in}$); *c*: Total requirement satisfied ($\sum_{i=1}^{n} r_i$); *d*: Total requirement ($\sum_{i=1}^{n} R_i$).

*Figure in bracket indicates the corresponding figure numbers.

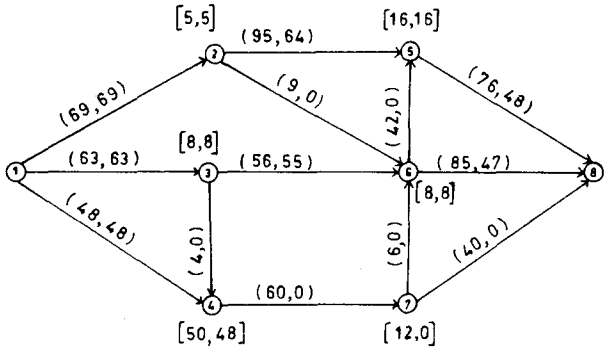


Figure 3.

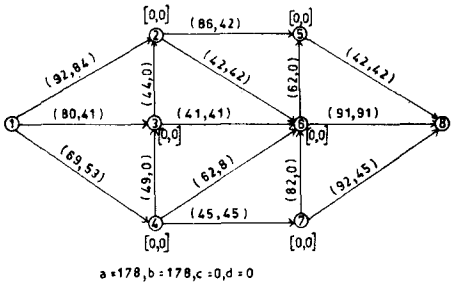


Figure 4.

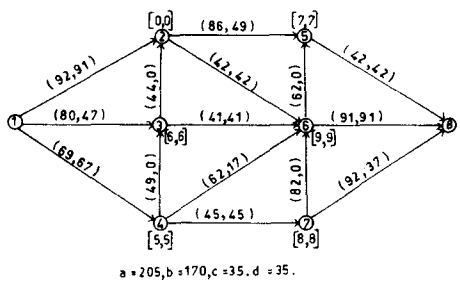


Figure 5.

Figure 3 gives the final picture of the flow values in arcs and the requirements. The numbers besides the arcs are (capacity, flow) and besides the nodes are [Requirement, Requirement satisfied] respectively.

With all the intermediate node requirements zero, we have the conventional MFP, for which we get a maximum flow (MF). By introducing the threshold requirements, we have MFPR and we get MF by the algorithm in §§ 5 and 6.

In general, the MF in MFP is not same as the MF in MFPR. However, introduction of the requirements may or may not necessarily make the situation worse. Thus, for example, we can see from figure 4, with requirements vector 'zero', the MF is 178 units whereas from figure 5, we observe that the MF is 170 units only, with the requirements vector $R = [0, 0, 6, 5, 7, 9, 8, 0]$. Here, we have the total requirement 35 units, satisfied completely. From figure 8, it is also seen that by selecting the requirements vector properly, we can have the MF equal to 178 units same as the MF in figure 4 and moreover, we can satisfy more of the requirements than in figure 5. In this network, we can satisfy 63 units of the requirement from the requirement vector $R = [0, 8, 39, 7, 0, 9, 0, 0]$.

A trivial way of introducing the requirements vecor without upsetting the MF is to have the requirements only at nodes directly connected to the source, their values being the residual capacities, $(C_{1j} - f_{1j})$ of the corresponding arcs. However, this simplicity disappears when requirement is introduced at a node not directly connected with the source.

This problem of determining a 'satisfactory requirements vector' is of considerable interest, particularly in the context analogous to 'Goal Programming' situation, where, in addition to achieving a primary objective, secondary objectives are also considered for achievement.

Thus, for instance, in planning for transport of a material through conduits, basic interest may be in getting the material through a network whose arc capacities may first be fixed on techno-economic considerations pertaining to the overall network system [even otherwise, already a network with given arc capacities may exist] and it may be required to see how far the same network may be made to serve a secondary function of transporting the materials also to the different intermediate nodes, so that new consumption units of appropriate capacities could be installed without detriment to the receipt at the sink.

The situation is somewhat similar to the 'Parametric linear programming', though slightly harder. This problem may be stated as follows:

Find real numbers $\{f_{ij}\}$, to maximise $T_2 = V + \sum_{j=1}^{n} r_j$, where $V = \sum_{j=1}^{n} f_{jn}$, such that

$$0 \leqslant f_{ij} \leqslant C_{ij} \tag{8}$$

$$0 \leqslant r_j \leqslant R_j; \tag{9}$$

and

$$\sum_{i=1}^{n} f_{ij} = r_j + \sum_{k=1}^{n} f_{jk} \tag{10}$$

where $f_{jk} = 0$, $k = 1$ to n; if $r_j \leqslant R_j$ for $j = 2$ to $(n-1)$.

This problem is being investigated in depth and will be reported later. However, to get a feel of the situation, we give below some networks for which different requirements and their effect on the flow to the destination are worked out. The table 11 gives the results of different networks solved on an IRIS-55 computer in FORTRAN-IV language.

Twelve different capacity matrices with different network structures are considered. For each of these capacity matrices two alternatives are considered; (i) with zero requirement vector and (ii) with a specified non-zero requirement vector. Under the second category, three further alternatives are considered as in cases (i), (ii) and (iii) described in the next paragraph.

CASE (i)

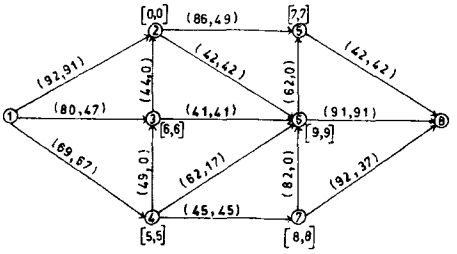
The given MFPR is solved as per the algorithm given in §§ 5 and 6.

CASE (ii)

The MFPR is modified by introducing a pseudo-node $(n+1)$ called the 'super sink', which is connected to (i) node n by a directed arc $(n, n+1)$ with capacity ∞, (ii) the intermediate node j by directed arc $(j, n+1)$ with capacity R_j for $j = 2$ to $(n-1)$.

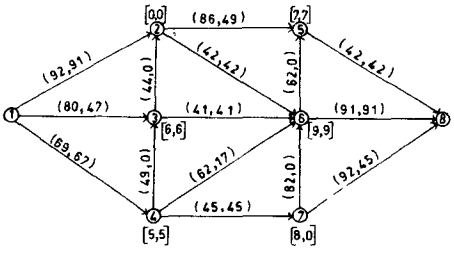
CASE (iii)

The MFPR is modified by introducing two pseudo nodes $(n+1)$ and $(n+2)$ which are connected in the following way: (i) node $(n+2)$ is connected to nodes n and $(n+1)$ by directed arcs $(n, n+2)$ and $(n+1, n+2)$ with capacities equal to ∞; (ii) node $(n+1)$



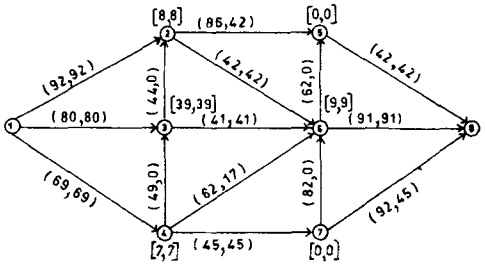
a =205, b =170, c =35, d =35.

Figure 6.



a = 205, b = 178, c = 27, d = 35.

Figure 7.



a = 241, b = 178, c = 63, d = 63.

Figure 8.

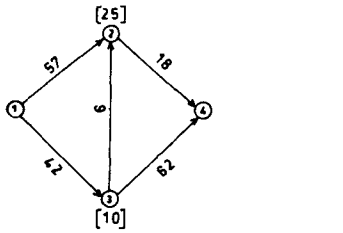


Figure 9.

Figure 10.

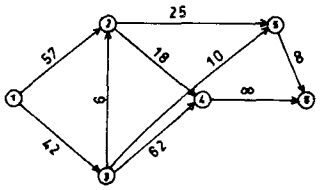


Figure 11.

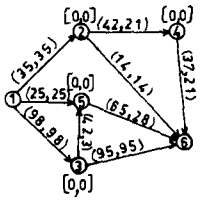
directly connected to an intermediate node j through a directed arc $(j, n+1)$ with capacity R_j for all $j = 2$ to $(n-1)$.

By these modifications cases (ii) and (iii) are converted to equivalent MFP which can be solved by the conventional Ford and Fulkerson algorithm and the solutions can easily be reinterpreted in the context of the cases (ii) and (iii) respectively.

From table 11, it is interesting to note that in most of the cases, we are getting more flow into the sink in case (iii) than in cases (i) and (ii) and the requirements are satisfied less in total amount in case (iii) for the same capacity matrix and the requirements vector than the other two cases.

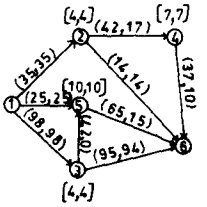
In case (ii) more of a total requirement is satisfied than in case (i) and in no single case the requirement satisfied in case (ii) is less than that of case (i) for the same capacity matrix and the requirements vector, which is not an unexpected result.

The MFPR in case (i) shown in figure 9 will look like as shown in figures 10 and 11 for cases (ii) and (iii) respectively after the modifications. However, in the figures that follow (figures 12–55) later these modifications are not shown because of complications and overlappings of the lines. But the problems are solved as per the modifications only and



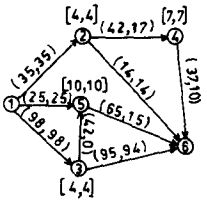
a =158, b =158, c = 0, d = 0.

Figure 12.



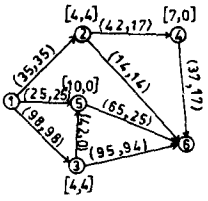
a =158, b =133, c =25, d =25.

Figure 13.



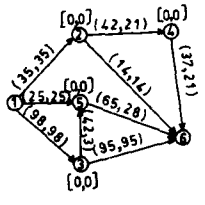
a =158, b =133, c =25, d =25.

Figure 14.



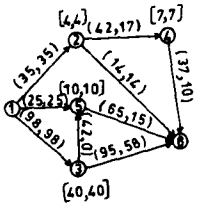
a =158, b =150, c =8, d =25.

Figure 15.



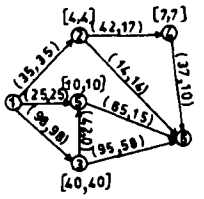
a =158, b =158, c = 0, d = 0.

Figure 16.



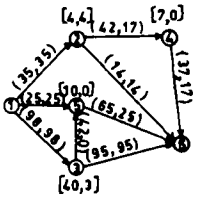
a =158, b =97, c =61, d =61.

Figure 17.



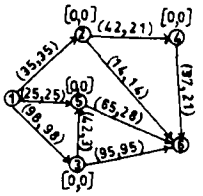
a = 158, b = 97, c = 61, d = 61.

Figure 18.



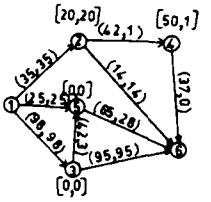
a = 158, b = 151, c = 7, d = 61.

Figure 19.



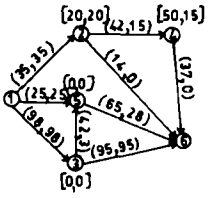
a = 158, b = 158, c = 0, d = 0.

Figure 20.



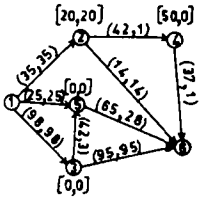
a = 158, b = 137, c = 21, d = 70.

Figure 21.



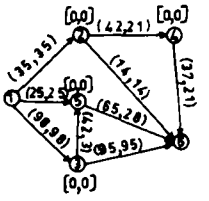
a = 158, b = 123, c = 35, d = 70.

Figure 22.



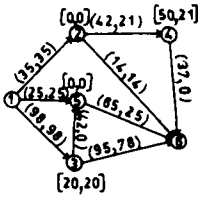
a = 158, b = 138, c = 20, d = 70.

Figure 23.



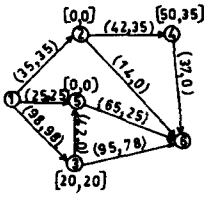
a = 158, b = 158, c = 0, d = 0.

Figure 24.



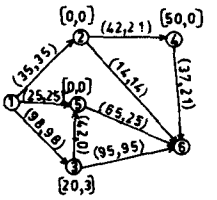
a = 158, b = 117, c = 41, d = 70.

Figure 25.



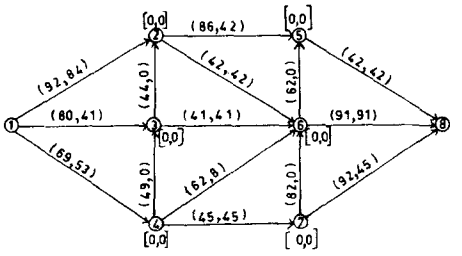
a = 158, b = 103, c = 55, d = 70.

Figure 26.



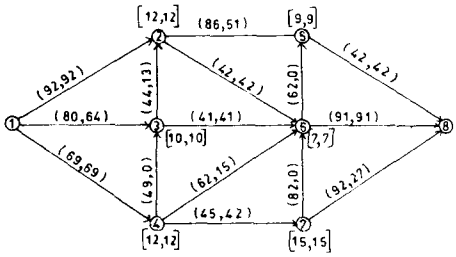
a = 158, b = 155, c = 3, d = 70

Figure 27.



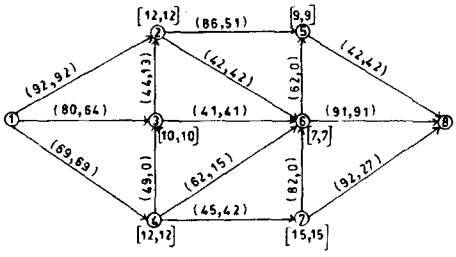
a = 178, b = 178, c = 0, d = 0.

Figure 28.



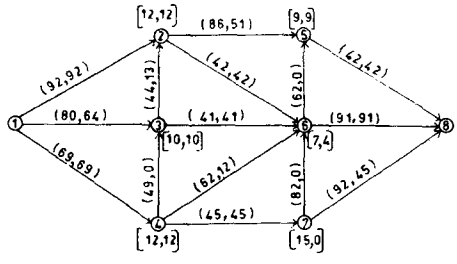
a = 225, b = 160, c = 65, d = 65

Figure 29.



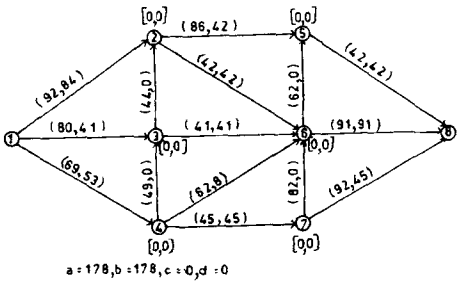
a = 225, b = 160, c = 65, d = 65.

Figure 30.



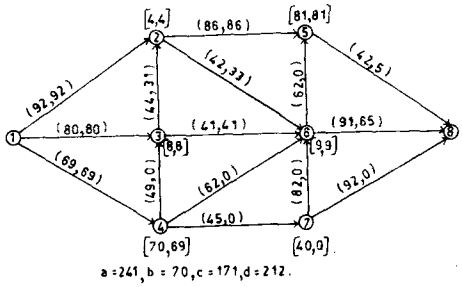
a = 225, b = 178, c = 47, d = 65.

Figure 31.



a = 178, b = 178, c = 0, d = 0

Figure 32.



a = 241, b = 70, c = 171, d = 212.

Figure 33.

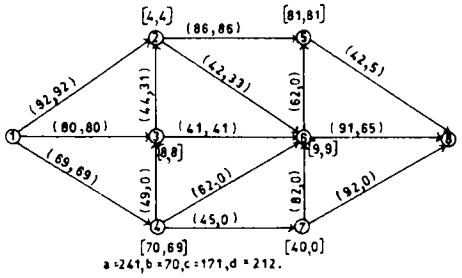


Figure 34.

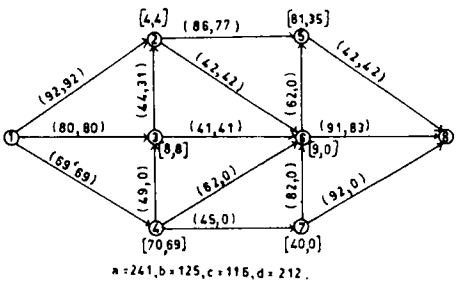
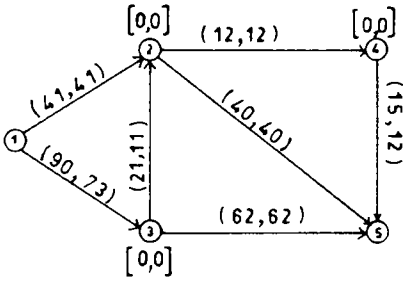
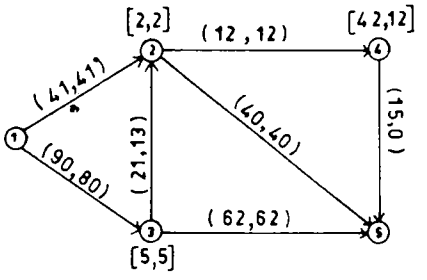


Figure 35.



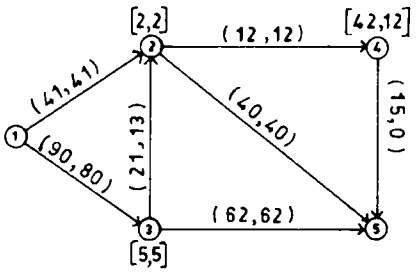
a = 114, b = 114, c = 0, d = 0.

Figure 36.



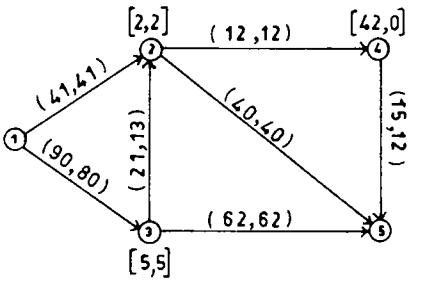
a =121, b =102, c =19, d =49.

Figure 37.



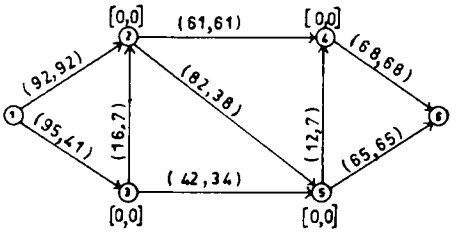
a = 121, b = 102, c = 19, d = 49.

Figure 38.



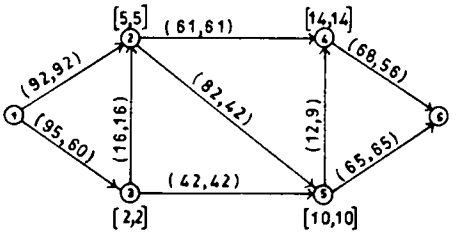
a =121, b =114, c = 7, d = 49.

Figure 39.



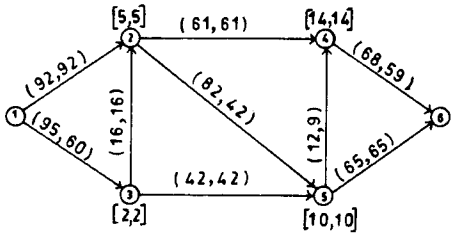
a=133, b =133, c = 0, d = 0.

Figure 40.



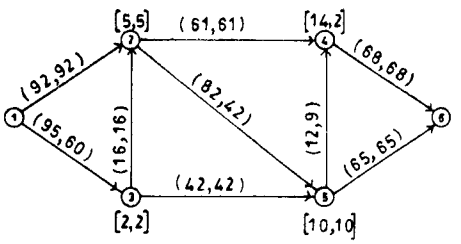
a = 152, b = 121, c = 31, d = 31.

Figure 41.



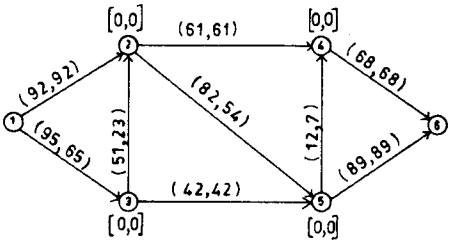
a = 152, b = 121, c = 31, d = 31

Figure 42.



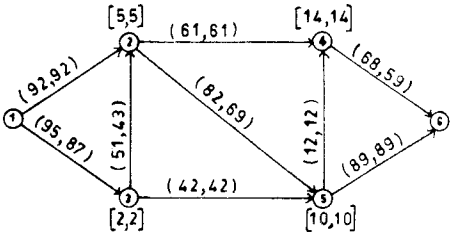
a = 152, b = 133, c = 19, d = 31

Figure 43.



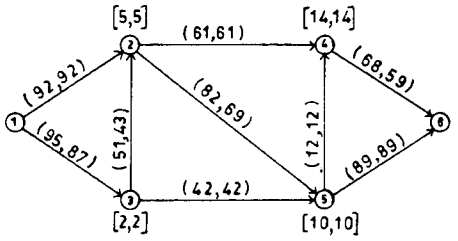
a = 157, b = 157, c = 0, d = 0.

Figure 44.



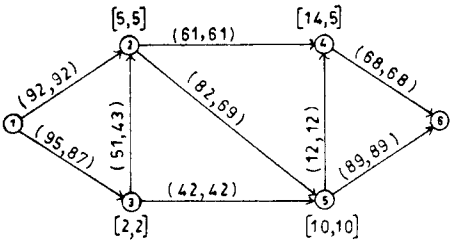
a = 179, b = 148, c = 31, d = 31

Figure 45.



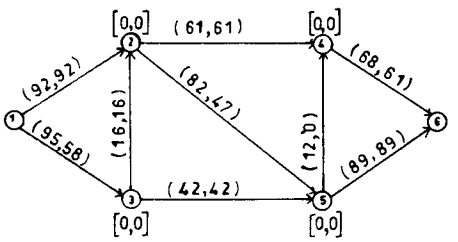
a = 179, b = 148, c = 31, d = 31.

Figure 46.



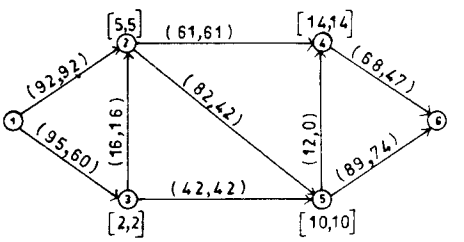
a = 179, b = 157, c = 22, d = 31.

Figure 47.



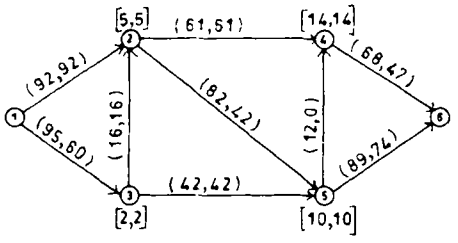
a = 150, b = 150, c = 0, d = 0.

Figure 48.



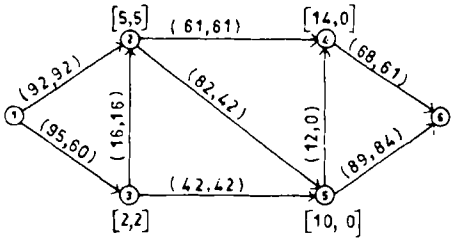
a = 152, b = 121, c = 31, d = 31

Figure 49.



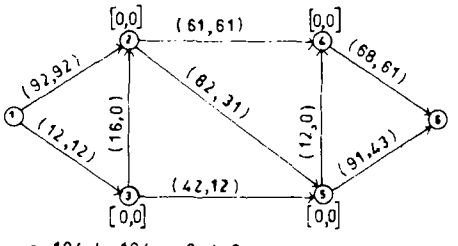
a =152, b =121, c =31, d =31.

Figure 50.



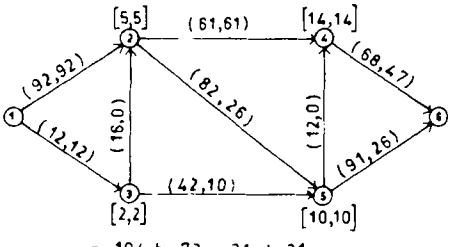
a =152, b=145, c = 7, d =31.

Figure 51.



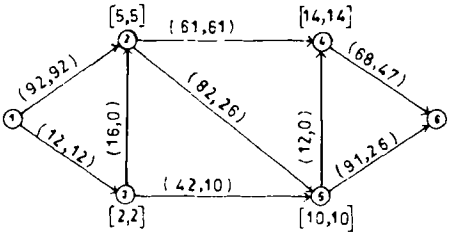
a = 104, b = 104, c = 0, d = 0

Figure 52.



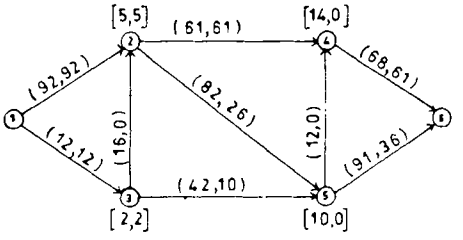
a = 104, b = 73, c = 31, d = 31.

Figure 53.



a = 104, b = 73, c = 31, d = 31.

Figure 54.



a = 104, b = 97, c = 7, d = 31.

Figure 55.

the flow values (the requirements met or satisfied) shown besides the nodes in square brackets such that the first number is the requirement and the second number is the flow (requirement met).

Table 12. A check list of figures for different cases.

	Capacity matrix											
Case	1	2	3	4	5	6	7	8	9	10	11	12
MFP	4	12	16	20	24	28	32	36	40	44	48	52
Case (i)	5	13	17	21	25	29	33	37	41	45	49	53
Case (ii)	6	14	18	22	26	30	34	38	42	46	50	54
Case (iii)	7	15	19	23	27	31	35	39	43	47	51	55

For each of the four cases, with each of the twelve capacity matrices, the corresponding network diagram with the optimal flows and other relevant details are given in the different figures as listed in table 12.

Acknowledgement

BKR thanks the CSIR for financial assistance in the form of JRF.

References

[1] Ford L R Jr and Fulkerson D R 1962 *Flows in Networks* (Princeton: Princeton Press)
[2] Hu T C 1969 *Integer Programming and Network Flows*, (Reading, Massachussetts: Addison-Wesley Publishing Company)