REVUE FRANÇAISE D'AUTOMATIQUE, D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE. RECHERCHE OPÉRATIONNELLE

B. GABUTTI
A. OSTANELLO-BORREANI

The labeling method on two-dimensional networks

Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle, tome 8, n° V2 (1974), p. 17-29.

<http://www.numdam.org/item?id=RO_1974__8_2_17_0>

R.A.I.R.O.
(8e année, mai 1974, V-2, p. 17 à 29)

THE LABELING METHOD ON TWO-DIMENSIONAL NETWORKS

par B. GABUTTI (1) et A. OSTANELLO-BORREANI (2)

Abstract. — *The Ford-Fulkerson labeling method has been extended, for solving the max-flow problem on two-dimensional networks, i.e. on networks with directed arcs on a cartesian plane* $\mathbf{R}^2$. *Computationally, the algorithm has a good efficiency for networks with at least one node of degree* $\leqslant 3$ *or with some symmetries.*

1. INTRODUCTION AND DEFINITIONS

1.1. A two-dimensional network $G_{\vec{b}} = (X, A\,; (\vec{b}_j))$ is defined by a set X, $|X| = n$, of nodes x_i, a set A, $|A| = m + 1$, of arcs $a_j = (x_i, x_k)$, and a family of $m + 1$ directions $\vec{b}_j = (\alpha_j, \beta_j)$, one for each arc, i.e. of vectors of $\mathbf{R}^2$ such that $\alpha_j^2 + \beta_j^2 = 1 \quad \forall_j$, [2].

The node-arc *incidence matrix* of $G_{\vec{b}}$ is the $2n$, $(m + 1)$ real matrix

$$M = \{ \vec{a}_{ij} \}, \tag{1}$$

whose general entry is defined as

$$\vec{a}_{ij} = \begin{cases} \vec{b}_j' & \text{if } a_j \text{ leaves } x_i \\ -\vec{b}_j' & \text{if } a_j \text{ enters } x_i \\ \binom{0}{0} & \text{if } a_j \text{ is not incident to } x_i \end{cases}$$

where $\vec{b}'$ is the transposed of $\vec{b}$.

---

(1) Laboratorio di Analisi Numerica (L.A.N.), Pavia, Italy.
(2) Politecnico di Torino, Torino, Italy.

This research was supported by the Gruppo Nazionale per l'Analisi Funzionale e le sue Applicazioni of the CNR (Consiglio Nazionale delle Ricerche) of Italy.

1.2. Suppose that each arc $(x, y) \in A$ has a given capacity $c(x, y) > 0$ and that two nodes of X, say s and d, are respectively the *source* and the *sink* for a flow « entering » $G_{\vec{b}}$ with a given direction $\vec{b}_l$, of the return arc $(d, s) = a_l$.

A *flow* $\vec{f}$ on $G_{\vec{b}}$ is any vector of

$$\mathbf{R}^{m+1} \quad , \quad \vec{f} = (f_1, f_2, \ldots, f_m, f_l),$$

satisfying the conservation equations (or of static equilibrium) at the nodes

$$M\vec{f} = \vec{0}. \tag{2}$$

Any node x, where (2) holds, is *equilibrated*. f_l is the *value* of the flow $\vec{f}$, which is *feasible* if and only if

$$-c(x, y) \leqslant f(x, y) \leqslant c(x, y) \qquad \forall (x, y) \in A - \{a_l\} \tag{3}$$

$$c_l = +\infty$$

An obvious feasible flow is $\vec{f} = \vec{0}$. The problem of finding the maximum value of f_l is the analogue of the max flow problem for the topological graph $G = (X, A)$.

The simplex method can obviously be used, since

$$\begin{array}{l} \text{« max } f_l \\ \quad \text{such that (2) and (3) hold »} \end{array} \tag{4}$$

is a linear program.

1.3. The motivation of this problem comes from the study of optimum condition for the static equilibrium of planar pinned trusses with concentrated loads (see for example [9]).

An optimization problem, on such kind of structures, is that of evaluating the maximum loads supported by the structure under equilibrium conditions, given that the internal stress of any single bar cannot be greater then a known value.

For different kinds of structures, with different kinds of constraints and loads, the problem is reducible to a linear program (see for example [3], [5]).

1.4. A graph-theoretical approach of the problem has been performed in [2] and [8], and some classical results, such as the flow-path decomposition and the max flow-min cut theorem, have been found using a suitable definition of elementary path on $G_{\vec{b}}$, or b-path.

In this paper we intend to present an extension of the Ford-Fulkerson labeling method, [6], using the notion of b-path (or simply *path* in this context). For the paper be self-containing, we shall report the definition given in [2].

2. *b*-PATH

2.1. A *b*-path of $G_b^{\rightarrow}$ from s to d is a subset $P \subseteq A$, connecting s to d and satisfying the rules given below.

Denote by $X(P)$ the set of nodes incident to the arcs of P.

Start : the return arc enters s; $a_t \in P$.

Rule (a) : If *one* arc $a_j \in P$ enters the node x and the set, $\omega(x)$, of arcs incident to x, contains either

1) one arc $a_k \parallel a_j$, $a_k \notin P$, or

2) two arcs $a_{k_1} \overset{\parallel}{\parallel} a_{k_2} \overset{\parallel}{\parallel} a_j$, a_{k_1} and $a_{k_2} \notin P$,

then $x \in X(P)$. The path leaves x through a_k or both a_{k_1} and a_{k_2}.

Rule (b) : If two or more arcs not pairwise parallel of P enter x and $\omega(x)$ contains either

1) one arc $a_k \notin P$, parallel to none of them, or

2) two arcs $a_{k_1} \overset{\parallel}{\parallel} a_{k_2} \overset{\parallel}{\parallel} a_j$, $\forall\, a_j \rightarrow a_j \in P \cap \omega(x)$, a_{k_1} and $a_{k_2} \notin P$, then $x \in X(P)$ and the path leaves x through a_k or a_{k_1} and a_{k_2}.

If $x \equiv d$ the case (a. 1) and (b. 1) also hold, but then the condition $a_k \notin P$ must be dropped when $a_k \equiv a_t$.

Rule (c) : If (only) *two parallel arcs* of P enter x, then $x \in X(P)$ and the corresponding branch of the path ends at x.

Rule (d) : If several arcs enter x and $\omega(x) - P = \varnothing$, then $x \in X(P)$ and the corresponding branch of P ends at x.

End : when none of the rules can be applied or when $d \in X(P)$ and all nodes, incident to the arcs of P, belong to $X(P)$.

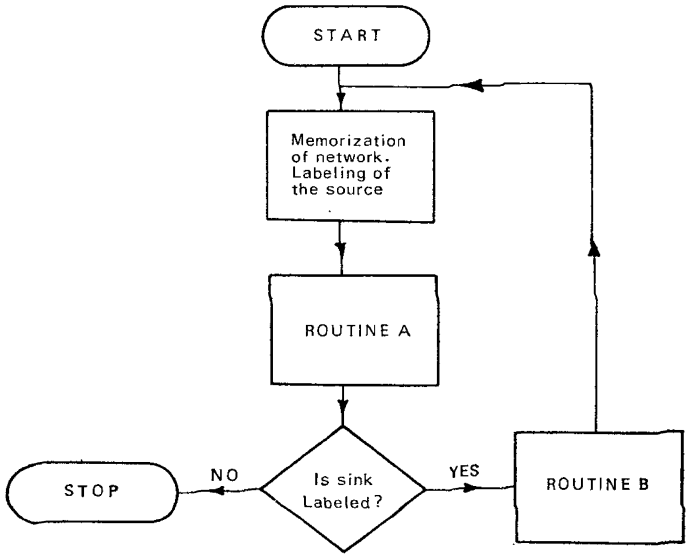
2.2. A path-flow $\vec{f}^{(k)} = (f_j^{(k)})$ is any flow whose support is a path P_k. The « support », $\| \vec{f}^{(k)} \|$, of a vector $\vec{f}^{(k)}$ on A is usually defined as the subset of arcs of A for which $f_j^{(k)} \neq 0$.

Using the definition of flow and the rules of 2.1, it's easy to show that, *if it exists, the support of a path-flow is minimal.* Thus any path, support of a flow, is called *elementary path.*

3. THE LABELING METHOD

3.1. Following the method of Ford-Fulkerson, [6], the present algorithm consists of two Routines :

Routine A for the labeling of nodes at the search of a flow augmenting path P_k from s to d;


START
Memorization of network. Labeling of the source
ROUTINE A
Is sink Labeled?
NO
STOP
YES
ROUTINE B

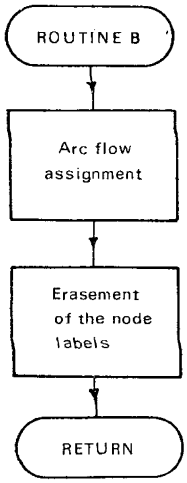

ROUTINE B
Arc flow assignment
Erasement of the node labels
RETURN

Figure 1 *a*

Routine B for increasing the flow through P_k and evaluating the residual capacities or for concluding that the present flow is maximal.

The algorithm can start with the zero flow.

The present state of any arc (x, y) is indicated by the label

$$[f(x, y) / c^s(x, y)], \tag{5}$$

where $f(x, y)$ is the present flow, $f \gtrless 0$, and c^s is the conformed capacity, referred to $f(x, y)$, positive if $f(x, y) \geqslant 0$, negative if $f(x, y) < 0$.

Since any incremental flow $\vec{f}^{(k)}$ can have arc flows $f_j^{(k)} \gtrless 0$, any arc has two *residual capacities* c_r^s, defined as

$$c_r^s = \begin{cases} c^s(x, y) - f(x, y) & \text{if} \quad f^{(k)}(x, y) \cdot f(x, y) > 0 \\ -c^s(x, y) - f(x, y) & \text{if} \quad f^{(k)}(x, y) \cdot f(x, y) < 0. \end{cases} \tag{6}$$

If $f(x, y) = 0$, then $c_r^s(x, y) \cdot f^{(k)}(x, y) > 0$, $|c_r^s| = c$.

3.2. Routine A

During this routine a node can be in one of the three states : a) unlabeled b) labeled and unscanned, c) labeled and scanned (or equilibrated).

Initially it is $\vec{f} = \vec{0}$ and all nodes are unlabeled.

The source receives the label [/, $\varepsilon(s)$], where ε is an arbitrary positive real number; s is now in state b), and all other nodes in state a).

In general, let x be any node in state b), having a label of the form $[y^{\pm}, \varepsilon(x)]$, with $\varepsilon(x) \gtrless 0$.

We can meet different cases, corresponding to the various rules of the definition of P_k.

Case I : only *one* arc « enter » x, with direction $\vec{b}$ [see rule (a)].

Here and in the following, the expression « arc entering x » is for arc belonging to the support of $\vec{f}^{(k)}$, and « arc leaving x » is for arc $\in \omega(x) - P_k$. Moreover I denotes the set of indices of the choosen leaving arcs, $I = \{1\}$ for case (a. 1), $I = \{1, 2\}$ for case (a. 2).

The *source s* is always in this case ; starting from s, a maximal incremental flow is sent through P_k, consistent with the residual capacities c_r^s, following the rules given below. Such a flow cannot be augmented along P_k, but only reduced.

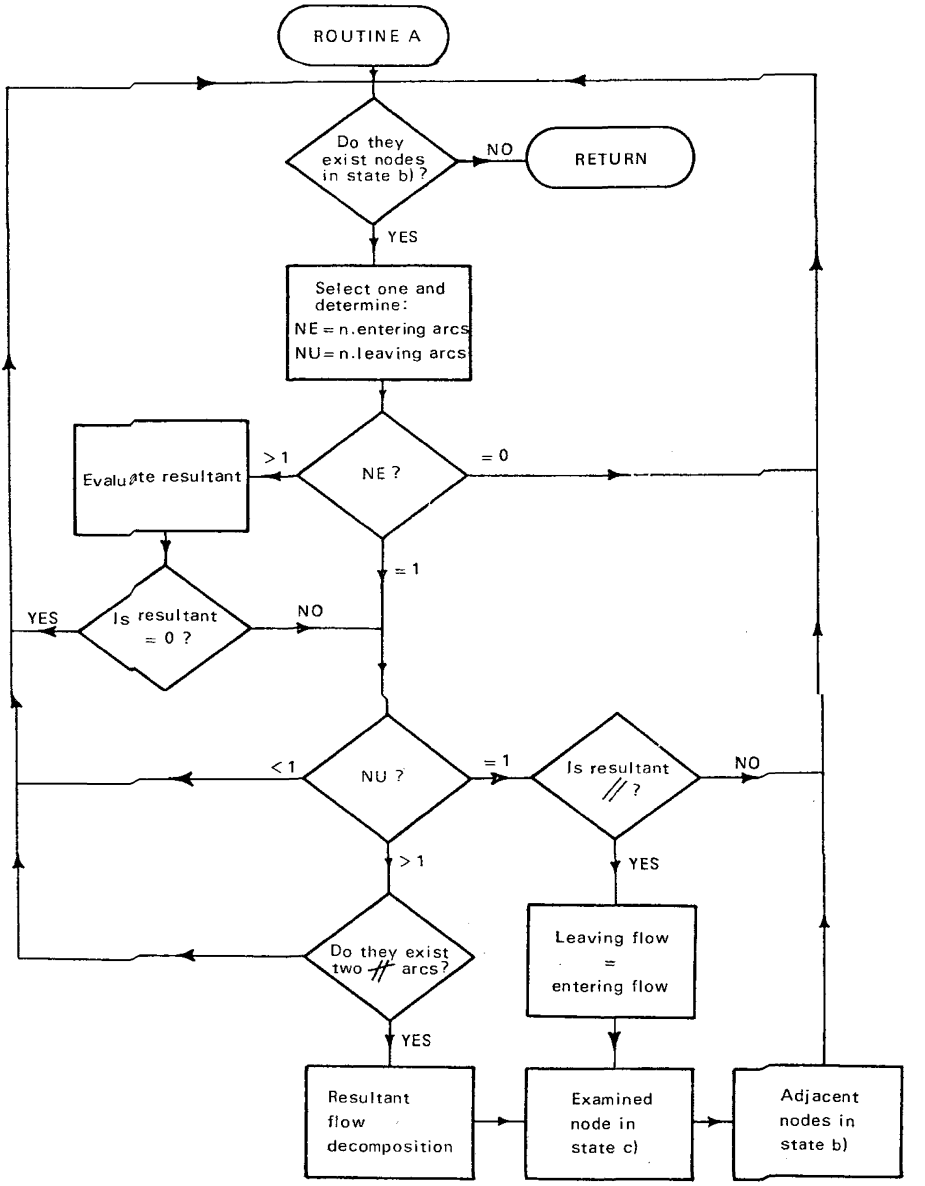

ROUTINE A
Do they exist nodes in state b) ?
NO
RETURN
YES
Select one and determine:
NE = n.entering arcs
NU = n.leaving arcs
Evaluate resultant
> 1
NE ?
= 0
= 1
YES
Is resultant = 0 ?
NO
< 1
NU ?
= 1
Is resultant // ?
NO
> 1
YES
Do they exist two // arcs?
Leaving flow = entering flow
YES
Resultant flow decomposition
Examined node in state c)
Adjacent nodes in state b)

Figure 1 *b*

If rule (a) holds, then the increment $\varepsilon(x)$ at the node x is distributed along the leaving arcs in the portions ε_i, $i \in I$, following a conservation equation of kind

$$\pm (\vec{b})\varepsilon(x) + \sum_{i \in I} \pm (\vec{b}_i)\varepsilon_i = \vec{0}. \tag{7}$$

In both cases there exists one and only one solution with $\varepsilon_i \neq 0$.

i) Correspondingly to the ε_i, evaluate the *relative residual capacities* v_i :

$$v_i = \frac{c^s_{ri} - \varepsilon_i}{\varepsilon_i} \qquad \forall\, i \in I. \tag{8}$$

The partial flows ε_i are or not consistent with the capacities, according to whether it is $v_i > 0$ or $v_i < 0$.

If $v_i = 0$ the arc a_i will result saturated.

ii) If $x = s$, for any v_i $\forall$ i,

or if $x \neq s$ and $\exists$ $k \in I$ such that $v_k < 0$,

then go to iv).

iii) If $x \neq s$ and $\forall$ $i \in I$ $v_i > 0$,

then go to v).

iv) Select

$$\frac{c^s_{rt}}{\varepsilon_t} = \min_k \frac{c^s_{rk}}{\varepsilon_k} = \lambda. \tag{9}$$

If $\lambda \neq 0$, reevaluate *all* the incremental arc-flows on P_k with

$$\varepsilon'_i = \lambda \varepsilon_i \qquad \forall\, a_i \in P_k \tag{10}$$

(and denote then again ε_i).

Then go to v).

If $\lambda = 0$, then $c^s_{ri} = 0$ and the Path is saturated.

In this case go back along P_k searching for another breakthrough path not using the arc a_t in the same « direction » as P_k.

v) The node x is scanned.

Assign to the adjacent nodes x_j the labels

$$[x^{\pm}, \varepsilon_i(x_j)] \tag{11}$$

where $\pm$ indicates an $\begin{cases} \text{increase of } |f_i| \\ \text{decrease of } |f_i| \end{cases}$

and

$$\varepsilon_i(x_j) = \pm \min [|\varepsilon_i|, |c^s_{ri}|] \tag{12}$$

with $+$ if $\varepsilon_i > 0$, $-$ if $\varepsilon_i < 0$.

Case II : two or more arcs of P_k enter x and there exists one or two leaving arcs [rule (b)].

The node x has been assigned several labels, $\varepsilon_i(x)$.

The vector-sum, $\varepsilon^*\vec{b}^*$, of the feasible entering flows must satisfy the conservation equation (7) with the leaving arcs (for the existence of P_k).

If so, then x is labeled and unscanned. Procede to the discussion of Case I, with $\vec{b} = \vec{b}^*$.

If not, then P_k does not continue after x :

(13) go back along P_k, searching for another node $y \in N(P_k)$, if it exists, from where another path $P'_k \neq P_k$ proceeds from s to d ; if it doesn't then go to vi).

Case III : two or more arcs of P_k enter x and none is leaving [rules (c) and (d)].

The equilibrium condition must be satisfied, for the existence of P_k. If not, go to (13).

vi) STOP :

Routine *A* ends when the sink d has been labeled (and scanned) and *all* other nodes of P_k are in state c) [breakthrough], or when either d has been labeled (and unscanned) and $\exists\ x \in P_k$ in state b), or d has not been labeled (non-breakthrough).

In the first case go to Routine B.

In the letter cases, GO TO 3.4.

3.3. Routine B

The sink has been marked with one or more labels and the total increment is $\Sigma\ \varepsilon_i(d)\vec{b}_i = \vec{\varepsilon}_l b_l$.

Then go back on P_k for a flow change : if y is labeled by $[x^{\pm}, \varepsilon_i(y)]\ \forall\ i$, replace on the arc $a_i = (x, y)$

$$f(x, y) \quad \text{by} \quad f'(x, y) = f(x, y) + \varepsilon_i(y). \tag{14}$$

Then change the labels on arcs. Discard the old labels on nodes, and start again with ROUTINE A.

3.4. Stop

When the non-breakthrough occurs, the present flow is maximal. It can be $\vec{f}_{\max} = \vec{0}$.

3.5. Block diagram

See fig. 1.

3.6.

The algorithm is finite since, at any new path P_k, at least one arc is saturated, and the arcs are at most $m + 1$.

Moreover it's finite the number of combinations of $2(m + 1)$ arcs, taken l at a time, $l = |P_k|$.

The proof of the algorithm is contained in the following theorem, which is an extension of a known theorem on ordinary networks, [4].

Theorem : A flow $\vec{f}$ an $G_b^{\rightarrow}$ is maximal if and only if it is $f^*_{l\max} = 0$, on an associated network $G_b^{*\rightarrow}$ obtained from $G_b^{\rightarrow}$ by replacing the capacities $\pm c_i$ by $c_i^* = \pm c_i - f_i$.

4. COMPUTATIONAL EXPERIENCE

For many networks, see [2], the application of the foregoing algorithm can present considerable difficulties. These are essentially due to the impossibility of stating « a priori » if a given succession of arcs constitutes or not an elementary path of $G_b^{\rightarrow}$. More precisely, in order to proceed to a node labeling, the program must test if any of the rules of 2.1 holds. If none of them holds, at some node x, we are not allowed to deny the existence of an elementary path, before all the combinations (according to the rules of 2.1), of the arcs leaving all the labeled and unscanned nodes, are tried.

The Program, in FORTRAN IV, has been tested on a computer IBM 360/44. The computer times run around 60 secs for networks with $|X| \cong 10$, $|A| \cong 20$.

The Program consists of the three following parts.

4.1. Network memorization

The memorization of the network has been performed introducing, for every arc, a record of kind :

(15)

NP (*I*)	NA (*I*)	ALFA (*I*)	*c* (*I*)

$I = 1, ..., M,$

where NP (I) = initial node of arc I,

NA (I) = terminal node of arc I,

ALFA (I) = inclination (degrees) of the directed arc I, referred to the return arc,

$c\,(I)$ = arc capacity (positive).

The return arc is taken as directed in the usual way, i.e. from sink to source, and it's defined by a record in similar way. The network is consequently characterized, independently from the frame of reference.

4.2. Routine A

Whenever we get the decomposition of the entering resultant flow at a node N, in state b), every arc leaving N take a record of kind :

(16)

NP (J)	NA (J)	ALFA (J)	$c\,(J)$	FK (J)

$J = 1, ..., |P_k|$

where FK (J) = arc flow on a_J along P_k.

The path P_k is singled out from the computer, memorizing, for any arc, the order index in the succession defined by the record (15).

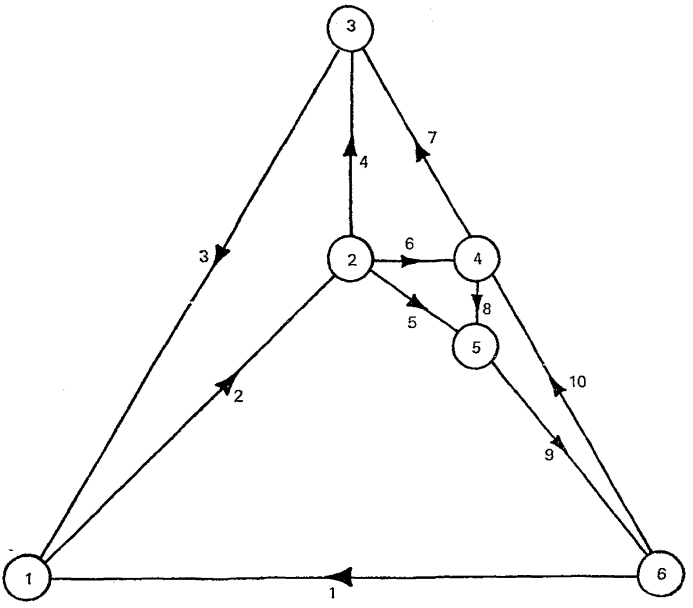


Figure 2

Such a device is necessary either to reduce the arc flows, conforming to (10), or to assign the arc flow in Routine B.

At the nodes, where more then two leaving arcs exist, the entering flow is first decomposed on any two arcs.

Nodes of this kind cannot be considered definitively equilibrated, i.e. in state c), but they are memorized as nodes in a « special state »; precisely : if the path P_k does exist from s to d, then they enter in state c); if doesn't, the last met node must be rescanned in search of another combination of leaving arcs, for the equilibrium of the entering flow.

For some networks, with many nodes in state b), the determination of the elementary path can require rather long times of calculation. We have partially obviate such a truble, by choosing, when possible, the strategy of turning about the nodes with more then two leaving arcs. That has been realized by imposing the *priority* of the scanning of nodes with only two leaving arcs. Often such a device allows to avoid a combinatorial analysis.

For example, in the case of the sample problem exposed in 4.4, the Program singles out the elementary path, following the node succession 1, 3, 2, 4, 5, 6. Without the mentioned priority, at node 2, we should have done a choice between couples of arcs 4, 5, 6.

4.3. Routine B

Any arc, belonging to the completed path P_k, is assigned, beside the record (16), the label

$$\text{FA}(J) \qquad J = 1, \ldots, |P_k|$$

where FA (J) = present arc flow.

Then FK (J) is set equal to zero and the Program goes to 4.2.

4.4. Sample Problem

In the network of fig. 2, it does exist only one elementary path; on it is optimal the flow reported in tab. 2. The features of the network are reported in tab. 1 (input of the Program).

TWO - DIMENSIONAL NETWORK CASE N. 1

N. NODES = 6
N. ARCS = 10
SOURCE = 1
SINK = 6

RETURN ARC	LEAVES	6	ENTERS	1	INCL. =	0.0 (DEGREES)		
ARC N. 2	LEAVES	1	ENTERS	2	INCL. =	45.00 (DEGREES)	CAPACITY =	1.00000
ARC N. 3	LEAVES	3	ENTERS	1	INCL. =	-120.00 (DEGREES)	CAPACITY =	1.00000
ARC N. 4	LEAVES	2	ENTERS	3	INCL. =	90.00 (DEGREES)	CAPACITY =	1.00000
ARC N. 5	LEAVES	2	ENTERS	5	INCL. =	- 45.00 (DEGREES)	CAPACITY =	1.00000
ARC N. 6	LEAVES	2	ENTERS	4	INCL. =	0.00 (DEGREES)	CAPACITY =	1.00000
ARC N. 7	LEAVES	4	ENTERS	3	INCL. =	135.00 (DEGREES)	CAPACITY =	1.00000
ARC N. 8	LEAVES	4	ENTERS	5	INCL. =	- 90.00 (DEGREES)	CAPACITY =	1.00000
ARC N. 9	LEAVES	5	ENTERS	6	INCL. =	-56.65 (DEGREES)	CAPACITY =	1.00000
ARC N. 10	LEAVES	6	ENTERS	4	INCL. =	100.00 (DEGREES)	CAPACITY =	1.00000

TABLE 1

RESULTS

FLOW PATH N. 1 COMPLETE

ARC N. 1	MAX. FLOW =	0.2679487 E 00
ARC N. 2	MAX. FLOW =	0.8965754 E 00
ARC N. 3	MAX. FLOW =	-0.7320511 E 00
ARC N. 4	MAX. FLOW =	0.1000000 E 01
ARC N. 7	MAX. FLOW =	-0.5176360 E 00
ARC N. 5	MAX. FLOW =	0.5176407 E 00
ARC N. 6	MAX. FLOW =	0.2679466 E 00
ARC N. 8	MAX. FLOW =	0.1901358 E 00
ARC N. 9	MAX. FLOW =	0.6658028 E 00
ARC N. 10	MAX. FLOW =	0.5647412 E 00

TABLE 2

REFERENCES

[1] BERGE C. and GHOUILA-HOURI A., *Programmes, jeux et réseaux de transport*, Dunod, Paris, 1962.

[2] BORREANI A., « Two-dimensional networks » ORC 70-6, february 1970. Operations Research Center, University of California Berkeley.

[3] CHARNES A. and COOPER W. W., *Management models and industrial applications of linear programming*, vol. II, J. Wiley & Sons, 1961.

[4] DANTZIG G. B., *Linear programming and extensions*. Princeton University Press, 1963.

[5] DORN W. S. and GREENBERG H. J., « Linear Programming and Plastic Limit Analysis of Structures ». Quarterly of Applied Math., vol. 15, pp. 155-167, 1957.

[6] FORD L. R. and FULKERSON D. R., *Flows in networks*. Princeton University Press, 1962.

[7] FULKERSON D. R., « Networks, Frames, Blocking Systems ». Mem. RM - 5368-PR, may 1967. *Rand Corporation.*

[8] OSTANELLO-BORREANI A., « Flussi sui reticoli : alcuni recenti sviluppi ». Atti Giornate AIRO-Torino, 1971.

[9] PRAGER W., « Mathematical Programming and theory of Structures ». J. of the Society for Industrial and Applied Mathematics, vol. 13, n° 1, pp. 312-332, 1965.