RM-74-18

AN ADAPTIVE ROUTING TECHNIQUE FOR CHANNEL SWITCHING NETWORKS

D. E. Bell

A. Butrimenko

October 1974

An Adaptive Routing Technique For Channel Switching Networks

D. E. Bell
A. Butrimenko

1. Introduction

The performance of a communication system depends greatly on the technique used for finding good routes for transmitting the information. In order to improve the robustness and reliability of such systems, we propose here a decentralized or distributed technique to control the information flows. Procedures of this kind have already been proposed [1, 2, 3]; the first two being oriented to channel switching (or circuit switching) systems have performed very well compared with non-adaptive routing techniques.

It seems that even better results could be obtained if the chosen criteria were incorporated directly into the algorithm. For example, neither [1] nor [2], both of which minimize the probability of losses, directly use the achieved probabilities as control parameters. The first is based on the shortest route, and the second takes into account the number of successes and failures in previous attempts to establish connections.

The routing technique to be described here is based on the same principle of distributed control as in [1]. We consider a network in which every node (exchange) receives control information only from its neighbouring nodes. In every node there is stored a special routing matrix and every node estimates continuously the probability of the trunks to each of its neighbours being blocked, that is, being unavailable for transmission, which will depend on the congestion in the system. The routing matrix at node i has entries W_{ij}^{k} which

represent the estimated probability of a message reaching destination k from the neighbouring node j. Clearly, when routing a message to destination k the node j, to which it should be sent next, is that which maximizes W_{ij}^{k} among those nodes j for which trunk (i,j) is unblocked. If all trunks are blocked, the message is considered to be lost.

To summarize, then, we wish to present an algorithm for such networks which converges to the optimal routing policy, under the assumptions:

i) only local information is available; and

ii) the blocking probabilities remain constant. (1.1)

In practice these assumptions will be fulfilled if the traffic is light, in which case the blocking probabilities will be approximately constant. In other cases this policy may not be optimal since the blocking probabilities will depend partially on the routing policy. The behaviour of this algorithm in this case will be investigated later but it seems likely that it will yield a good if not optimal policy.

2. The Algorithm and Its Convergence

2.1 Example

Let us begin with an example. In Figure 1 there is an example of a network in which the aim is to transmit a call from node 1 to the destination.

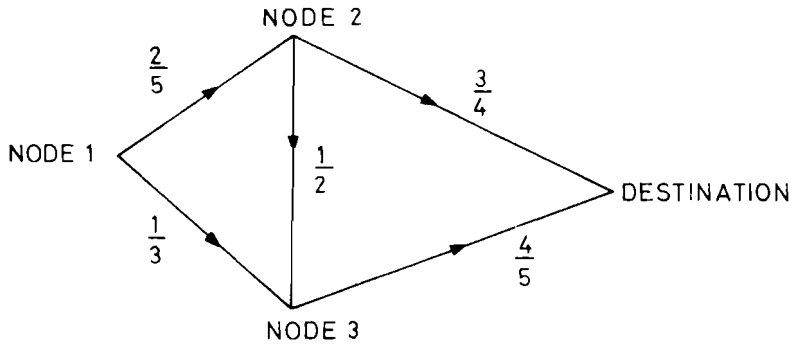


FIGURE 1

Let P_1, P_2, P_3 be the estimates at each node of the probability of successfully transmitting a message to the destination. Using conditional probabilities we can see that

$$P_3 = \frac{4}{5}$$

$$P_2 = \frac{3}{4} + (1 - \frac{3}{4}) \frac{1}{2} P_3 = \frac{17}{20} \quad .$$

Now it is clear that at node 1 it is better to transmit the message to node 2 than node 3 since $P_2 > P_3$, hence

$$P_1 = \frac{2}{5} P_2 + (1 - \frac{2}{5}) \frac{1}{3} P_3$$

$$= \frac{2}{5} \cdot \frac{17}{20} + \frac{3}{5} \cdot \frac{1}{3} \cdot \frac{4}{5} = \frac{1}{2} \quad .$$

But suppose P_2, P_3 were not known exactly, but rather were only estimates. If, in our example, P_2, P_3 were currently estimated to be $\frac{1}{2}$, $\frac{4}{5}$ respectively, the policy at node 1 would be to transmit messages to node 3 in preference to node 2, giving a new estimate of

$$P_1 = \frac{1}{3} P_3 + \frac{2}{3} \cdot \frac{2}{5} P_2 = \frac{2}{5}$$

at node 1.

What will be proved here is that for all networks the estimates obtained in this way will converge if repeated iteratively, and to the correct solution, from any set of initial estimates. Incidentally, Fig. 1 is an acyclic network and it is easy to show that such an algorithm converges finitely in no more than n - 1 steps (n nodes) (use Theorem 16.2 in Harary [4]).

This algorithm may be repeated once for each possible destination so we need only consider the case with one destination.

Recall also that we assume that the probability p_{ij} of arc (i,j) being unblocked is constant, independent of other arcs and independent with respect to time.

2.2 The Algorithm

Let $P = (P_1,\ldots,P_n)$ be the set of node probabilities ("optimal node probabilities") corresponding to an optimal routing policy (in the sense of assumption (1.1)) where P_i is the probability of a message at node i being successfully transmitted to the destination and $P_{destination}$ is always equal to 1. Suppose that all but P_k for some node k are known and that it only remains to find P_k. It is evident that the best policy for a message at k is to transmit it to the node which, amongst those to which the arcs are unblocked, gives the best opportunity for reaching the destination. Hence, the probability P_k may be calculated by the following recursion in which we assume, without loss of generality, that $P_1 \geq P_2 \geq \cdots \geq P_n$:

$$P_k = p_{k1}P_1 + (1 - p_{k1})p_{k2}P_2 + (1 - p_{k1}) \cdot (1 - p_{k2})p_{k3}P_3 + \cdots \quad . \tag{2.2.1}$$

For compactness, define a vector $F(p,P)$ on a matrix p of arc probabilities and vector P of node probabilities, having components

$$F_k(p,P) = p_{ki_1}P_{i_1} + (1 - p_{ki_1})p_{ki_2}P_{i_2} + \cdots \tag{2.2.2}$$

where

$$P_{i_1} \geq P_{i_2} \geq \cdots \geq P_{i_n} \quad .$$

The algorithm can now be stated quite simply. Begin with an initial guess P^0, $0 \leq P^0 \leq 1$ of the node probabilities and proceed with a recursion defined by

$$P_i^{t+1} = F_i(p,P^t) \quad , \qquad i = 1,\ldots,n.$$

It will be shown that this recursion converges to the optimal node probabilities for all initial values.

2.3 Convergence

Note that relation 2.2.1 may be written

$$P_k = \sum_j a_{kj}P_j$$

for some constants a_{kj} which depend on the particular policy selected. By optimality

$$F_k(p,P) \geq \sum_j a_{kj}P_j$$

for any policy which gives rise to the a_{kj}.

Theorem 1. If for two arbitrary vectors P,Q, $P \leq Q$ then

$$F(p,P) \leq F(p,Q) \quad .$$

Proof. $F_k(p,P) = \sum_j a_{kj}P_j$ for some a_{kj}

$\leq \sum a_{kj}Q_j$ since $P \leq Q$

$\leq F_k(p,Q)$ by definition of F .

Corollary 1. If $P^o \leq Q^o \leq R^o$ are three initial vectors for the recursion then $P^t \leq Q^t \leq R^t$ for all $t \geq 0$.

Corollary 2. If $P^t \leq P^{t-1}$ then $P^{t+1} \leq P^t$ and conversely if

$$P^t \geq P^{t-1}$$

then

$$P^{t+1} \geq P^t \quad .$$

Proof. If $P^t \leq P^{t-1}$ then by the theorem

$$F(p,P^t) \leq F(p,P^{t-1}) \quad ,$$

that is $F^{t+1} \leq P^t$. Similarly for the other case.

Corollary 3. If for any $t \geq 0$, $P^t \geq P^{t+1}$ or $P^t \leq P^{t+1}$ then the sequence converges.

Proof. The sequence is always bounded (by zero and one) and, if the conditions of the collorary apply, is monotonic by Corollary 2. Hence it converges.

Corollary 4. The recursion converges for initial values $P^o = 1$ and $P^o = 0$ (with $P^o_{destination} = 1$).

Proof. Corollary 3 applies to the case $t = 0$.

Theorem 2. The algorithm with initial values $P^o = 1$ converges to the optimal node probabilities.

Proof. By Corollary 4 the sequence converges. By Corollary 1 this solution is an upper bound for all solutions and hence is an upper bound for the optimal node probabilities. But by using the policy implied by this solution the upper bound

may be attained. Hence the solution is the vector of optimal node probabilities.

Theorem 3. The algorithm with initial values $P^O = 0$ converges to the optimal node probabilities.

Proof. Note that in this case P_k^t may be interpreted as the maximum probability of reaching the destination in at most t steps. Hence as $t \to \infty$ the recursion yields the maximum probability that the destination may be reached in an infinite number of steps, which is exactly the meaning of the optimal node probabilities.

Corollary 5. The recursion converges to the optimal node probabilities for all initial values P^O.

Proof. Theorems 3 and 4 show that with initial values 0 and 1 the algorithm converges to the optimal node probabilities. Since $0 \leq P^O \leq 1$ for all initial values, Corollary 1 implies the result.

3. Computational Experience

To investigate the performance of this algorithm over a range of blocking probabilities, a simulation program was written (Appendix 1). The algorithm was applied to the network as shown in Fig. 2.

Three kinds of initial situation were considered:

a) $P_i^O = 1 \quad \forall i$

b) $P_i^O = 0 \quad \forall i$

c) $P_i^O = R$ where R = random number $0 < R < 1$

except for destination nodes, whose destination probabilities are constant and equal to 1.

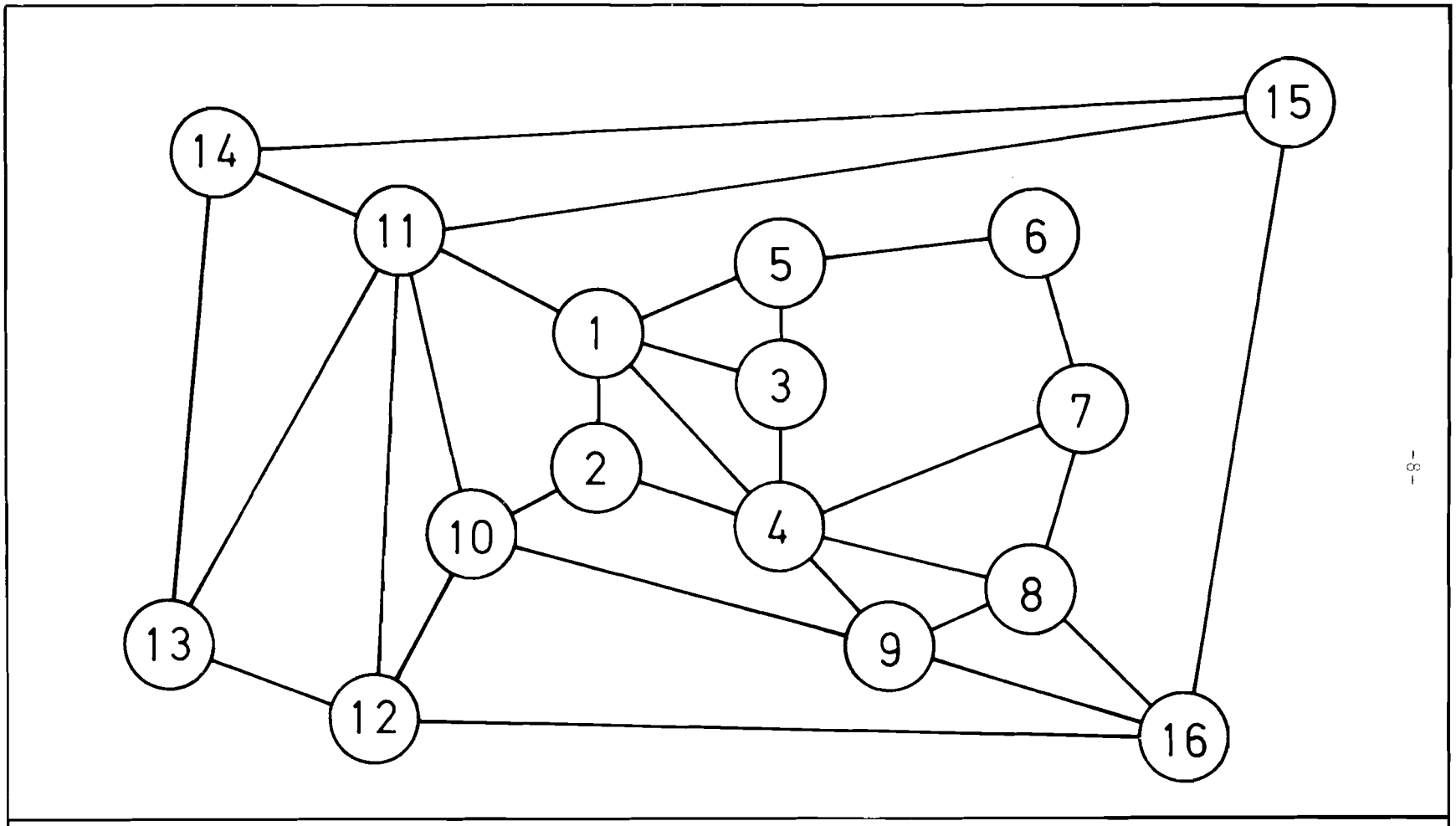


FIGURE 2 EXAMPLE OF A NETWORK

The probability of blocking was set constant and chosen at different intervals.

In Fig. 3 the dependence of the number of iterations on precision is represented taking the initial situation (a) where precision is defined as

$$\max_k \left| P_k^{\,t} - P_k^{\,t-1} \right| \leq \varepsilon \quad .$$

Figs. 4 and 5 show similar curves for initial situations (b) and (c) respectively. From these curves we can see that the system of equations converges very rapidly and that the number of iterations grows as a logarithm of precision.

The results described above apply to the case when all the equations are changed in every step and all $P_k^{\,t}$, $k = 1,2,\ldots,$ are available for the calculations in step $t + 1$. If we think of the practical implementation of the algorithm, it corresponds to the synchronous mode of control. This means that in each node some calculations have to be made at every iteration to update $P_k^{\,t}$. This must then be transmitted to all neighbouring nodes and is stored by them. If not all the new results are known, it is not possible to calculate the next $P_k^{\,t+1}$. Actually, the described calculations should be carried out by the same equipment as is responsible for the switching and there is no reason to assume the synchronization of it.

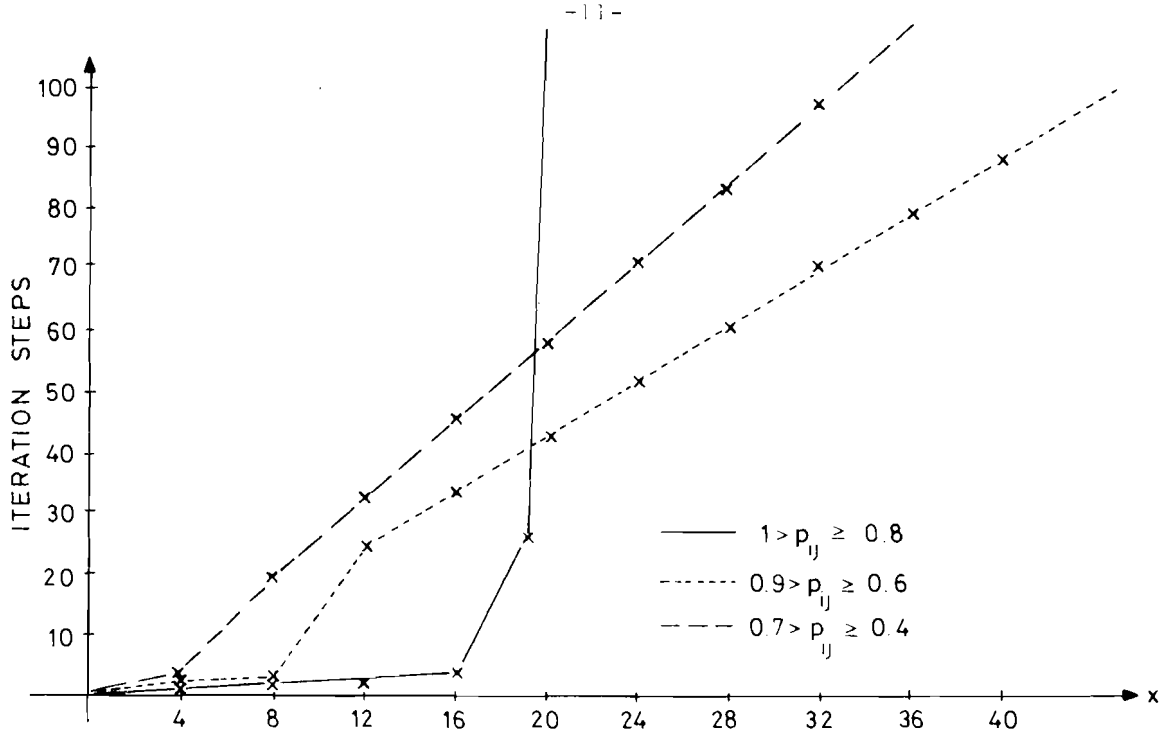
In Figs. 6 to 8 results are shown for a case when not all calculations are made simultaneously. An iteration of the algorithm consisted of choosing one node at random (equal probabilities) and calculating its new node probability, which was then available for the next iteration. A sequence of n (number of nodes of the network) of these iterations is counted as a step in the figures for comparison with Figs. 3 to 5.

Looking at the curves we can see that this desynchronization does not alter the speed of convergence very much.

We also considered another model where each node i missed the calculation with probability m_i. This model corresponds to the synchronous mode, but some nodes are allowed to miss calculation when the equipment is not available for the job; m_i is the probability of non-availability of the equipment.

In conclusion, we describe some additional investigations necessary to clarify the applicability of this algorithm.

1. The main aim of the algorithm is to make a communication system more robust and adaptive to the changing of the loading and network structure. This leads immediately to a variable probability of trunk blocking, p_{ij}, which we have considered as constant. Two problems now arise:
   a) Measurement (and perhaps prediction) of p_{ij}.
   b) In what way does the fluctuation of p_{ij} affect the convergence of the algorithm.
2. The next step for investigation should be the simulation of communication channel switching networks controlled by the help of this algorithm and by comparison with the control of others (e.g. [1, 2, 3]).



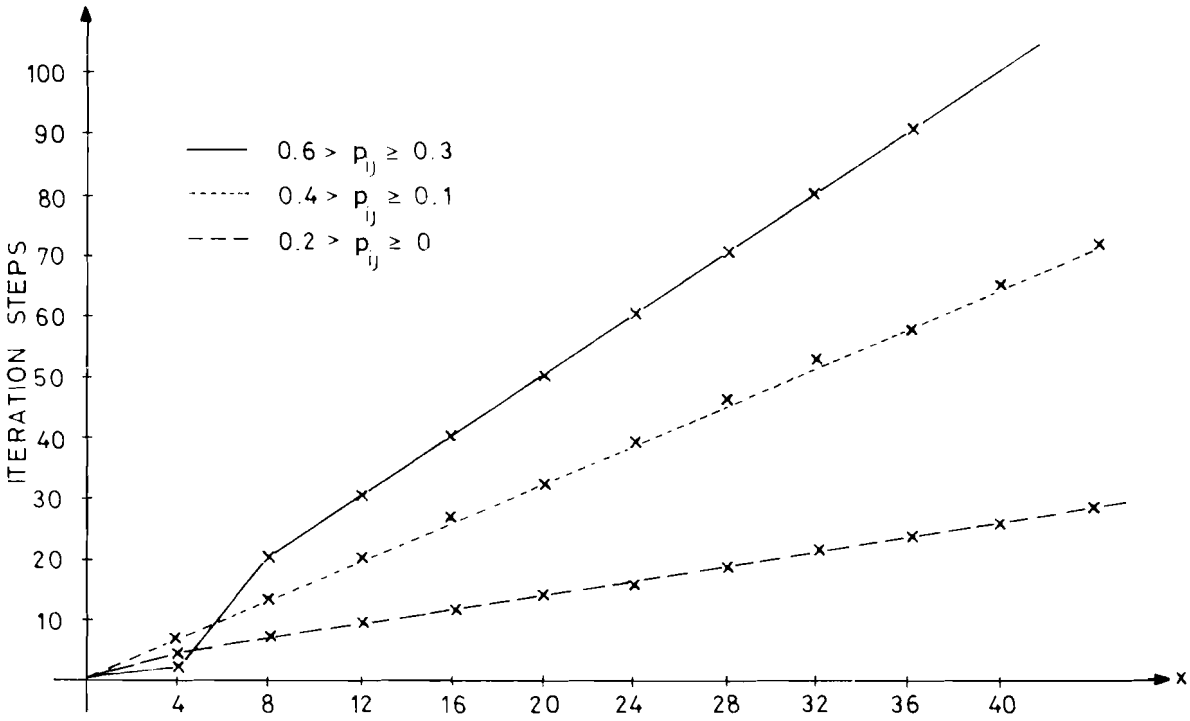
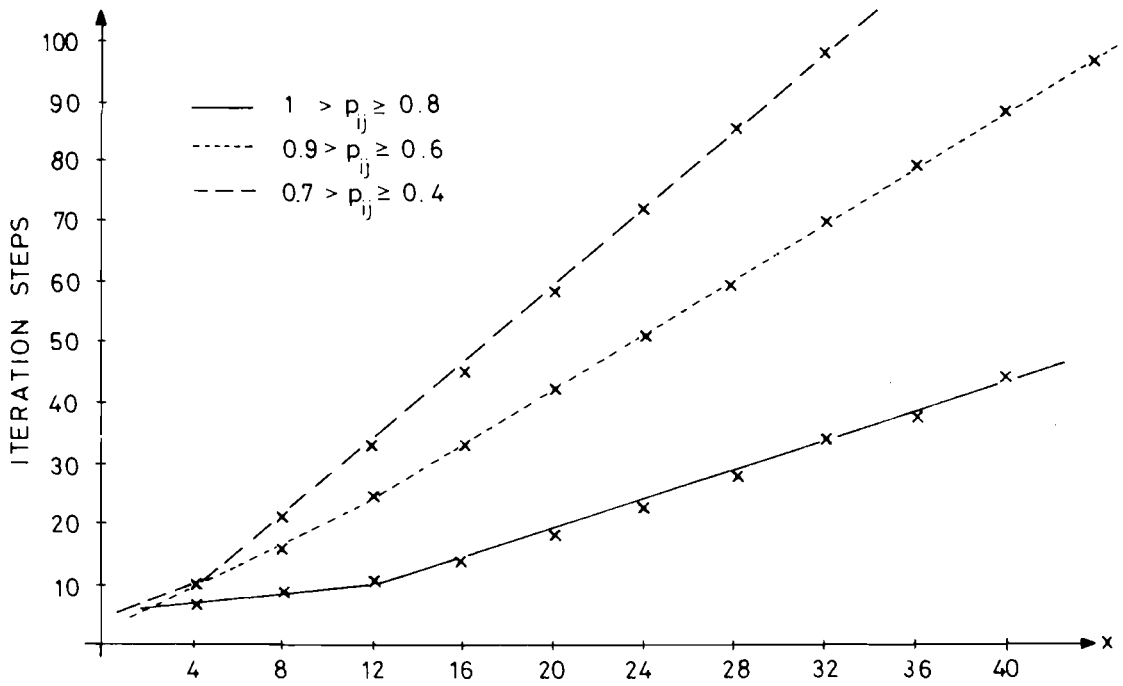


FIGURE 3

$eps = 2^{-x}$

THE NUMBER OF ITERATION STEPS DEPENDS ON eps.


ITERATION STEPS
100
90
80
70
60
50
40
30
20
10
4
8
12
16
20
24
28
32
36
40
x
1 > p_ij ≥ 0.8
0.9 > p_ij ≥ 0.6
0.7 > p_ij ≥ 0.4

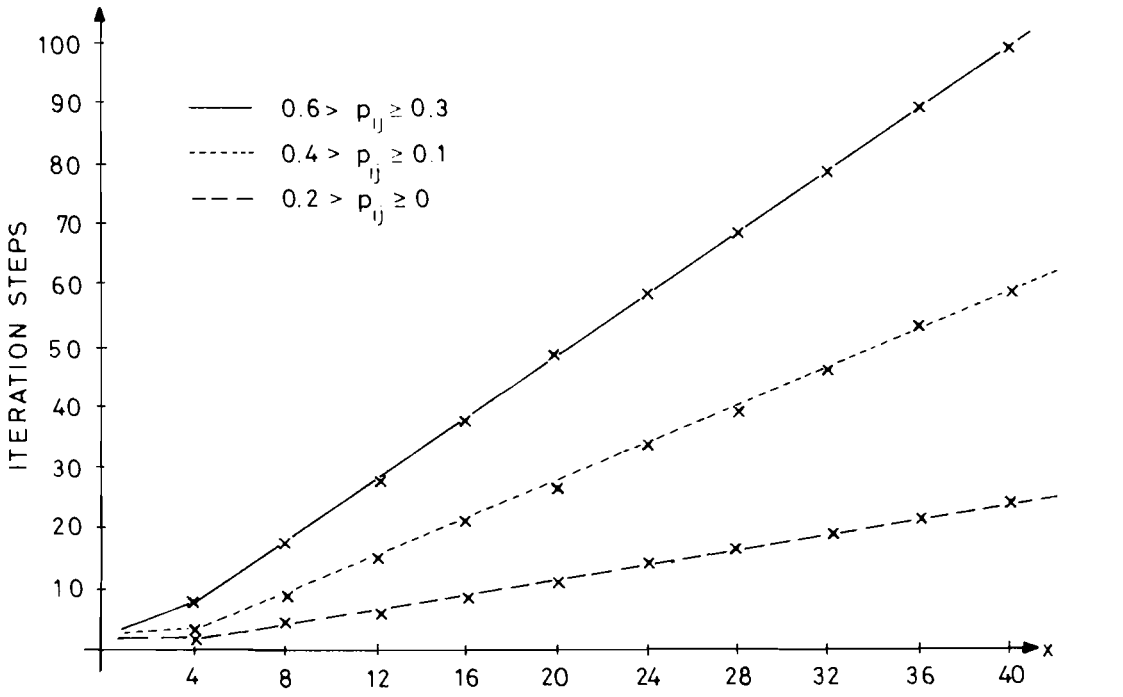
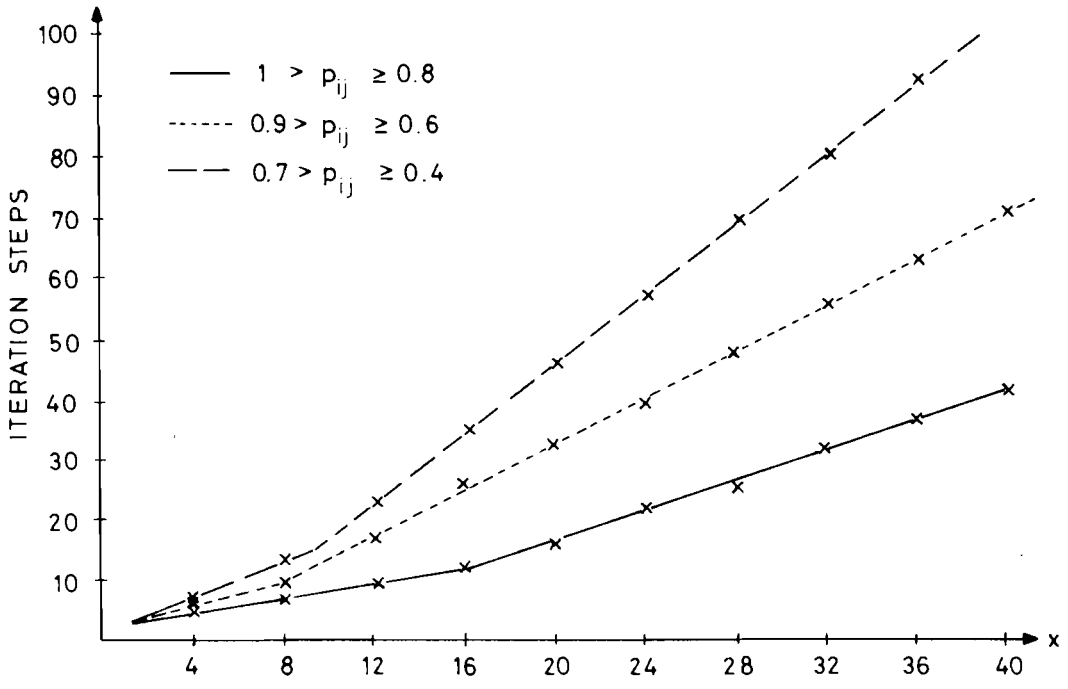

ITERATION STEPS
100
90
80
70
60
50
40
30
20
10
4
8
12
16
20
24
28
32
36
40
x
0.6 > p_ij ≥ 0.3
0.4 > p_ij ≥ 0.1
0.2 > p_ij ≥ 0

FIGURE 4


ITERATION STEPS
100
90
80
70
60
50
40
30
20
10
1 > p_{ij} ≥ 0.8
0.9 > p_{ij} ≥ 0.6
0.7 > p_{ij} ≥ 0.4
4
8
12
16
20
24
28
32
36
40
x

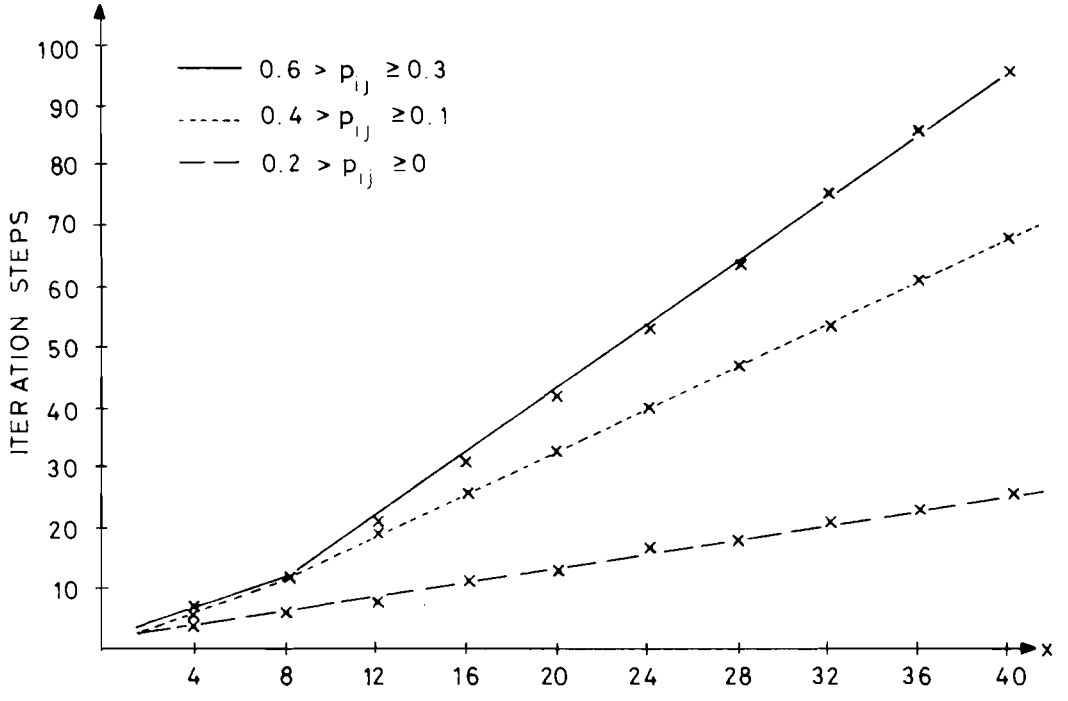

ITERATION STEPS
100
90
80
70
60
50
40
30
20
10
0.6 > p_{ij} ≥ 0.3
0.4 > p_{ij} ≥ 0.1
0.2 > p_{ij} ≥ 0
4
8
12
16
20
24
28
32
36
40
x

FIGURE 5

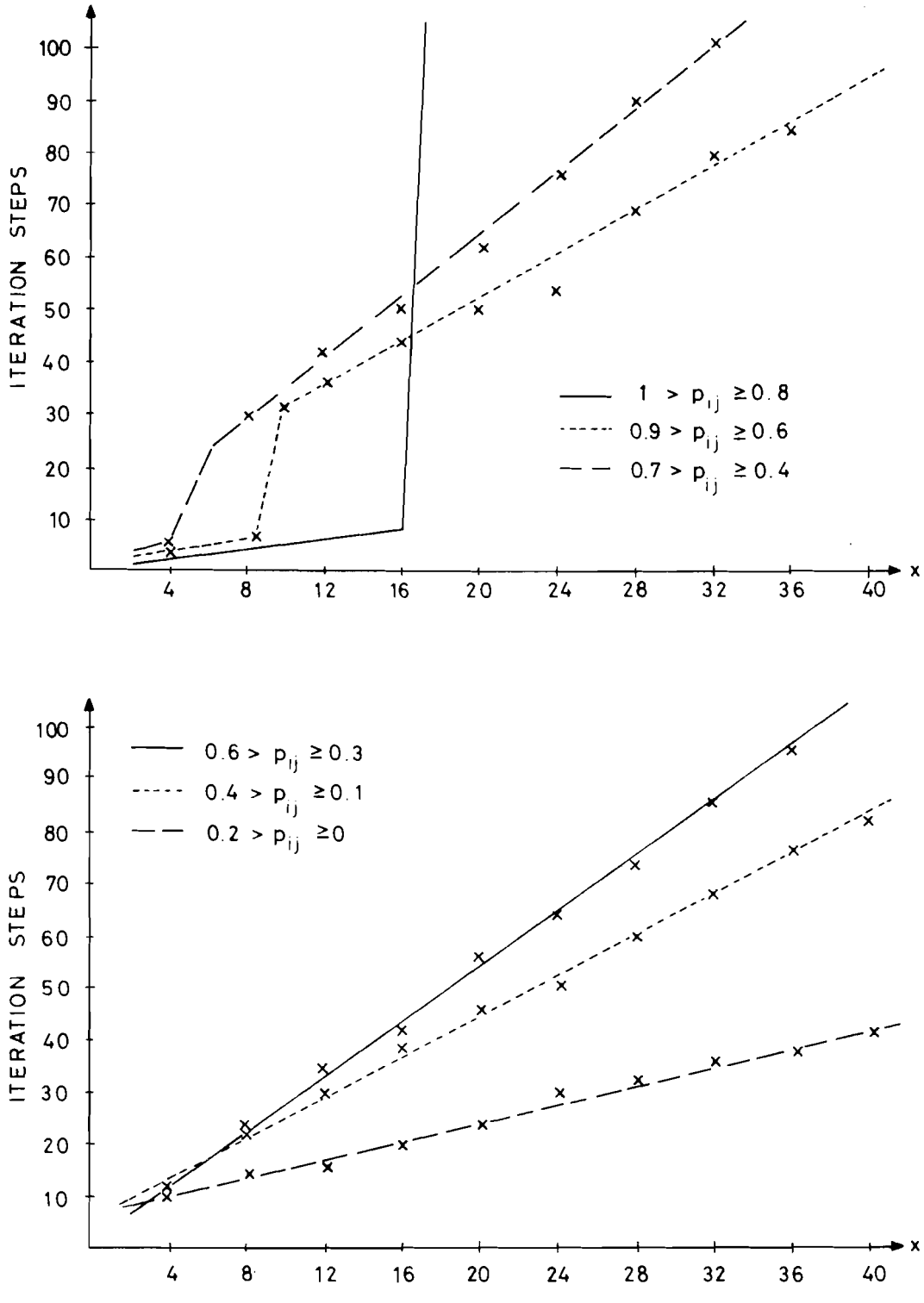
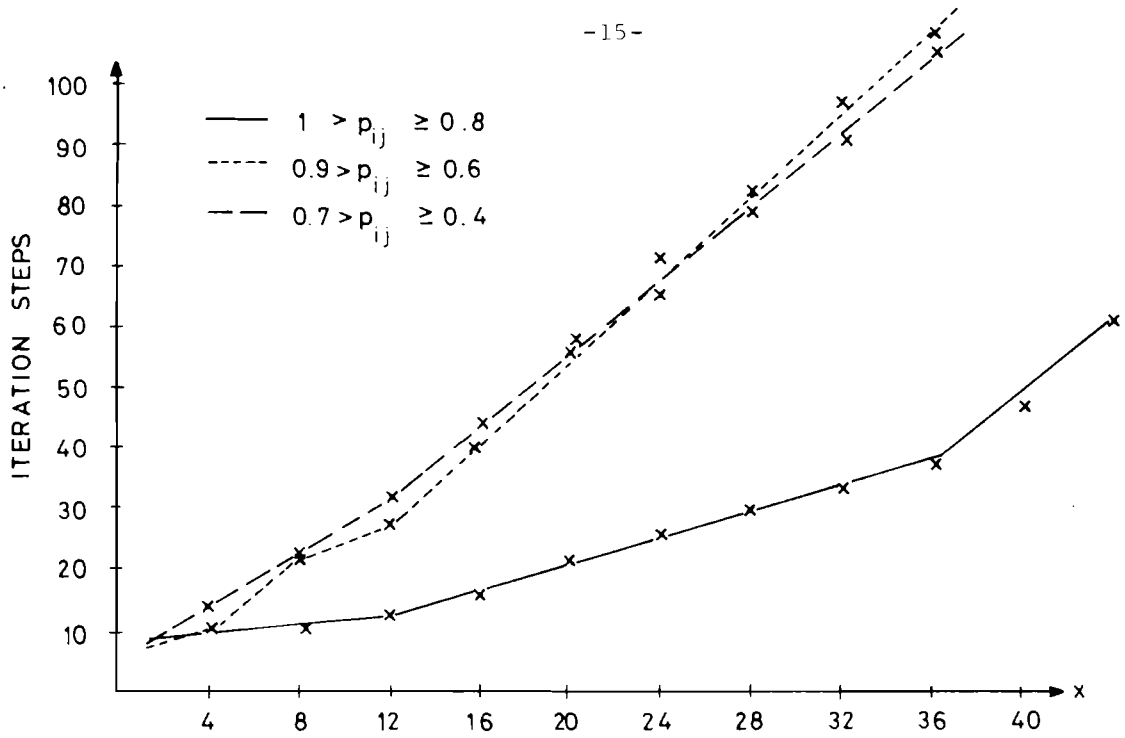

ITERATION STEPS
100
90
80
70
60
50
40
30
20
10
4
8
12
16
20
24
28
32
36
40
x
1 > p_ij ≥ 0.8
0.9 > p_ij ≥ 0.6
0.7 > p_ij ≥ 0.4
0.6 > p_ij ≥ 0.3
0.4 > p_ij ≥ 0.1
0.2 > p_ij ≥ 0

FIGURE 6


ITERATION STEPS
100
90
80
70
60
50
40
30
20
10
1 > p_{ij} ≥ 0.8
0.9 > p_{ij} ≥ 0.6
0.7 > p_{ij} ≥ 0.4
4
8
12
16
20
24
28
32
36
40
x

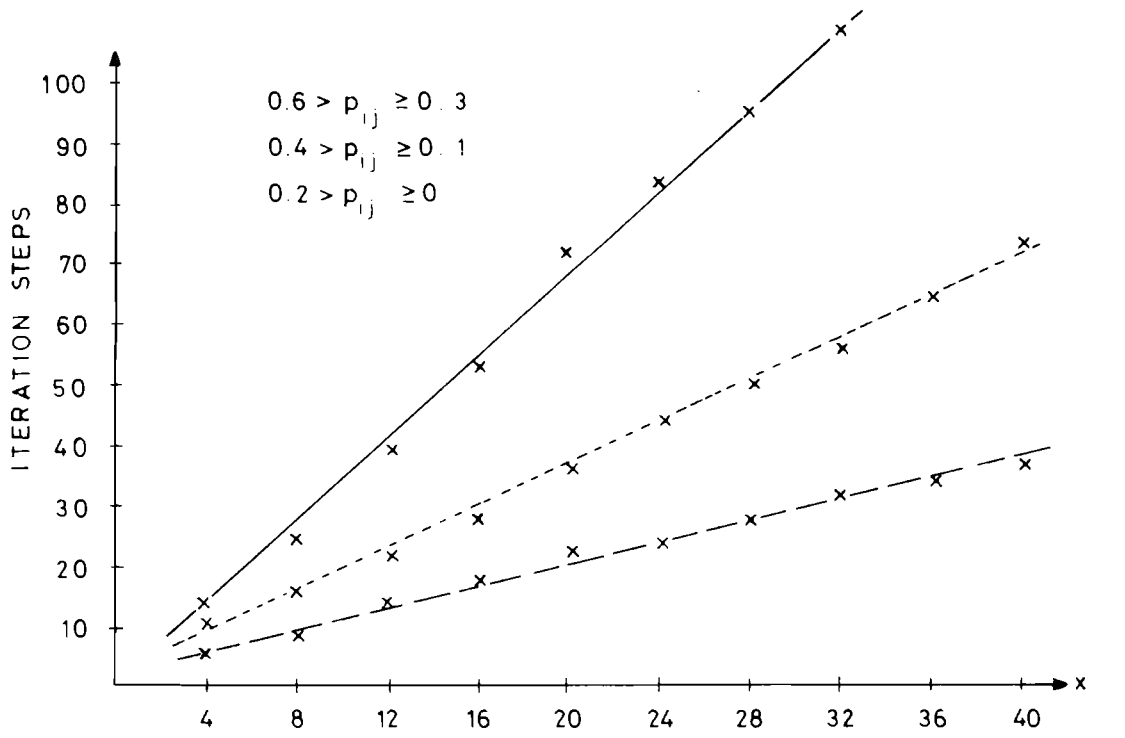

ITERATION STEPS
100
90
80
70
60
50
40
30
20
10
0.6 > p_{ij} ≥ 0.3
0.4 > p_{ij} ≥ 0.1
0.2 > p_{ij} ≥ 0
4
8
12
16
20
24
28
32
36
40
x

FIGURE 7

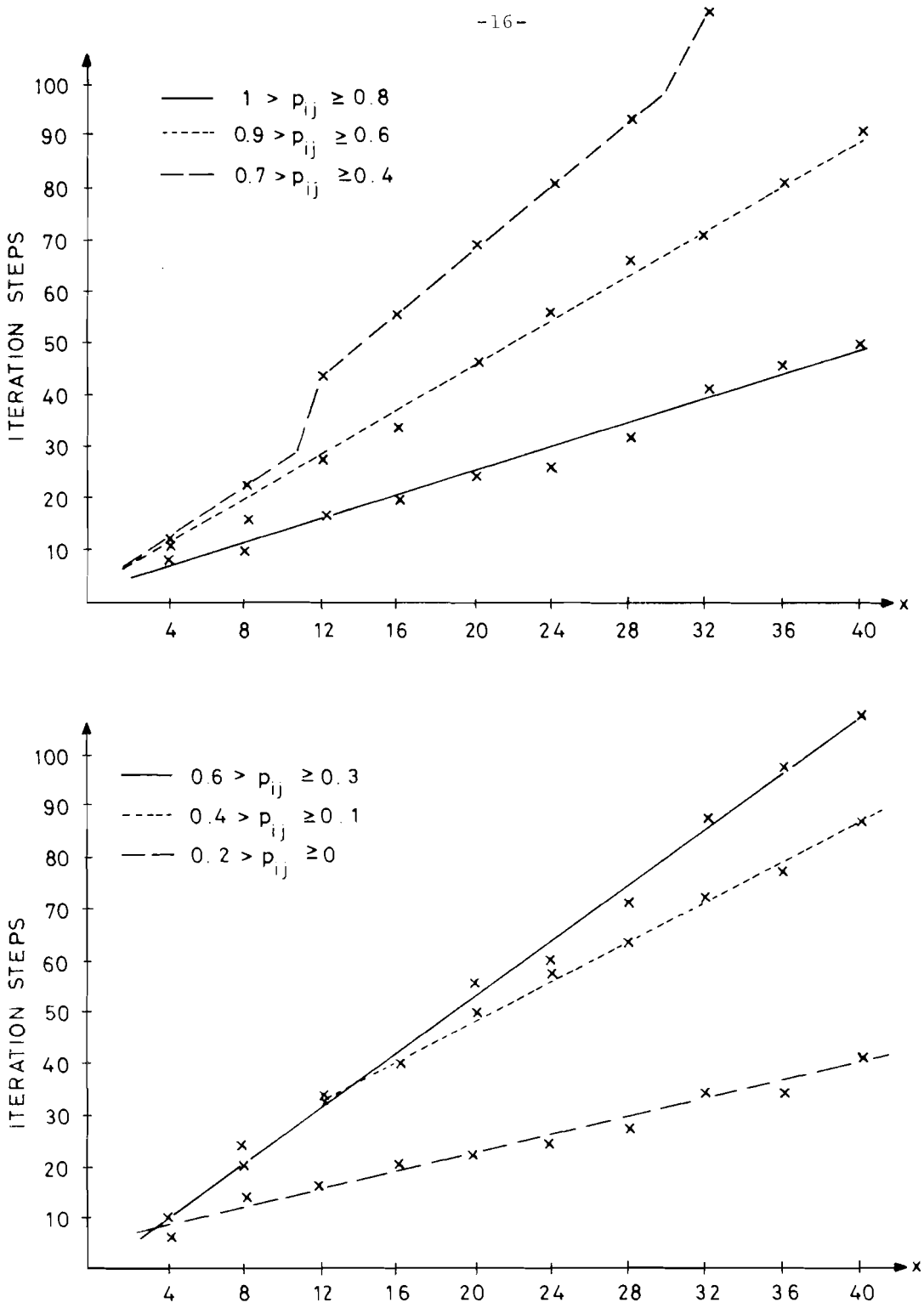

ITERATION STEPS
100
90
80
70
60
50
40
30
20
10
1 > p_ij ≥ 0.8
0.9 > p_ij ≥ 0.6
0.7 > p_ij ≥ 0.4
4
8
12
16
20
24
28
32
36
40
x
ITERATION STEPS
100
90
80
70
60
50
40
30
20
10
0.6 > p_ij ≥ 0.3
0.4 > p_ij ≥ 0.1
0.2 > p_ij ≥ 0
4
8
12
16
20
24
28
32
36
40
x

FIGURE 8

APPENDIX 1

Description of the Implemented Algorithms

This program was used to test the convergence of the algorithm. It began with EPS being 1, and if DIFA becomes less than EPS, then EPS becomes EPS/2 until a maximum of 100 iterations has been done or DIFA becomes equal to zero. To avoid numerical errors, the program was run in double precision and addition was done in an increasing order.

For real time simulation this does not seem to be necessary and thus the program would be faster.

List of the most important variables:

Integers

ITER:	Number of current iteration
ITERM:	Maximum number of iterations
KANAL:	Current channel
KANALA:	Number of the first channel of current node
KANALE:	Number of the last channel of current node
KNOT:	Number of current node
KNOTZ:	Number of current adjacent node
NBKNOT:	Highest node number
NKANAL:	Number of channels
NKNOT:	Number of nodes
NKNOTZ:	Highest number of destination nodes.

Integer arrays

KN(60):	KN(I) number of node at the end of Channel I
M(17):	M(I), M(I) + 1, M(I+1) - 1, numbers of the channels leaving node I.

Reals

PB:	Lower bound of PBLOK if created
PBH:	Upper bound of PBLOK if created.

Real array

PBLOK(60): PBLOK(I) probability of channel I being blocked

Double precision

DOSKN: Addition variable

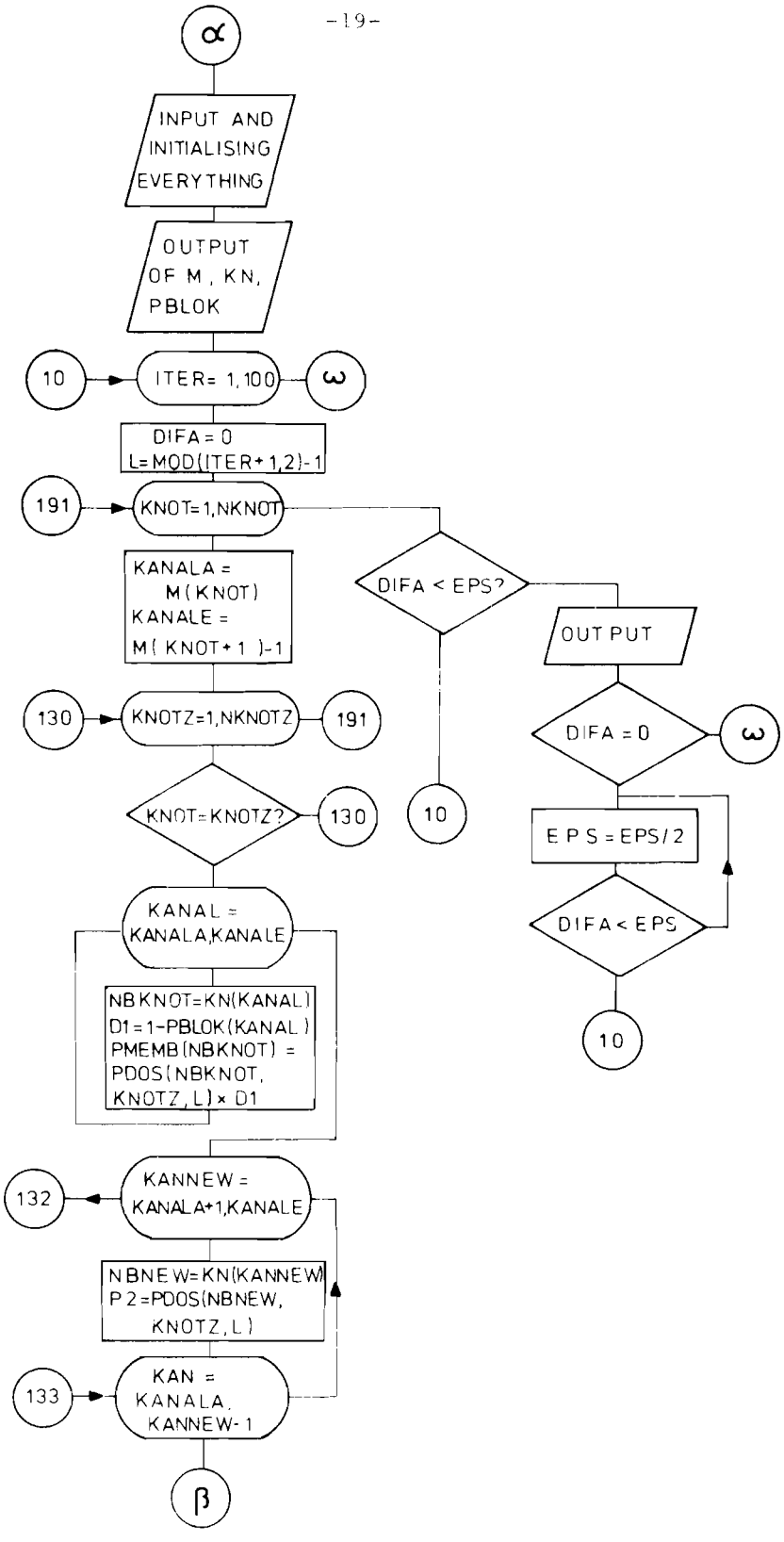
DIFA: Highest difference between old and calculated value of PDOS (node, destination) for all nodes and all destinations

DIF: Difference between old and new value of PDOS for current node and current destination.

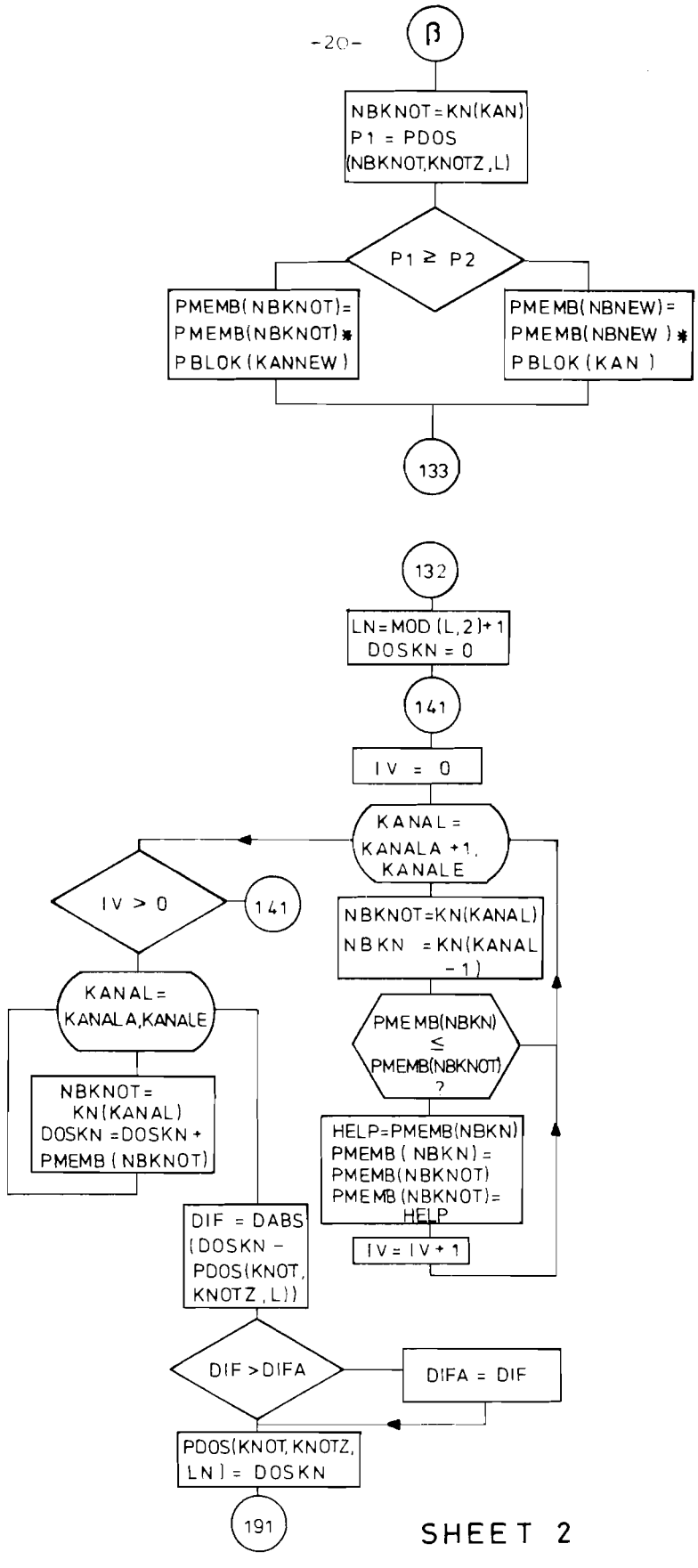
Double precision array

PDOS(16,16,2): PDOS (node, destination, L): probability of reaching destination from node

PMEM B (16): working array.


α
INPUT AND INITIALISING EVERYTHING
OUTPUT OF M, KN, PBLOK
10
ITER= 1,100
3
DIFA = 0
L=MOD(ITER+1,2)-1
191
KNOT=1,NKNOT
KANALA = M(KNOT)
KANALE = M(KNOT+1)-1
130
KNOTZ=1,NKNOTZ
191
KNOT=KNOTZ?
130
KANAL = KANALA,KANALE
NBKNOT=KN(KANAL)
D1=1-PBLOK(KANAL)
PMEMB(NBKNOT) = PDOS(NBKNOT, KNOTZ, L)× D1
132
KANNEW = KANALA+1,KANALE
NBNEW=KN(KANNEW)
P2=PDOS(NBNEW, KNOTZ, L)
133
KAN = KANALA, KANNEW-1
β
DIFA < EPS?
10
OUT PUT
DIFA = 0
3
EPS = EPS/2
DIFA < EPS
10

SHEET 1


β
NBKNOT=KN(KAN)
P1 = PDOS
(NBKNOT,KNOTZ,L)
P1 ≥ P2
PMEMB(NBKNOT)=
PMEMB(NBKNOT)*
PBLOK(KANNEW)
PMEMB(NBNEW)=
PMEMB(NBNEW)*
PBLOK(KAN)
133
132
LN=MOD(L,2)+1
DOSKN = 0
141
IV = 0
KANAL=
KANALA+1,
KANALE
IV > 0
141
NBKNOT=KN(KANAL)
NBKN =KN(KANAL
-1)
KANAL=
KANALA,KANALE
PMEMB(NBKN)
≤
PMEMB(NBKNOT)
?
NBKNOT=
KN(KANAL)
DOSKN =DOSKN+
PMEMB(NBKNOT)
HELP=PMEMB(NBKN)
PMEMB(NBKN)=
PMEMB(NBKNOT)
PMEMB(NBKNOT)=
HELP
IV=IV+1
DIF = DABS
(DOSKN -
PDOS(KNOT,
KNOTZ,L))
DIF>DIFA
DIFA = DIF
PDOS(KNOT,KNOTZ,
LN) = DOSKN
191

SHEET 2

```
RSRED                                          CDC 6600 FTN V3.0-P355 OPT=1  09/12/74

      PROGRAM RSRED(INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT)
      DOUBLE PRECISION HELP,P1,P2
      DOUBLE PRECISION PMEMB,DOSKN,DIF,DIFA,PDOS,EPS,D1
      DIMENSION KN(60),M(17),PDOS(16,16,2),PMEMB(16),PBLOK(60)
      READ(1,811) PB,PBH
  811 FORMAT(2F2.2)
      NKANAL=60
      NKNOTZ=16
      NKNOT=16
      ITERM=100
      READ(1,322) (KN(I),I=1,60),(M(I),I=1,16),(PBLOK(I),I=1,60)
  322 FORMAT (40I2/20I2/16I2/40F2.2/20F2.2)
C
C  BECAUSE ONLY THE NUMBER OF THE ADJACENT NODES  OF NODE I IS AS YET
C  STORED IN M(I) M NOW GETS  NORMALIZED.
C
      DO 320 I=2,NKNOT
      M(I)=M(I-1)+M(I)
  320 CONTINUE
      DO 321 I=1,NKNOT
      M(NKNOT-I+2)=M(NKNOT-I+1)+1
  321 CONTINUE
      M(1)=1
      M(NKNOT+1)=NKANAL+1
      WRITE(2,323)  (M(I),I=1,17),(KN(I),I=1,60)
  323 FORMAT(17(2X,I3)/30(2X,I2)/30(2X,I2)/30(1H*),/////30(1H*))
   40 EPS=1.000
C
C  IF THE VALUES OF PBLOK COME FROME THE INPUT THIS DO-LOOP IS OMITTED.
C
      DO 20 KANAL=1,NKANAL
   22 R=RANF(0.0)
      IF(R-PBH)1,1,22
    1 IF(PB-R) 23,22,22
   23 PBLOK(KANAL)=R
   20 CONTINUE
      WRITE(2,622) (PBLOK(I),I=1,60)
  622  FORMAT (6(10F6.3/)/)
C
C  THE STARTING VALUES FOR PDOS ARE CREATED
C
      DO 193 KNOT=1,NKNOT
      DO 194 KNOTZ=1,NKNOTZ
      IF(KNOT.EQ.KNOTZ) GO TO 195
      PDOS(KNOT,KNOTZ,2)=RANF(0.0)
      PDOS(KNOT,KNOTZ,1)=PDOS(KNOT,KNOTZ,2)
      GO TO 194
  195 PDOS(KNOT,KNOTZ,1)=1.0
      PDOS(KNOT,KNOTZ,2)=1.0
  194 CONTINUE
  193 CONTINUE
C
C  NOW THE ITERATION STARTS
C
      DO 10 ITER=1,100
```

```
      DIFA=0.                                                          GRA00020
      L=MOD(ITER+1,2)+1
      DO 191 KNOT=1,NKNOT
      KANALA=M(KNOT)
      KANALE=M(KNOT+1)-1
      DO 130 KNOTZ=1 ,NKNOTZ
      IF(KNOT.EQ.KNOTZ) GO TO 130
      DO 197 KANAL=KANALA,KANALE
      NBKNOT=KN(KANAL)
      D1=1.000-PBLOK(KANAL)
      PMEMB(NBKNOT)=POOS(NBKNOT,KNOTZ,L)*D1
  197 CONTINUE
      KAN1 = KANALA+1
      DO 132 KANNEW=KAN1     ,KANALE
      NBNEW=KN(KANNEW)
      P2=POOS(NBNEW,KNOTZ,L)
      KANEW1 = KANNEW-1
      DO 133  KAN=KANALA,KANEW1
      NBKNOT=KN(KAN)
      P1=POOS(NBKNOT,KNOTZ,L)
      IF(P1.GE.P2) GO TO 134
      PMEMB(NBKNOT)=PMEMB(NBKNOT)*PBLOK(KANNEW)
      GO TO 133
  134 PMEMB(NBNEW)=PMEMB(NBNEW)*PBLOK(KAN)
  133 CONTINUE
  132 CONTINUE
      LN=MOD(L,2)+1
      DOSKN=0.0
      KAN1=KANALA+1
C
C  THE VALUES OF PMEMB ARE PUTTED IN INCREASING ORDER
C
  141 IV=0
      DO 140 KANAL=KAN1,KANALE
      NBKNOT=KN(KANAL)
      NBKN=KN(KANAL-1)
      IF(PMEMB(NBKN).LE.PMEMB(NBKNOT)) GO TO 140
      HELP=PMEMB(NBKN)
      PMEMB(NBKN)=PMEMB(NBKNOT)
      PMEMB(NBKNOT)=HELP
      IV=IV+1
  140 CONTINUE
      IF(IV.GT.0) GO TO 141
      DO 135 KANAL=KANALA,KANALE
      NBKNOT=KN(KANAL)
      DOSKN=PMEMB(NBKNOT)+DOSKN
  135 CONTINUE
      DIF=DABS(DOSKN-POOS(KNOT,KNOTZ,L ))
      IF(DIF.GT.DIFA) DIFA=DIF                                         GRA00030
  803 POOS(KNOT,KNOTZ,LN)=DOSKN                                        GRA00013
  130 CONTINUE
  191 CONTINUE
   70 CONTINUE
  804 IF(DABS(DIFA)-EPS)      137,137,10
  137 IF(DIFA.EQ.0.000)GOTO 13
```

```
RSRED                                   CDC 6600 FTN V3.0-P355 OPT=1   09/12/74

      EPS=EPS/2.0D0
      IF(DABS(DIFA)-EPS)137,137,10
   10 CONTINUE
      WRITE(2,7) ITER,EPS,DIFA
    7 FORMAT(1X,5(1H*),4HNACH,I5,42H ITER.-SCHRITTEN OHNE ERFOLG BEENDET
     *++EPS=,D11.3,5H DIFA=,D11.3)
   13 WRITE(2,14)PB,PBH,DIFA
   14 FORMAT(/22H PBLOK LIEGT /WISCHEN ,F6.3,4H UND,F6.3,6H******//
     *1X,13(1H*),D40.30,13(1H*))
  707 STOP
      END
```

References

[1] Butrimenko, A. Adaptive Routing Techniques and Simulation of Communication Networks. Fourth International Traffic Congress, Munich, 1970.

[2] Lasarev, W.G. and Parshennkov, N. A Game Method for Dynamic Control of Communication Networks (in Russian). Postroenije Uprovljaushich Ustrojsv e System. "Nauka", 1974.

[3] Butrimenko, A. Routing Techniques for Message Switching Networks with Message Outdating. Symposium on Computer Communication Networks and Teletraffic. Polytechnic Institute of Brooklyn, 1972.

[4] Harary, F. Graph Theory. Addison-Wesley, 1970.