A MATHEMATICAL MODEL OF PATH SEARCHING IN GENERAL TYPE SWITCHING NETWORKS*

By

L. T. Kóczy

Institute of Communication Electronics, Technical University, Budapest

Received September 13, 1980
Presented by Prof. Dr. S. Csibi

Introduction

Let us consider a simple 4-stage folded switching network (Fig. 1.). Path searching has been carried out from the caller switching matrix A1 and the called matrix A4 until stage D, also the adjoining free D–D links have been detected:

In the figure only the free paths running uninterruptedly from A1 to A4 are marked. Now we use the minimum index principle by choosing a "free path":

$$A1-B2-C3-D1-D2$$

from the caller. Now for the called matrix only one possibility remains:

$$D2-C2-B2-A4\,.$$

By this we have chosen link B2–C2 twice, but this fact has not yet been detected. In order to eliminate double selection the algorithm must check all pairs of links between the name stages. After the detection of failure a new "foregoing" path will be chosen:

$$A1-B2-C2-D2-D1$$

The returning path will be now

$$D1-C2-B2-A4\,.$$

Here the same failure occurs.
The next version:

$$A1-B2-C2-D2-D3-C2-B2-A4\,.$$

Failure.

* Lecture submitted to the 9th International Teletraffic Congress

In our very simple example only the fourth version will be sufficient:

$$A1-B2-C2-D2-D3-C3-B2-A4\,.$$

As a matter of course, by using another principle in choosing the paths — e.g. the maximum index principle or random choice — much earlier a "good" path would have been found. It is clear, however, that repeated checking and rejection of paths requires a considerable amount of time.

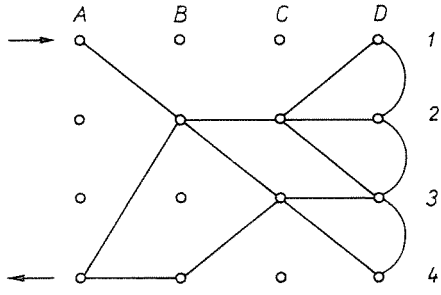


Fig. 1

Let us consider a second example. Two demands of connection are raised successively, in a 4-stage "one direction" network (Fig. 2.; A1 to D2 and A3 to D3.

After the examination of link conditions, for the first connection we choose:

$$A1-B2-C1-D2\,.$$

In the case of Fig. 2. the second demand must be rejected. In the case, however, when first we choose

$$A1-B2-C2-D2\,,$$

there remains a free path for the second connection:

$$A3-B2-C1-D3\,.$$

It is again very difficult to check all alternative possibilities for one or several always existing connections, in order to gain a rearrangement strategy — performing these examinations successively.

In both examples the main problem is common: How to find at least two disjoint paths in the same network, at the same time. (Namely, in folded networks a speach path is always composed of several simple paths, from one end of the network to the other; in the first example of two simple paths.) In the

first part of this paper a transformation rule for mapping the above mentioned problem into a, though more complicated, end-to-end path searching problem, is given.

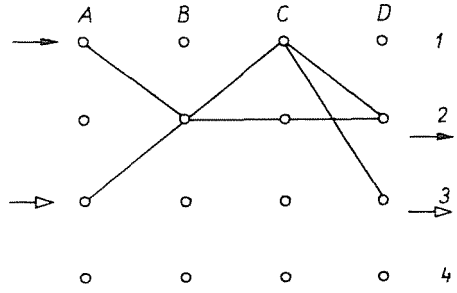


Fig. 2

Transformation of multiple path searching tasks

Immense time consumption of the algorithms sketched in the above section was caused by the impossibility of immediately recognizing double selection of links.

For the elimination of the problems we introduce a mapping on the structure of networks, which results in a more complicated network but always in a very simple path searching task.

Definitions

1. *Switching Network Graph (SNG)*

A graph is named a switching network graph if there is a subset of its nodes (start nodes) so, that

a. there is no path of length one between any pair of these nodes
b. there exists another subset of nodes (termination nodes) so, that there is at least one continuous path between an arbitrary pair when the elements are chosen from both subsets (it is not necessary that these subsets be disjoint or different at all.)

2. *Simple Switching Network Graph (SSNG)*

A graph is named a simple SNG if there exist two disjoint subsets of nodes fulfilling condition 1a and these two fulfilling 1b.

3. *Switching Path Bundle (SP)*

In a SNG we name a set of paths switching path bundle, if all elements of the set are continuous paths from a start node to a termination node, and the elements are disjoint.

4. *Simple switching path (SSP)*

A SP in a SSNG is a simple SP, if the number of paths is one.

Statement A

If given a SNG G, there exists an invertible mapping M, so, that $G^* = M(G)$ *is* SSNG and for all SP-s P in G if the number of elements in SP is less than a given N, $P^* = M(P)$ is SSP in G^*. We omit the proof, as it has no practical importance in the general case.

Now we establish the concrete implementation of the above statement for the two important problems mentioned in the Introduction.

5. *Staged one-direction SSNG*

If the nodes of a SSNG can be partitioned into $n > 2$ subsets $S_1, S_2, \ldots, S_n / S_i \cap S_j = \emptyset$ if $i \neq j$, $\bigcup_{1=1}^{n} S_i = G$, the set of graph nodes, and there is no edge between any pair of nodes belonging to the same S_i, and S_1 is the set of termination nodes; the graph is named a staged one-direction SSNG (DG).

Statement B

If given a DG G, there exists the mapping M, according to statement A, so that G^* is also DG, $n^* = n$, and if $m_{i,j}$ $(i = 0, \ldots, n-1; i < j \leqq n)$ are the numbers of links between stages S_i^* and S_j^*,

$$m_{i,j}^* = \binom{m_{i,j}}{N}$$

(N as in definition 3.)

Proof

Let us construct S_i^* in G^* in the following way:

If

$$\# S_i = \tilde{S}_i,$$

then

$$\# S_i^* = \binom{\tilde{S}_i + N - 1}{N};$$

and

$$M(M_{i,k_1}, M_{i,k_2}, \ldots, M_{i,k_N}) = M^*_{i,k_1,k_2,\ldots,k_n} \quad (0 \leqq k_1, k_2, \ldots, k_N \leqq \tilde{S}_i)$$

and

$$M(e_{ij,c_1}, e_{ij,c_2}, \ldots, e_{ij,c_N}) = e^*_{ij,c_1c_2,\ldots,c_N} \quad (e_{ij,c_p} = e_{ij,c_r} \text{ if } p \neq r)$$

where $(e^*_{ij,c})$ is an edge between elements of S_i and $S_j (j > i)$,

$$e^*_{ij,c_1,c_2,\ldots,c_N} \text{ is an edge}$$

between S_i and S_j.

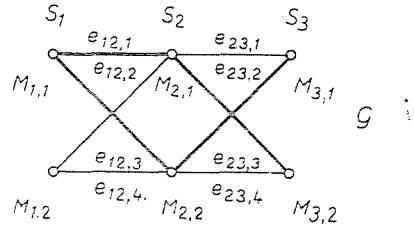


Fig. 3

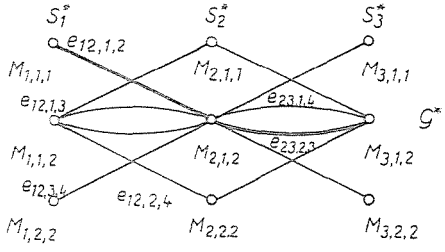


Fig. 4

So to all "cuts of SP-s" between S_i and S_j there exists one and only one link between S_i^* and S_j^*.

Let us examine an example where $n = 3$, $\tilde{S}_i = 2$ for $i = 1, 2, 3$ and $N = 2$. The structure is given in Fig. 3. Then G^* is to be seen in Fig. 4.

We pointed out a SP in G and its equivalent SPP in G^*.

6. *Staged folded SSNG*

If a SSNG is partitioned as in def. 5 and there is no edge between any pair of nodes belonging to the same S_i $(i = 1, \ldots, n-1)$, but there is at least one edge to any element of S_n going to an (not necessarily different) element of S_n and S_1 is the set of start and termination nodes; the graph is named a staged folded SSNG (FG).

Statement C

If given a FG G, there exists the mapping M according to statement A, so that S^* is DG, $n^* = n$, and if $m_{i,j}$ $(i = 0, \ldots, n-1; 1 < j \leqq n, i \neq j)$ are the numbers of links between stages S_i and S_j; the numbers of links between stages S_i^* and S_j^* $(i \neq j)$ are

$$m_{i,j}^* = \binom{m_{i,j}}{2N}$$

(N as in definition 3), and $\tilde{S}_n^*$ is equal to the number of N-tuples of different pairs in S_n which are linked by S_nS_n type edges.

Proof

Let us construct S_i^* in G^* in the following way:

If

$$\# S_i = \tilde{S}_i,$$

then

$$\# S_i^* = \binom{\tilde{S}_i + 2N - 1}{2N} \qquad (i \neq n)$$

and

$$M(M_{i,k_1}, M_{i,k_2}) = M_{i,k_1,k_2} \qquad (0 \leqq k_1, k_2 \leqq \tilde{s}_i)$$

and

$$M(e_{ij,c_1}, e_{ij,c_2}) = e_{ij,c_1,c_2}$$

$$(e_{ij,c_p} \neq e_{ij,c_r} \text{ if } p \neq r; \quad i \neq j)$$

$$M(e_{ij,c_1,c_2}) = \text{“no edge”}$$

So to all “cut pairs of SP-s” between S_i and S_j there exists one and only one link between S_i^* and S_j^*.

An example is shown in Figs. 5 and 6, $n = 3$, $\# S_i = 3$ for $i = 1, 2, 3$ and $N = 1$.

A folded path and its equivalent is shown on the figures.

Conclusion 1

By using the practical version B and C of Statement A we have obtained a way of transforming a very general class of switching networks and path searching problems into a quite simple network and problem. By giving, however, the number of nodes and edges (virtual switching matrices and links)

we pointed out the difficulties of this method. When considering a simple n-stage network with a constant number of machines in a stage(s), the original value ns grows by the transformation to about

$$n\binom{s+1}{2} = ns(s+1)/2$$

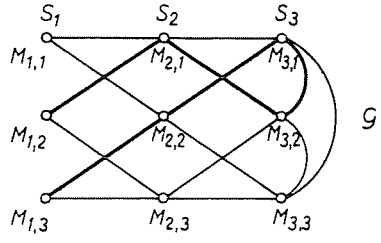


Fig. 5

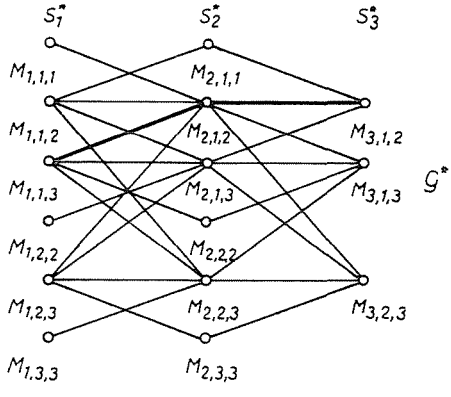


Fig. 6

Because of considerations based on the statements of the second part of this paper, more important is the growth of the number of links. Let us assume, that in the above network, there are t links between two adjoining stages. Now their total number

$$(n-1)t \text{ grows to}$$

$$(n-1)\binom{t}{2} = (n-1)\,t\,(t-1)/2$$

by the transformation.

Modelling stages by incidence matrices

In the first part the problem caused by the complicatedness of path or path bundle was discussed. Now we concentrate on the irregularity of networks, but we consider only DG-s and SSP-s. The statements in the first part allow this restriction.

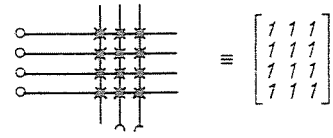


Fig. 7

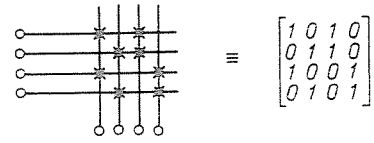


Fig. 8

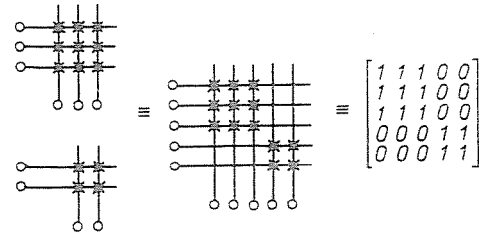


Fig. 9

What are the main features of an irregular network?

1. The dimensions of the switching matrices (SM) are different even in the same stage.
2. Not all crosspoints exist really (e.g. because of failures this can be time-variant).
3. Some links connect not adjoining stages. Such links are called passing links.

In our algorithm we model SM-s by binary incidence matrices. A simple $n \times m$ SM is modelled by a $n \times m$ matrix full with $1-s$ (figure 7).

If in a SM there are empty places of crosspoints, there we put 0-s in the matrix (Fig. 8).

Now the question arises, how to manage different SM-s belonging to the same stage together. It means no difficulty, if we merge two SM-s into one in the way, that the resulting SM has as many inputs and outputs as the two original ones together, and there are no crosspoints on the places where an input of the first SM meets an output of the other (figure a). So, with the help of the above method (Fig. 9), they can be modelled together by one matrix.

By using this idea an arbitrary number of SM-s can be merged into one matrix. What about passing links? We assume on each passing link 1×1 virtual SM-s in each stage, where the link passes through.

By this method an arbitrary SSNG with n stages can be modelled by n matrices (and the link structure), each of them belonging to one stage.

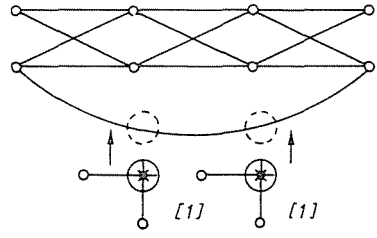


Fig. 10

Path searching in SSNG-s

Let us consider now one stage with m inputs and n outputs. The input and output link bundles can be modelled by state vectors, where 1 means a free, 0 a busy link.

It is easy to see, that if a link state vector f_{i-1} and the stage matrix F_i are given,

$$f_i' = f_{i-1} \circ F_i$$

results in a vector, which features the possibilities to reach the links between the ith and $(i+1)$th stages — taking into consideration only the state of the previous link bundle and the structure of SM-s in stage i. The symbol "$\circ$" stands for the conjunctive composition of

$$[a_{ij}] \circ [b_{jk}] = [c_{ik}],$$

$$c_{ik} = \bigvee_j (a_{ij} \wedge b_{jk}).$$

A SSNG consisting of S stages is given, the stages featured by matrices are F_1, F_2, ..., F_s. The state of the network is given by the link state vectors f_1, ..., f_{s-1}. A SSP searching task can be defined by vectors f_0 and f_s, the indicator vectors of possible starting and termination points in the network (1 indicates "good" points). Then a complete free path map can be gained by the following algorithm:

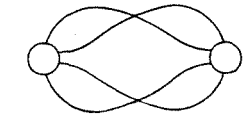
$$p_0^T = f_0^T$$

$$p_i^T = (p_{i-1}^T \circ F_i) \wedge f_i^T \qquad i = 1, \ldots, s$$

where "$\wedge$" stands now for vectorial conjunction:

$$[a_i] \wedge [b_i] = [a_i \wedge b_i] .$$

As we stated above, $p_{i-1} \circ F_i$ is the vector of reachable links in stage i, it must be, however, masked by the state of these links f_i. There is still another problem.



$$p_i = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Fig. 11

The sequence of the same links considered as outputs of the ith stage and as inputs of the $(i+1)$th stage is not the same. So their suitable permutation must be produced by permutation matrices (Fig. 11).

As the link permutation matrices are structure determined similarly to the stage matrices, it is an obvious aim to merge them pair by pair.

For permutation matrices conjunctive composition and matrix multiplication are identical, so we can write:

$$f_i^T = (f_{i-1}^T p_i) \circ F_i = (f_{i-1}^T \circ p_i) \circ F_i = f_{i-1}^T \circ (p_i \circ F_i) = f_{i-1}^T \circ I_i$$

Here we assumed the associativity of conjunctive composition, which property can easily be proven:

$$([a_{ij}] \circ [b_{jk}]) \circ [c_{kl}] = [\bigvee_j (a_{ij} \wedge b_{jk})] \circ [c_{kl}] =$$

$$= [\bigvee_k ((\bigvee_j) a_{ij} \wedge b_{jk}) (\wedge c_{kl})] = [\bigvee_k (\bigvee_j (a_{ij} \wedge b_{jk} \wedge c_{kl}))] =$$

$$= [\bigvee_{j,k} (a_{ij} \wedge b_{jk} \wedge c_{kl})] = [\bigvee_j (\bigvee_k) b_{jk} \wedge c_{kl} (\wedge a_{ij})] =$$

$$= [a_{ij}] \circ ([b_{jk}] \circ [c_{kl}])$$

Statement of this part of the paper is *Statement D.*

If given a SSNG G with structural matrices I_i belonging to the subsets S_i ($i = 1, \ldots, s$), where I_i are gained from $P_i \circ F$ if $i > 1$, and $I_1 = F_1$, and P_i are the permutation matrices to ordering the sequence of F_{i-1} columns so, as to be in accordance with that of the lines in F_i, finally F_i are the incidence matrices

indicating with elements 1 the meeting of a line and a column if the links (edges) corresponding to them are allowed to participate in the same SSP, furthermore given are the mask vectors f_i $(i>1)$, with elements 1 corresponding to links allowed to participate in an arbitrary SSP; all solutions for the path searching problem given by vector p_0, where a path is searched to S_n, are determined by the algorithm:

$$p_0 \quad \text{given},$$

$$p_i^T = (p_{i-1}^T \circ I_i) \wedge f_i^T; \qquad i=1, \ldots, s$$

where p_i indicates the links taking part in a path by elements 1.

The proof is obvious from the above considerations. An example is shown in Fig. 12.

It is obvious, that the analogue of statement D: namely, the solution of the problem when the indicator vector p_s' for the last stage is given and paths are searched from arbitrary elements of stage 1 is also true; then

$$p_s' \quad \text{given},$$

$$p_i' = f_i \wedge (I_{i+1} \circ p_{i+1}') \qquad i=1, \ldots, s$$

Statement E

At the conditions of statement D, subsets of S_1 and S_s are given by the indicator vectors p_0, p_s; then all solutions for the path searching task are given by:

$$w_0 = p_0 \wedge (I_1 \circ p_1')$$

$$w_1 = (p_0^T \circ I_1)^T \wedge (I_2 \circ p_2') \wedge f_1$$

$$w_2 = (p_0^T \circ I_2)^T \wedge (I_3 \circ p_3') \wedge f_2$$

$$\vdots$$

$$w_s = (p_{s-1}^T \circ I_s)^T \wedge p_s'$$

going on with the example in Fig. 12, let $p_3'^T$ be [010]. Then first we get $p_2' \ldots p_0'$ and finally $w_0 \ldots w_3$ as visible in Fig. 13.

Conclusion 2

By statement E we have an algorithm for parallelly finding all paths in a SSNG. The algorithm operates with binary matrices and so with logical operations. Even so, the algorithm requires a high number of operations, i.e.

much time. When the original problem is more complicated and we must use the transformation of statement A, the dimensions of stage matrices grow even more. This is a serious difficulty of practical application.

It is suggested to use a special hardware besides the control processor ensuring fast matrix operations. Such hardware equipment does exist, as e.g.

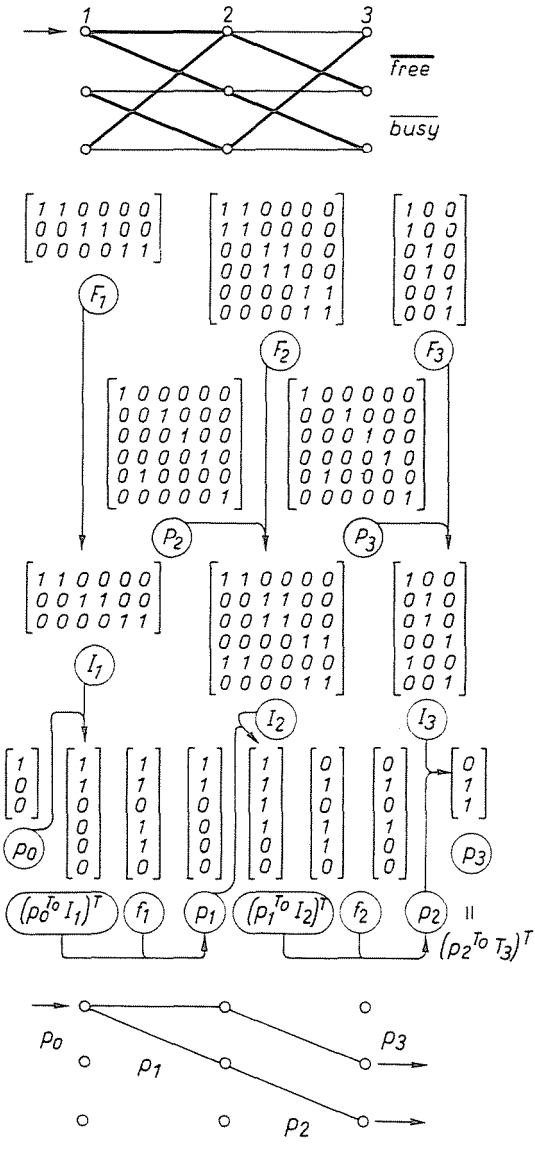


Fig. 12

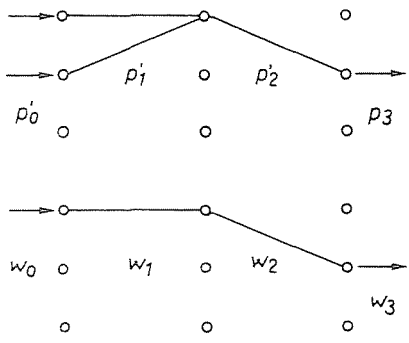


Fig. 13

cellular automaton fields. In any case, however, we have given a complete model of path searching, adequate for a very general class of switching networks.

Summary

In the stored program control of switching networks, one of the most interesting algorithmic problems is the full examination of path conditions at a given time point and state of the system. Although theoretically, there exists no problem, it is impossible from the practical point of view to examine sequentially all paths in consideration, as the time consumption of this method is enormous. In most existing systems this problem was solved by developing switching networks of specific structure suitable for fast path searching.

It is, however, sometimes impossible to satisfy such conditions, because of other reasons, as e.g. traffic considerations. Moreover, these fast algorithms are heuristic and suitable only for a given type of network, and a given problem group. There is, e.g. the question: is there a possibility to rearrange existing speach connections so, that a new connection could be established in a virtually blocked network?

This paper intends to give a mathematical model that allows a switching network of arbitrary structure and n-fold path searching parallelly. We are aware of the fact, that even a compact algorithm able to solve such a universal task needs much time and memory capacity, so using conventional control processors real time functioning is impossible. There exist, however, such special hardware facilities that connected to one of several conventional processors, form an able control system.

dr. László T. Kóczy H-1521 Budapest