Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

PISA

ON THE FAST SYNCHRONIZATION OF TREE CONNECTED NETWORKS

F. Romani

Nota Interna B76-5
Marzo 1976

ON THE FAST SYNCHRONIZATION OF TREE CONNECTED NETWORKS

Francesco Romani

ABSTRACT

In extending the Firing Squad Synchronization problem to the case of tree connected networks and applying the controlled synchronization of linear arrays we obtain the optimal time for trees with a bounded number of nodes.

We also investigate methods for synchronizing general trees and show how to apply them to networks arbitrarily connected.

1. INTRODUCTION

In a previous paper [7] it is shown how tree synchronization is relevant in obtaining easy and fast synchronization of networks however connected. In this paper we study optimal tree synchronization techniques.

Let us outline how the network synchronization problem is stated [1-7].

We consider a set of n identical finite state machines: at time t each of them enters a new state that depends on the states at time t-1 both of the automaton itself and of a finite number of some other automata called its neighbours, arbitrarily chosen. The neighbourhood relation is supposed symmetrical.

We wish to define the single automaton structure so that the network will behave as follows: at time t=0 all the automata are in a quiescent state S called Soldier, except one that is in an active state G called General. After a sequence of transitions, at time t=f all the automata enter, for the first time, the state F called Fire.

In [7] we have shown that, by using tree synchronization techniques, it is possible to synchronize networks however connected, and that the optimal synchronization time for trees is d+r where d stands for the diameter (i.e. the maximal distance between two automata in the network) and r for the radius (i.e. the maximal distance between the General and another automaton in the network).

On the other hand our methods for tree synchronization have obtained computation times of 2n-2 and p+q , where n is the

number of automata in the tree, p and q are respectively the sizes of the largest and second largest subtrees departing from the General.

These times are not optimal and for particular trees they are much larger than d+r. E.g. in the binary tree of fig.1 the radius is r, the diameter is 2r and the number of automata is 2**(r+1)-1.

We describe in section 2 the controlled synchronization of linear arrays and star graphs by using the results of Rosenstiehl and Waksman; in section 3 the techniques to find radial and diametral paths in a tree; in section 4 an optimal synchronization method for trees with two nodes (we call node an automaton with more than two neighbours); finally, in section 5, the optimal synchronization for any tree and the applications of such techniques to networks however connected.

2. THE CONTROLLED SYNCHRONIZATION OF LINEAR ARRAYS AND ITS APPLICATION TO STAR GRAPHS.

In fig.2 we see an example of the Waksman synchronization method for linear arrays [1]; this method is optimal (i.e. it reaches the firing status in 2n-2 steps).

Furthermore the Rosenstiehl freezing-unfreezing technique [6] can be used to delay the synchronization of a linear array by an arbitrary interval (see fig.3).

One of the major problems in star graphs and tree synchronization is how to find the exact instant to unfreeze a frozen ray. Our solution is a slightly modified version of

Rosenstiehl's method.

In fig.4 we see how to divide a linear array into two equal rays. Thus it is possible to synchronize optimally a linear array with the General in any position (fig.5). We use two Firing Squads AH and HB; HB is synchronized in the Waksman mode, AH is frozen and unfrozen by the synchronism signals.

In fig.6-a we see how to freeze and unfreeze also a secondary ray, by using a slow signal (v=1/3), a marker (v=0) and a fast signal (v=1). This method also operates for secondary rays not departing from the General (fig.6-b).

In fig.7 we see an example of star graph synchronization useful to familiarize ourselves with fast tree synchronization techniques. We note that the diametral path (i.e. the two longest rays) is synchronized in an optimal way, and simultaneously a set of synchronism signals is used to delay the secondary ray synchronization by the intervals required.

Our method to synchronize a tree is based on a similar principle: to synchronize in an optimal way the diametral path and to freeze and unfreeze all the secondary rays in order to obtain the firing state at the same time in each automaton.

For this purpose we have developed methods to search for diametral and radial paths in a tree connected network.

3. RADIAL AND DIAMETRAL PATHS IN A TREE CONNECTED NETWORK

We call radial path a minimal path of length r departing from the General; diametral path a minimal path of length d.

Now we describe a method for finding a radial path in a tree

connected network of finite automata; though this method is not used in the tree synchronization, it is useful to understand the subsequent techniques dealing with trees.

The General sends fast signals (velocity = 1) into each subtree departing from it. These signals go toward the ends of the tree, duplicating themselves at nodes, thus entering each ray of the tree. When a signal reaches an end it is reflected and comes back toward the General at the same unitary velocity. At every node only the last expected signal proceeds toward the General and sets a switch to record the ray it came from while the other signals are suppressed. Note that an automaton in a node is capable to count the expected signals. In fact in a network of finite automata, the number of neighbours of each automaton is bounded a priori.

The last signal reaches the General after being reflected at the farthest end, thus passing through a radial path; then is again reflected by the General and can mark the radial path by using the set switches (see fig.8).

The second problem we consider is how to find a diametral path in a tree. We use a preliminary theorem whose proof is given in appendix 1:

"For a tree T, a diametral path is contained in the following subgraph

1) the radial path GA;

2) the second radial path GB (i.e. the radial path of the tree obtained by suppressing the subtree which contains A);

3) every radial path of the subtrees departing from GA (taking as radix the node automaton)."

Therefore we can consider a reduced figure (fig.9). Suppose that at time t=0 the General sends everywhere a fast signal (v=1) which is duplicated at each node and reflected at the ends as for the previous algorithm. Thus the return signals at the nodes have the following transit times.

from B to G t=2|GB|

from A to H t=|GA|+|AH|

from C to H t=|GC|+|CH| etc..

Let us suppose also that the return signal from B is suppressed when reaching G and a slow signal (v=1/3) is generated toward K and H. Therefore the arrival time from D to K is |GD|+|DK| and that from G to K is 2|GB|+3|GK|. Whence

tDK>tBK => |GD|+|DK|>2|GB|+3|GK| =>

=> |GK|+2|DK|>2|GB|+3|GK| => 2|DK|>2(|GB|+|GK|) => |AD|>|AB|

=> |DK| > |GB|+|GK| => |AK|+|KD| > |AK|+|KG|+|GB|

therefore AB is not a diametral path.

Obviously if tDK<tBK then |AD|<|AB| and AD is not a diametral path. In case |AB|<|AD| in order to test in a similar way whether |AC|<|AD| it is suitable to suppress the slow signal and originate another slow signal when the return from D reaches the node K. Analogously in the second case in order to test whether |AC|<|AB| it is suitable to make the slow signal pass along the HK branch.

Now we can give the algorithm to find a diametral path in a tree. The General sends everywhere a fast signal which is propagated and reflected as in the radial path algorithm. At each node only the last expected signal passes toward the General and sets a switch.

The last but one signal that reaches the General is changed into a slow signal which is sent along the ray whence the last return is expected.

When the slow signal reaches a node two cases may arise:

1) that node was reached by all but one of its return signals then the slow signal proceeds along the ray whence we expect the last return and sets the switch;

2) that node is waiting for more than one signal, then the slow signal is suppressed and the last but one signal, when arriving, is changed in a slow signal that sets the switch and goes along the ray whence we expect the last return.

When the slow signal and a return signal meet in a ray they are suppressed while two fast signals are generated in either way to mark the automata of the diametral path by following the previously set switches. An example of how this method operates is seen in fig.10.

If two signals meet at a node then two paths have the same length and either one can be selected arbitrarily (e.g. according to priority) thus selecting one of the possible diametral paths. Analogously for more than two simultaneous signals at a node.

4. OPTIMAL SYNCHRONIZATION OF TWO-NODE TREES

In fig.11-a we see a two-node tree (2N for short) with the General at one node. 2N trees differ from star graphs only in having one more node in a ray.

To synchronize such a network optimally it is necessary

1) to find a diametral path;

2) to freeze the secondary rays;

3) to synchronize the diametral path optimally and to unfreeze the secondary rays at the exact instants.

Note that searching for a diametral path has to be embedded in the synchronization process in order to obtain a minimal time algorithm.

The process begins with the General sending fast signals toward the ends as in the previous algorithms. When these signals reach an end they start the Waksman synchronization process of that ray and are reflected toward the General.

At the General and at the other node each return signal but the last one causes the generation of the appropriate synchronism signal and the freezing of the corresponding ray which is a secondary one and does not belong to the diametral path.

For the remaining part of the tree (fig.11-b), now our problem is to select in real time the diametral path: is it ANB or ANGC?. This decision is to be taken by the N automaton.

The General, when receiving the return signal from C sends a slow signal (v=1/3) toward N.

If the return signal from NB arrives before the slow signal from G then NB does not belong to a diametral path and it can be frozen. Otherwise if the return signal from B arrives after the slow signal then NB belongs to the diametral path and NGC is to be frozen. But the freezing of NGC is to be initiated when the fast synchronization signal from G reaches N, that is before the slow signal arrives.

To solve this problem we use a more complex synchronization automaton composed of one control automaton C and two Waksman synchronization automata W1,W2 (as known it is possible to build a finite state machine capable of simulating a bounded number of finite state machines). We define as firing state of the composite automaton any state where at least one Waksman automaton is in the firing state.

The synchronization proceeds as long as possible in both the Waksman automata, when the fast synchronization signal from the General reaches the node N the W1 Firing Squad is frozen and N becomes a virtual end for it also the synchronism signals are generated while the W2 Firing Squad proceeds in its synchronization.

When the slow signal from G reaches N it is possible to decide which of either Firing Squad is suitably active in order to deactivate the wrong synchronization process, thus the entire synchronization will terminate on the other set of Waksman automata. In fact the diametral path thus found is synchronized with the Rosenstiehl method (slightly modified), and the other rays are unfrozen at the exact instants (for an example see fig.12).

Let us now prove that this synchronization algorithm is optimal (i.e. it reaches the firing state with d+r steps). We distinguish two cases: if the General is on a diametral path (say AC as in fig.11.b) d+r is the minimal time required to synchronize AC and the diametral path searching process causes no delay. Otherwise r' steps are required to start the synchronization process of the diametral path, where r' is the

distance between the General and the nearest automaton on the diametral path. Then the diametral path is synchronized in d+r" steps, where r" is the distance from N to the farthest end of the diametral path. Therefore the overall synchronization process requires d+r"+r' steps, but is easy to see that r'+r"=r and also in this case d+r is the syncronization time for the tree.

5. FAST SYNCHRONIZATION OF GENERAL TREES AND APPLICATION TO CONNECTED NETWORK SYNCHRONIZATION

Our optimal synchronization method was described in the case of 2N trees for simplicity sake. It is possible to extend it method to n-node trees but n has to be known when designing the single automaton.

In fact the synchronization of each frozen ray is controlled by synchronism signals that travel in a longer ray, then in the diametral path there merge all the synchronism signals from the rays terminating in it, therefore the number of states has to be great enough to distinguish each signal, so that it is possible to single out both the node and the ray from which the signal came. To avoid ambiguities the number of states in the single automaton has to depend on the number of nodes.

For any n the method of optimal synchronization appears to be just an extension of the 2N case, and we think more interesting to investigate a fast synchronization method for any tree by applying a n-node method.

The synchronization method now described for any tree is

faster than other general tecniques for tree synchronization (e.g. circular paths or flower graph methods [7]).

Assume an automaton Z capable of synchronizing a n-node tree in optimal time; we combine v automata of the Z type and one control automaton (v is the upper bound for the number of neighbours in the tree).

It is easy to prove that, if we put a bound on the number s of nodes in a ray then the total number of nodes is also bounded. Therefore for any pair n,v it is possible to find s such that the number of nodes in the tree is lower than n.

To start the process the General sends everywhere fast synchronization signals by using one only of the Z automata. In each cell the control automaton counts the number of nodes on that ray and, as long as this number is less than s, forwards the process on one Z automaton only (we remark that the method is also optimal when the tree is small enough). Otherwise at the (s+1)-th node the control automaton starts a ring reduction process of the subtree departing from it, by using as many Z automata as necessary. In this case the synchronization signals meet at the center of the ring and divide it into two linear arrays of the same length which are synchronized simultaneously as an extension of the ray they depart from (see fig.13).

This method is a generalization both of the circular path method which uses the synchronization of linear arrays (i.e. 0-node trees) and of the flower graph method which uses that of star graphs (i.e. 1N trees).

The synchronization time is $d'+r'$, where d' and r' are the diameter and the radius of the k-node tree ($k \leq n$) obtained in

the reduction process. We have $d \leqslant d'$ and $r \leqslant r'$.

Now we see how this technique is applicable to the problem of synchronizing networks arbitrarily connected.

We use a reduction process similar to those explained in [7], the only difference being that now we want to reduce the graph to a n-node tree: the General starts a tree reduction process of the network, on each ray a control automaton counts the nodes and, when the (s+1)-th node is counted, this cell starts a ring reduction process of the virtual subtree departing from it (see fig.14). The synchronization can start two steps after the reduction process.

In this case the synchronization time is $d''+r''+2$ where d'' and r'' are the diameter and the radius of the tree obtained from the reduction process. The two step delay is due to the tree reduction process if this is performed according to [7].

Calling d' and r' the diameter and the radius of the tree obtained without putting bounds it is easy to prove that $r' < r''$ and $d' < d''$, where the equal sign holds when the maximum number of nodes in a ray of the standard tree is less than s.

On the other hand if d and r are the diameter and the radius of the network to be synchronized then $r=r'$ and $d \leqslant d'$; therefore $d''+r''+2 > d'+r' \geqslant d+r$.

The difference between $d''+r''$ and $d'+r'$ decreases by increasing n and vanishes for n large enough (n is the upper bound to the number of nodes in the tree); furthermore the difference between $d'+r'$ and $d+r$ depends on the reduction process and the neighbourhood of the original network [7].

6. CONCLUSION

We have shown that the controlled synchronization of linear arrays is applicable to synchronize optimally a tree with a bounded number of nodes, and that these results can be used to synchronize any tree though in this case the synchronization time is not optimal in general.

Now we conjecture that it is possible to synchronize optimally any tree by using as nodes more powerful automata like push-down automata or multi-push-down automata.

Moreover all the results concerning tree synchronization are applicable to the synchronization of networks arbitrarily connected.

APPENDIX 1

We prove the theorem:

For a tree T, a diametral path is contained in the following subgraph

1) the radial path GA;

2) the second radial path GB (i.e. the radial path of the tree obtained by suppressing the subtree which contains GA);

3) every radial path of the subtrees departing from GA (taking as radix the node automaton).

PROOF: assuming the General to be on a diametral path this path consists of the two longest minimal paths departing from the General: the radial path and the second radial path.

Otherwise if the General is not on a diametral path then the

end of the radial path is also an end of a diametral path; in fact (see fig.15) assuming BC to be the diametral path and A not to be an end of a diametral path then we have (putting |HC|>|HB|):

|GA| > |BH|+|HG| > |BH| => |HG|+|GA| > |BH| =>

=> |CH|+|HG|+|GA| > |CH|+|HB| and BC is not a diametral path. Thus AF is a diametral path, where F is an end belonging to the same subtree of GA; then F belongs to one of the radial paths of the subtrees departing from GA.

Therefore the subgraph of the hypothesis in either case contains a diametral path.

REFERENCES

1. Waksman,A. (1966) An optimum solution to the Firing Squad Synchronization problem, "Information and Control", 9, 66-78.

2. Balzer,R.M. (1967) An 8-state minimal time solution to the Firing Squad Synchronization problem, "Information and Control", 10, 22-42.

3. Moore,F.R. and G.G.Langdon (1968) A generalized Firing Squad problem, "Information and Control", 12, 212,220.

4. Grasselli,A. (1975) Synchronization of cellular arrays: The Firing Squad problem in two dimensions, "Information and Control", 28, 113-124.

5. Rosenstiehl,P. (1966) Existence d'automates finis capables de s'accorder bien qu'arbitrairement connectés et nombreux, "Internat. Comp. Centre", 5, 245-261.

6 Rosenstiehl,P. et al. (1973) Intelligent graphs: Networks of finite automata capable of solving graph problems, in R.C. Read ed., "Graph Theory and Computing", 219-265, Academic Press.

7. Romani,F. (1975) Cellular automata synchronization, to appear in "Information Sciences".

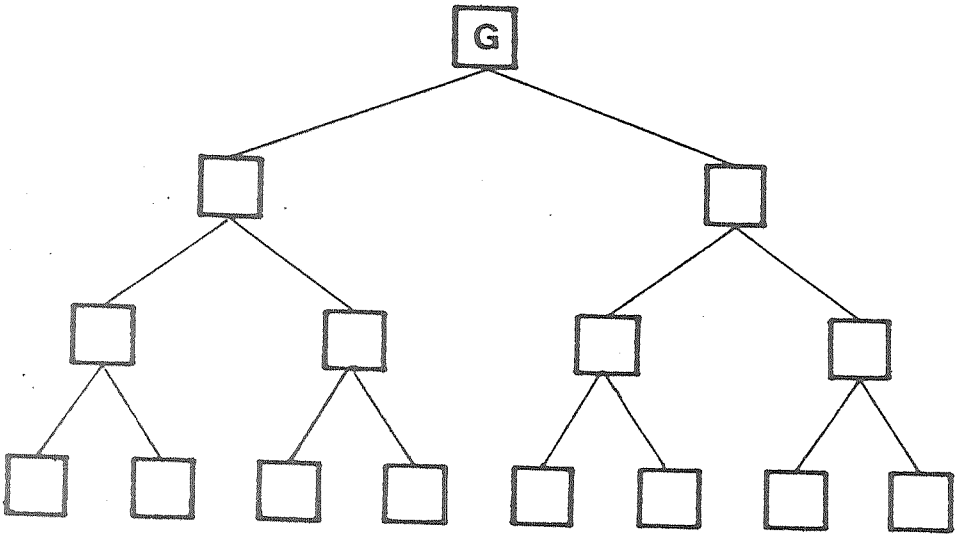

Fig.1 A binary tree.

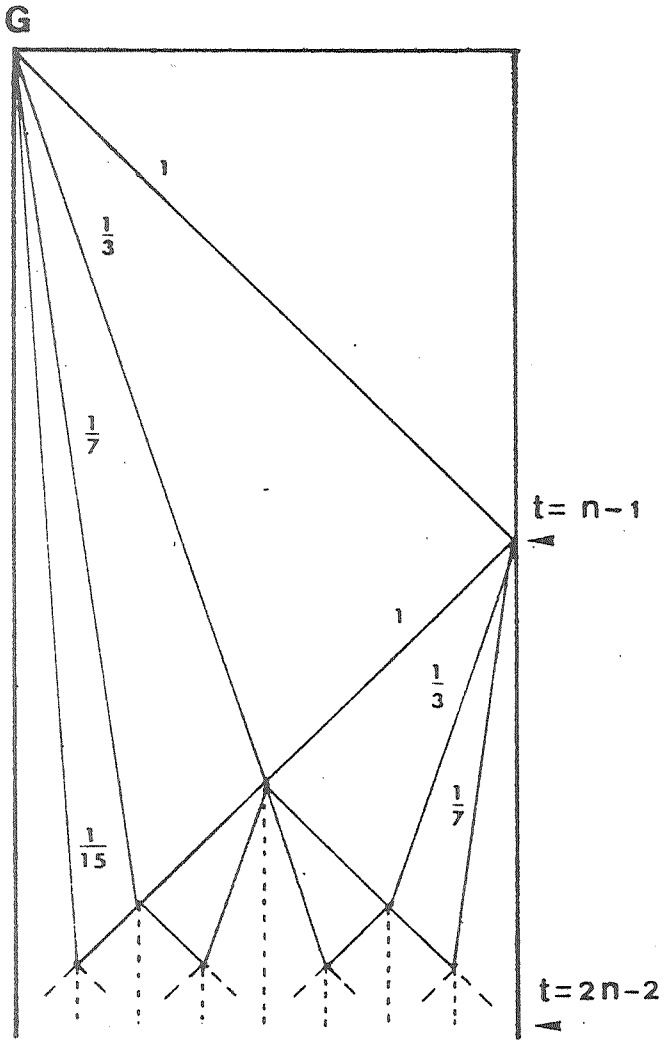

Fig.2 The Waksman method to synchronize a linear array with the General at one end.

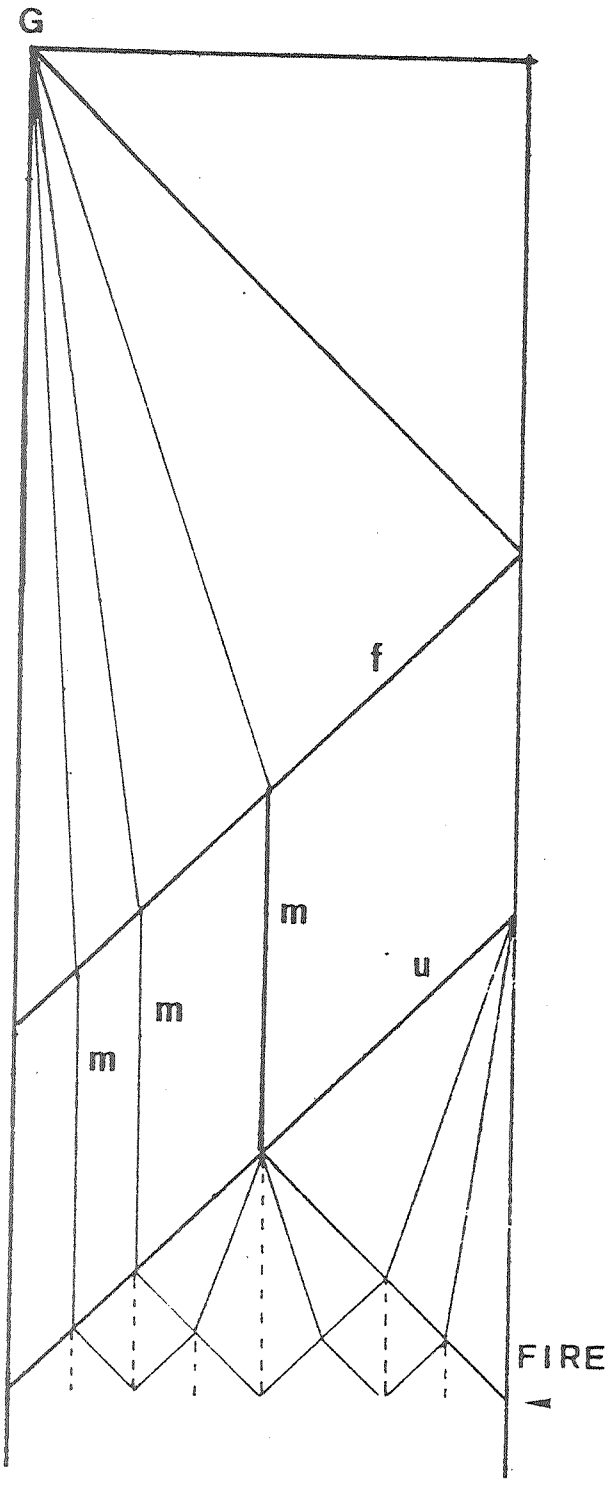

Fig.3 The delayed Waksman synchronization; f: freezing signal, u: unfreezing signal, m: marker (v=C).

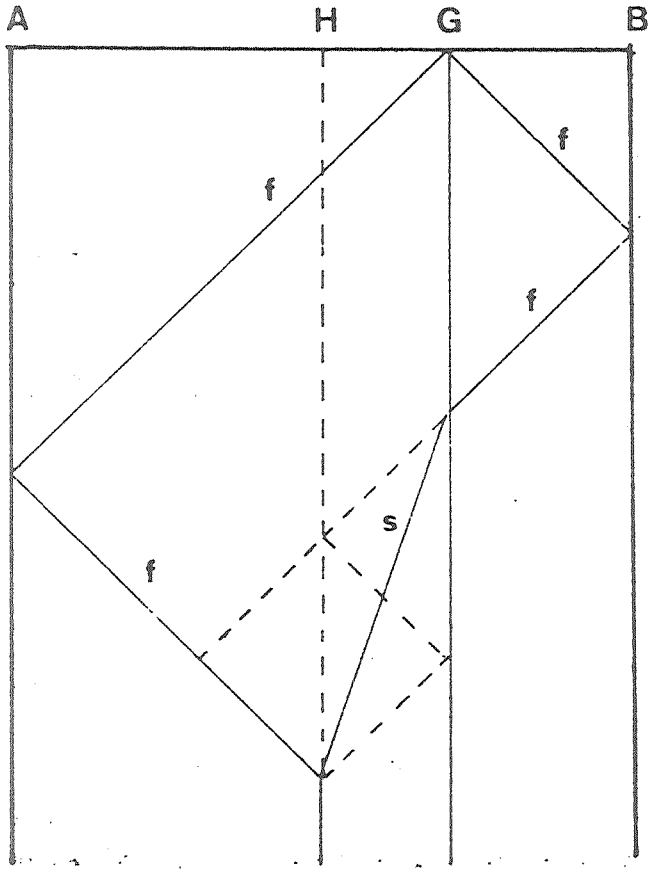

Fig.4 How to split a linear array into two equal parts; f: fast signal, s: slow signal (v=1/3). One has: ¦AH¦=¦HB¦.

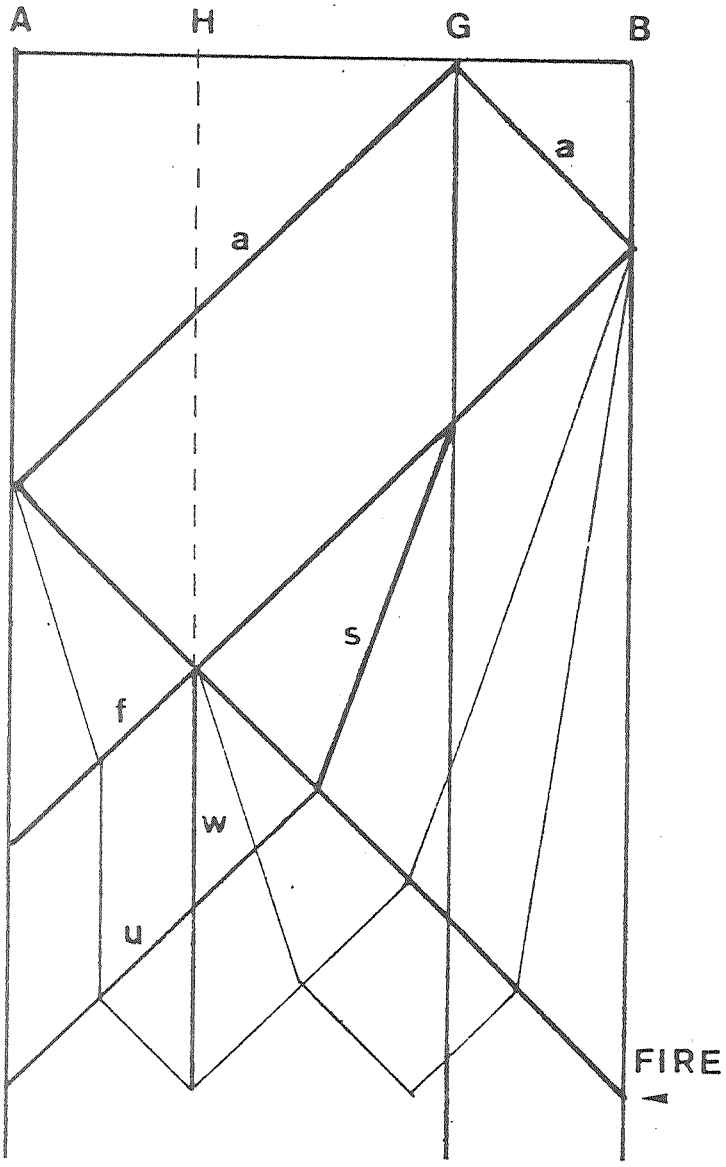

Fig.5 The Rosenstiehl synchronization of a linear array with the General anywhere; a: activating signal, s: slow signal (v=1/3), f: freezing signal, u: unfreezing signal,w: wall (v=C); the other signals are the Waksman synchronization signals of the two sub-arrays AH and HB.

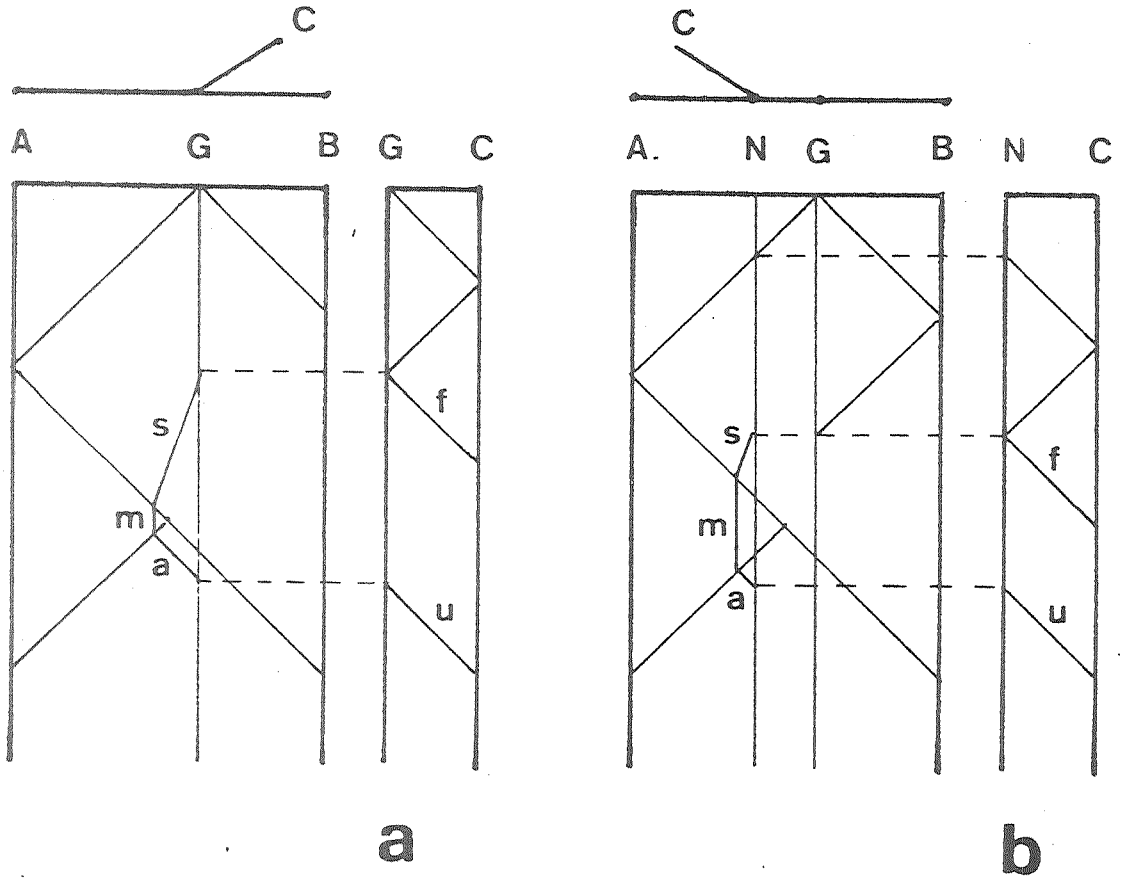

Fig.6 The set of synchronism signals for a secondary ray departing from the General (a) and from another node (b); s: slow signal, m: marker, a: activating signal, f: freezing signal, u: unfreezing signal.

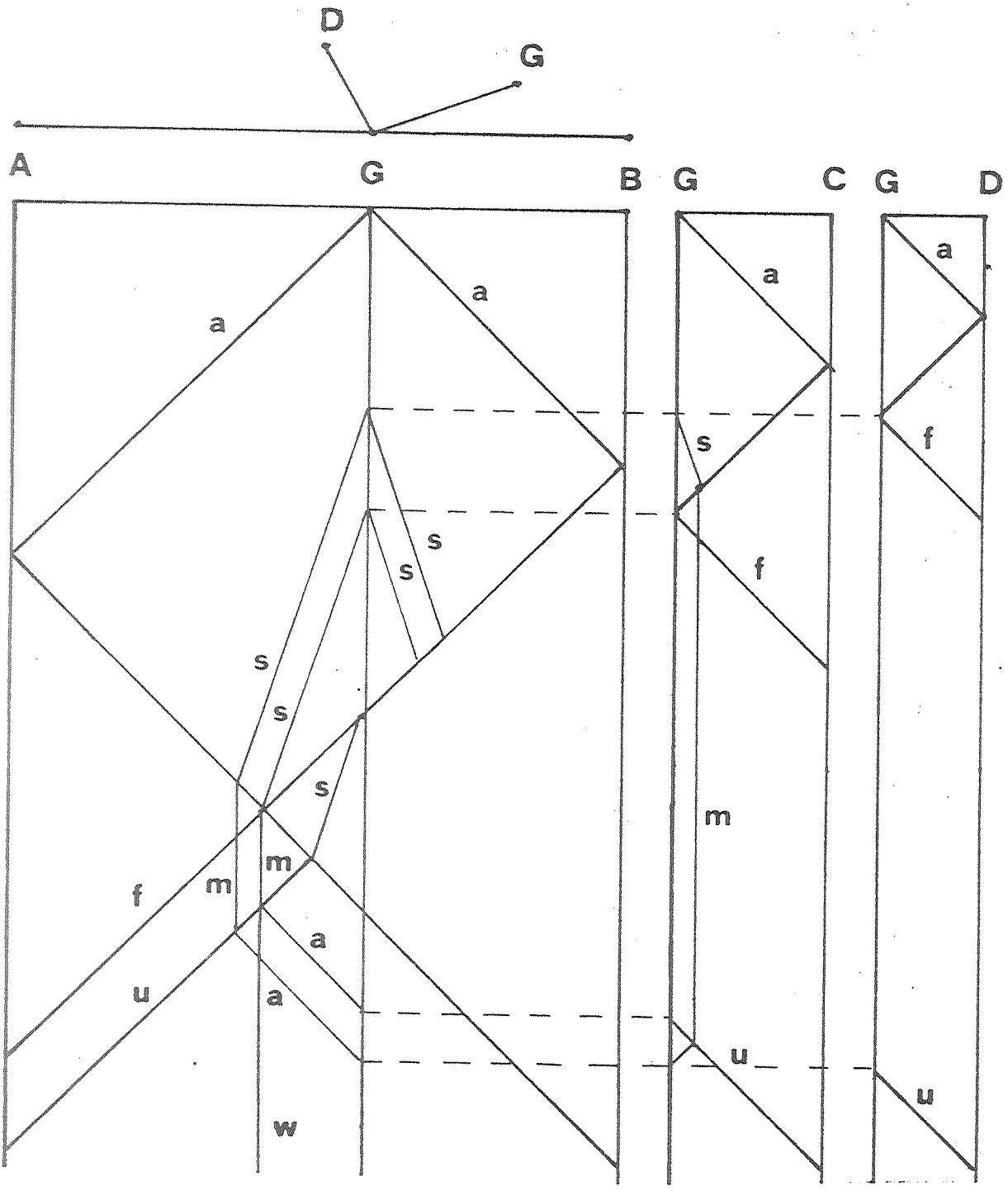

Fig.7 An example of star graph synchronization; a: activating signal, s: slow signal (v=1/3), m: marker (v=0), f: freezing signal, u: unfreezing signal, w: wall (v=0).

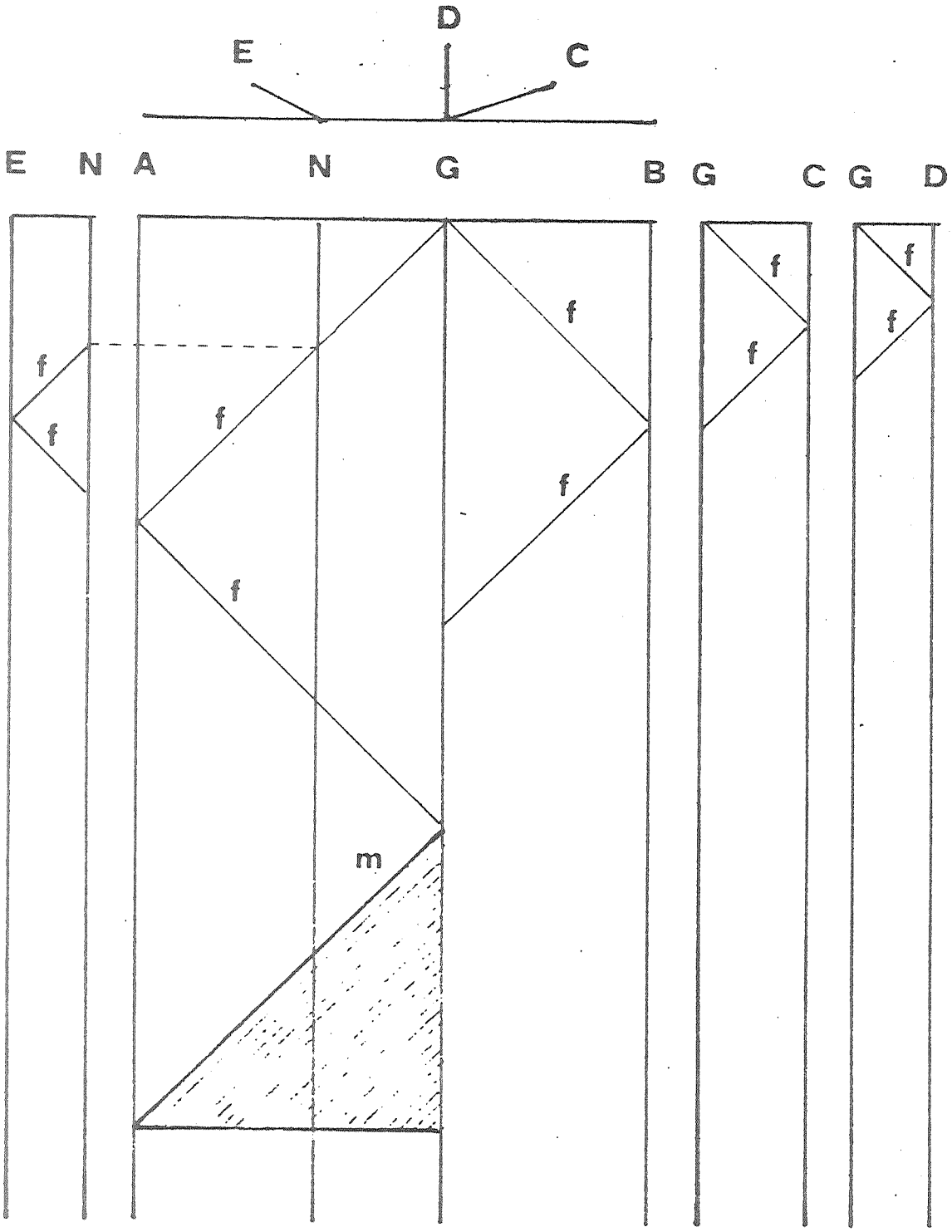

Fig.8 Radial path search; f: fast signal, m: marking signal.

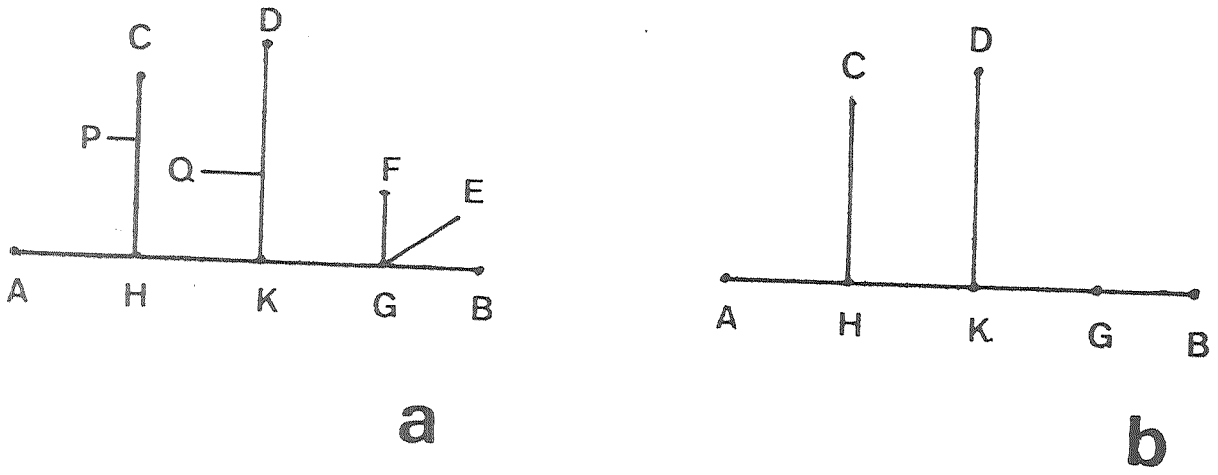

Fig.9 (a) A tree; (b) the reduced figure containing a diametral path.

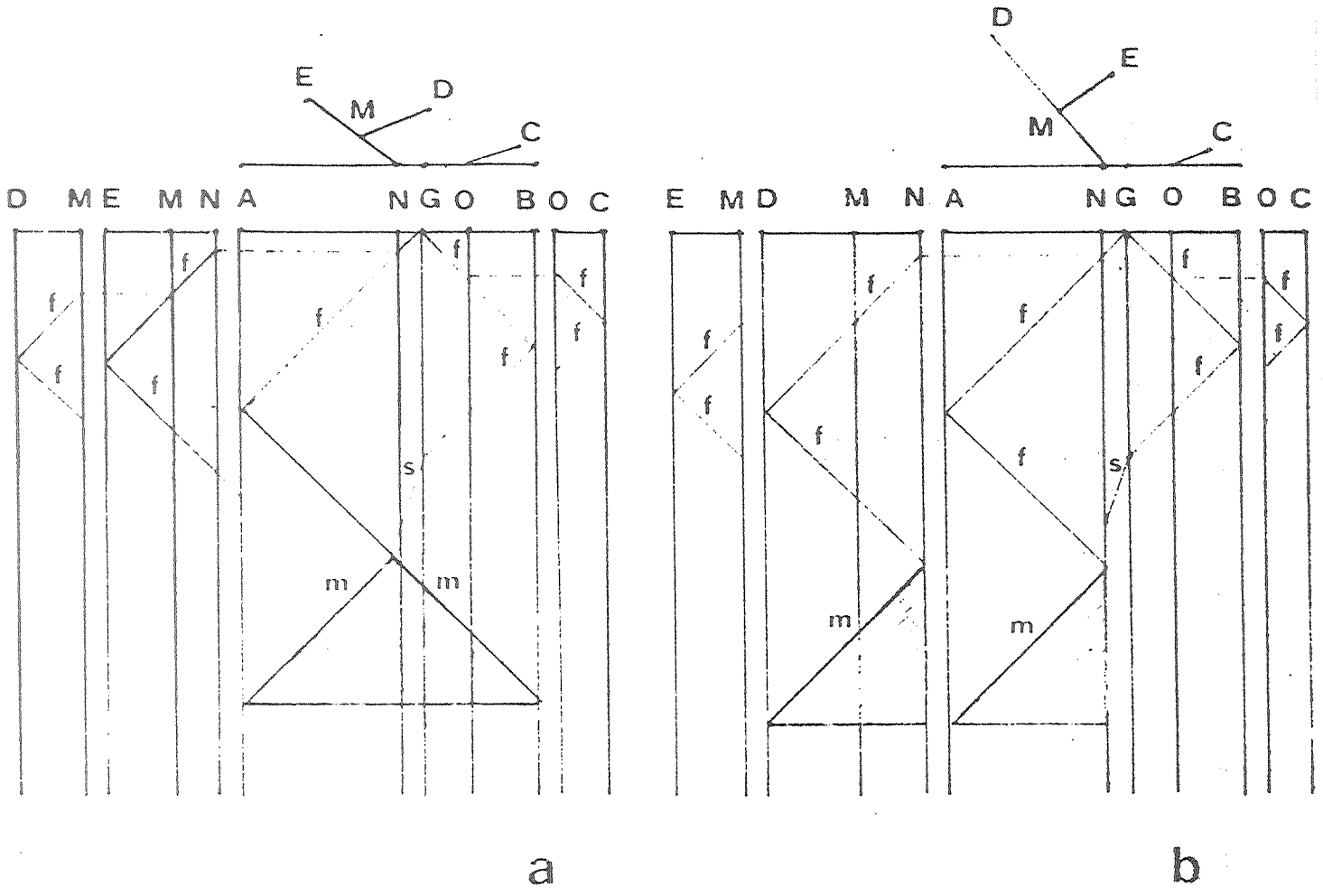

Fig.10 Two examples of diametral path search; f: fast signal, s: slow signal (v=1/3), m: marking signal (v=0).

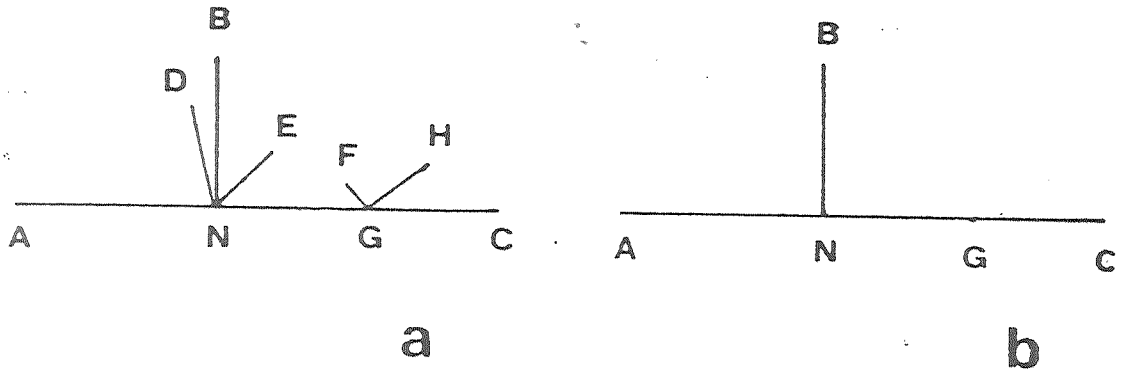

Fig.11 (a) A 2N tree; (b) the reduced tree containing a diametral path.

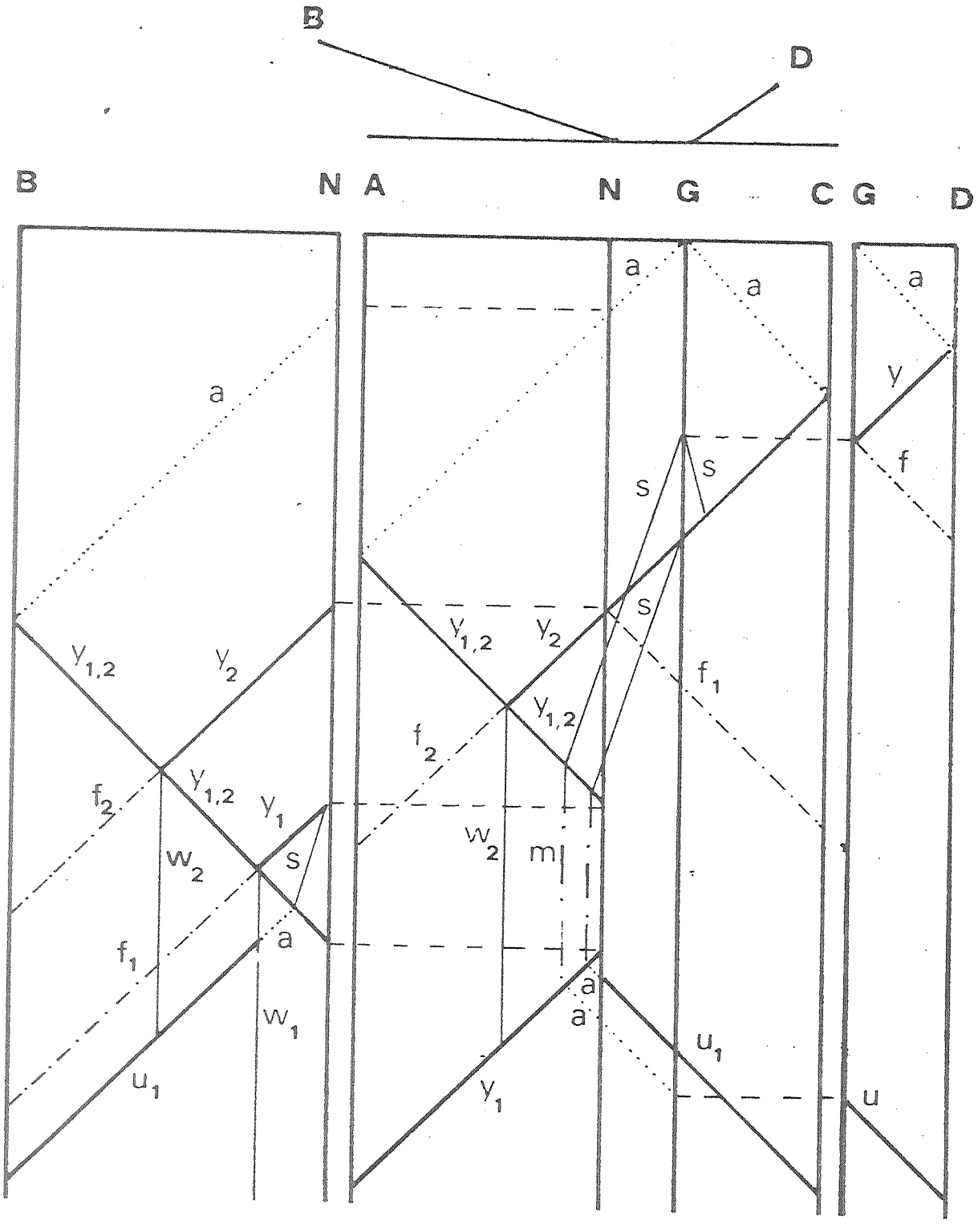

Fig.12 An example of 2N tree optimal synchronization; a: activating signal, y: Firing Squad activity, s: slow signal (v=1/3), f: freezing signal, w: wall (v=0), m: marker (v=0); subscripts indicate the corresponding set of Waksman automata. Comment: the states of the single automaton do not correspond one to one to the symbols used in this schematic diagram.

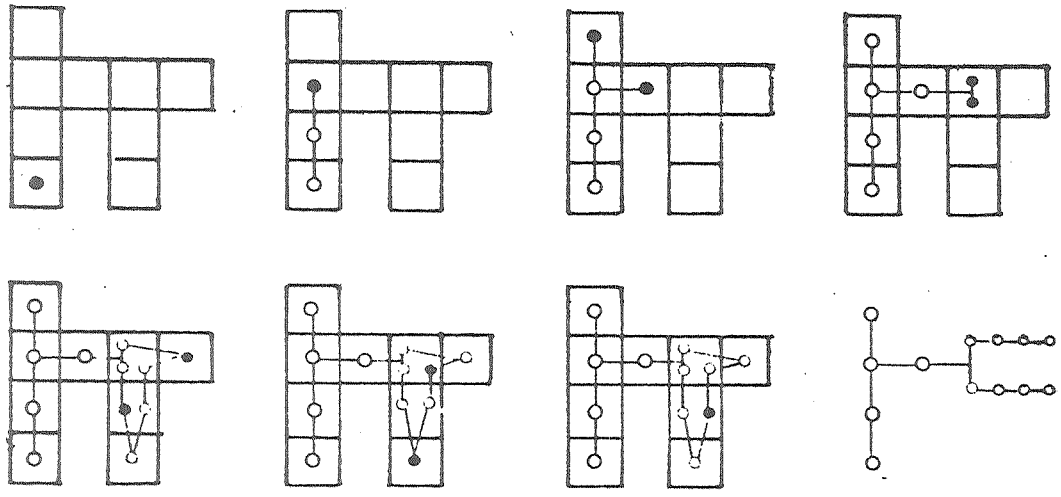

Fig.13 Some steps of the reduction process from tree to n-node tree.

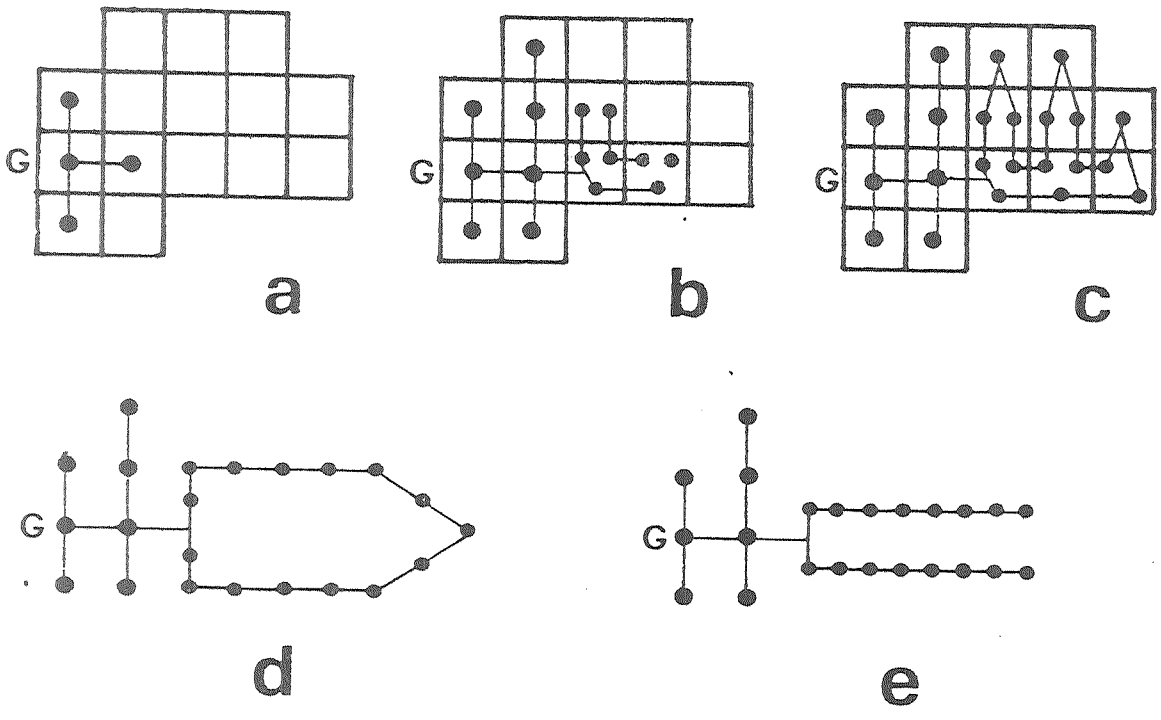

Fig.14 Some steps of the reduction process from connected network to n-node tree.

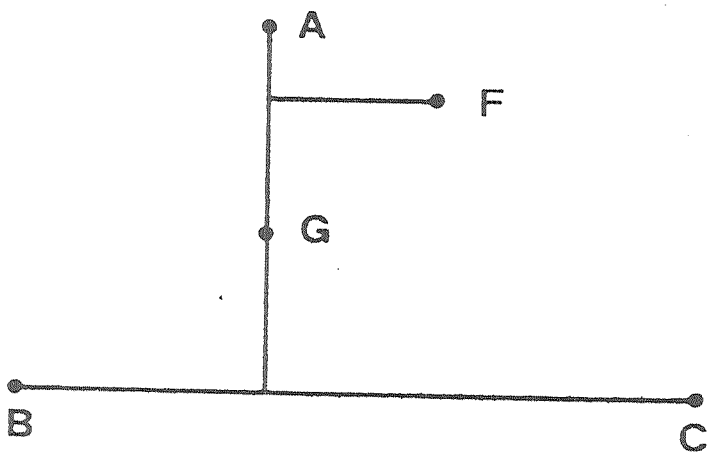

Fig.15 The tree according to the conjecture: GA is the radial path and BC is the diametral path.