THE UNIVERSITY OF MICHIGAN

COLLEGE OF ENGINEERING
Department of Electrical Engineering
Information Systems Laboratory

Technical Note

A PATH-BUILDING PROCEDURE FOR ITERATIVE CIRCUIT COMPUTERS

Rodolfo Gonzalez
Sandra Palais

ORA Project 04794

under contract with:

UNITED STATES AIR FORCE
AERONAUTICAL SYSTEMS DIVISION
CONTRACT NO. AF 33(657)-7391
WRIGHT-PATTERSON AIR FORCE BASE, OHIO

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

October 1962

Engm
UMR
1652

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (Concluded)

SUMMARY

Two of the main factors affecting the efficiency of an I.C.C. are studied in this report: path building and path interference. The first is equivalent to finding successive paths connecting any two squares on a rectangular grid, on which an ever-increasing number of obstacles is generated. Special emphasis is placed on solutions that are amenable to fast calculation by parallel processes.

The successive paths, a number of which are saved for possible future use, create obstacles which hinder, in a special way, the number and shape of new paths that may be traced through modules already occupied. The path traced, while not unique, is of minimum length. Further, because of the restrictions imposed, this length is pre-determined by the coordinates of the two squares to be connected.

A procedure is derived for detecting the formation of "full barriers," or obstacles that isolate a region. An algorithm is explained which, considering a restricted class of paths, permits a non-sequential elimination of squares in unsuitable positions without resorting to an exhaustive search and classification.

1. INTRODUCTION

The need for increased computational capabilities, brought into evidence when dealing with problems in the fields of pattern recognition, game playing or simulation of physical models, has suggested the use of highly parallel iterative circuit computers.[8-10] Furthermore, the recent advances in the technology of micro-miniature, integrated and "grown" electronic devices show promise that in the near future the availability of low cost modules essential for the practical realization of this type of machine will make even more pressing the need for detailed studies of this new concept.

A study of the pertinent literature[8-10] reveals that increase in versatility is accompanied by the introduction of several new problems inherent in this novel machine organization. Some of these problems are:

(a) Data allocation difficulties due to the "floating" address, as used in Holland's paper.[8]

(b) Programming difficulties due to the unlimited interaction possible between concurrently running programs.

(c) Necessity of some allocation protection method due to the presence of several programs running simultaneously.

(d) Problems in the flow of control due either to the lack of a centralized organ of command or to its impractically enormous complexity.

(e) Problems in programming and in interconnections generated by the lack of continuity of the geometrical properties of the space over which the machine is spread.

A number of these problems appear only in some of the proposed machine organizations, but others are germane to the very essential features of the general class of I.C.C.'s. Therefore, these latter problems deserve especial consideration and detailed study. The flow of information and control, as determined by the procedures used to connect operators and operands, is one of the factors that lies at the crux of the successful operation of I.C.C.'s. One approach to this problem is given in Holland's paper.[8] In Holland's theory, access to new operands is gained by adding or deleting modules from the termination of a fixed path. This method is essentially a counting procedure along a direction specified by the current instruction, and requires a small amount of hardware to implement it. This advantage is offset by the long time needed for path building since the procedure is a sequential one, and by the fact that the time needed for completion of the path-building phase is a function of the relative position of the modules to be connected.

At the other extreme, one could suggest a method using coincidence of addresses detection, resulting in a very fast but expensive procedure. As in many physical situations, speed is traded for complexity since these are the only two factors that can be rearranged in this case.

A different technique would be to generate a fresh path from each successively active module, and to leave the already used paths connected for possible future use. The potential gain in speed could, however, be offset by the difficulty in finding a new connection through all the pre-existent paths. An algorithm for path tracing is presented for the case of paths of some restricted shapes. In many cases, a situation will be reached in which paths belonging to one or more programs will form an obstacle such that some region of the network becomes isolated.

In this report, attention is given to the problem of path interference as related to the question of generation of barriers. An algorithm is also given for the detection of barriers formed by paths of any shape.

2. GENERAL OUTLINE OF THE PATH-BUILDING PROBLEM

A network of modules arranged in an n x n grid is given. The modules are all alike and contain a switching network (Fig. 1) and four registers that can be connected to any of the four outputs. Figure 2 indicates these for one of the registers.

The problem consists of successively connecting pairs of modules leaving intact a fixed number of the previously established connections for possible future use. This is referred to as path building. The end modules, i.e., the pair connected, are the originators and receptors of information and are called the _terminal modules_. The intermediate modules in the path serve only as connections and these are called the _connecting modules_. See Fig. 3.

The connecting modules can be in any of the nine states shown in Fig. 4 (a through i). A terminal module of one path can be at the same time a connecting module for another path, as shown in Fig. 5. Here the shaded squares indicate different states of the connecting and terminal modules. A preexistent path may be crossed by a new one, and in general, all the possible connections which the internal structure of the modules will allow can be made as long as the independence of the paths is maintained.

Therefore, any connecting module can be part of up to two different paths, and any terminal module can be associated with as many as four different paths, as the connection diagrams of Fig. 5 show. In this and subsequent diagrams, the modules are shown contiguous to each other, but this is simply an illustrational convenience not implying any other connections between the individual modules.

The paths are composed of segments, these being defined as the connecting lines between adjacent modules. However, since the internal configuration of the modules is not shown on the diagrams the connecting segments are con-

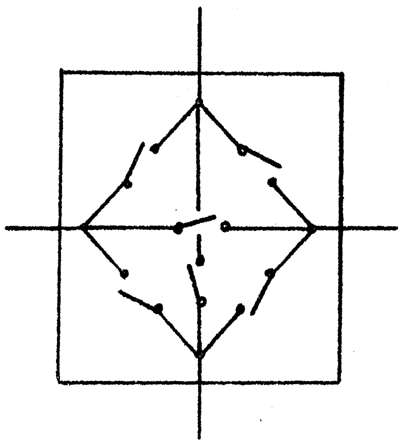


Fig. 1. Internal switching network.

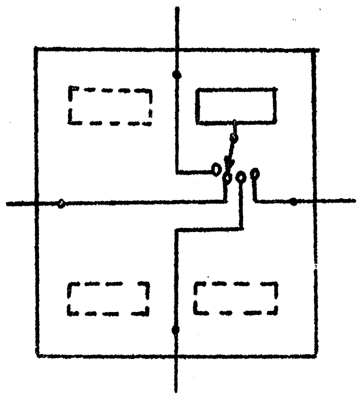


Fig. 2. Internal registers.

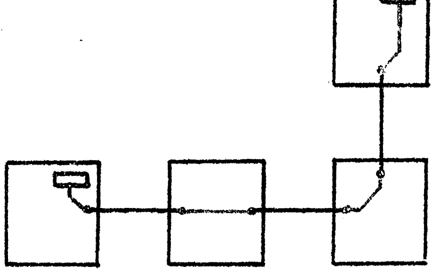


Fig. 3. Terminal and connecting modules.

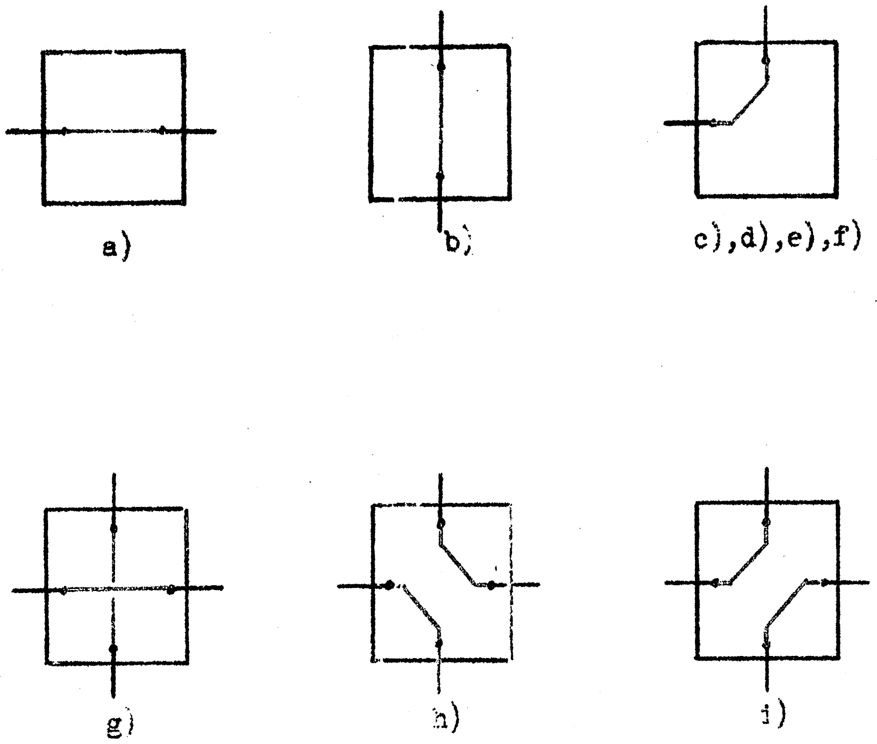


Fig. 4. States of connecting modules.

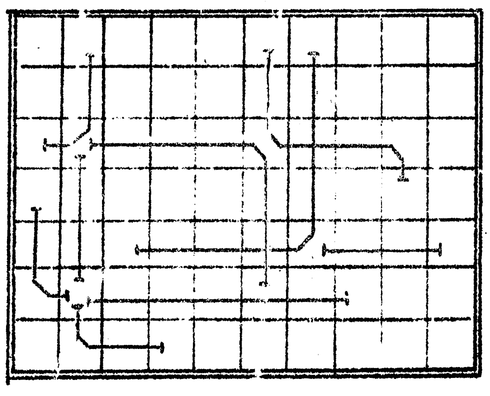


Fig. 5. Modules belonging to several paths.

sidered to extend from center to center of the modules.

As the number of pre-existing paths increases, it becomes more and more difficult to find a path connecting two given modules. The difficulty is caused by the accumulation of "obstacles" of various types. In some cases, these obstacles (previous paths) combine in such a way that a region of the network becomes isolated from the rest of the modules. In other cases, a single path of special shape suffices to isolate a region. Whenever this happens, we call the isolating line a barrier. Thus, a barrier is defined as a path or combination of path segments such that because of its particular shape and neighborhood relations it divides the whole network into two disconnected regions. In other words, it becomes impossible to connect two modules lying on opposite sides of the barrier. However, it is not easy to show the barrier as a physical entity because it entails geometrical shape as well as a positional relationship of the path or paths.

In general, if a path or a combination of path segments divide the network into two disconnected regions, one can visualize the barrier as the set of modules through which the segments constituting the path run.

A barrier may be generated by:

(a) A single path: See Fig. 6 a, b, c, d.

(b) Several paths: Any number of paths may contribute segments to form a barrier, as in Fig. 7. Paths a and b form a barrier between B and C; paths c, d, and e form a barrier between A and C.

Since a barrier is a function of the shape of the paths and of some neighborhood relation, it becomes imperative to find a mapping such that when both conditions defining a barrier are met, then some easily detected geometrical property is simultaneously satisfied. This leads to the definition of a "dual" of a path, as explained in Section 4.

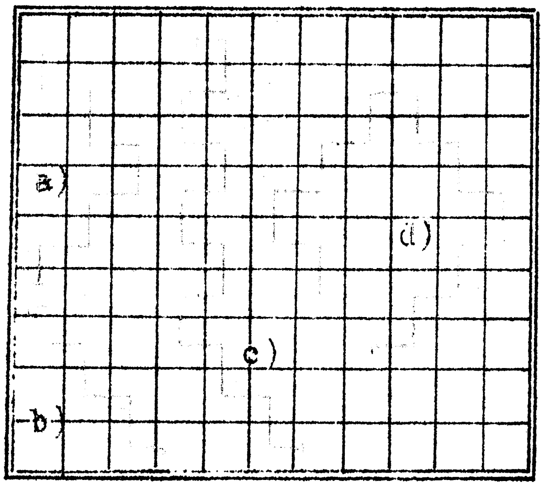


Fig. 6. Barriers formed by single paths.

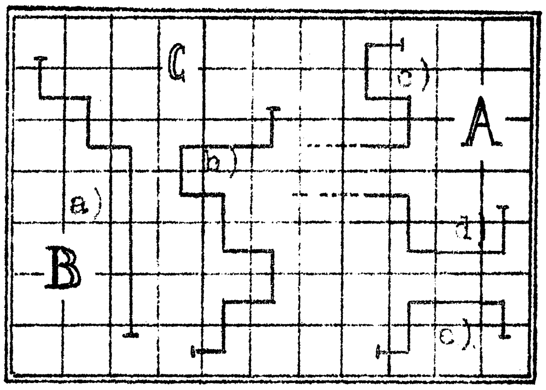


Fig. 7. Multiple path barriers.

Once this easy identification of barriers is obtained, another procedure is needed to ascertain whether two given modules lie in the same region or in disconnected regions with respect to every barrier detected up to then. If the two modules belong to disconnected regions there exists no path connecting them, and one has to resort to erasing some of the existent paths. This is done even if the allowable fixed number of paths has not yet been reached.

It must be remembered that here we are concerned with a path-finding procedure that must be repeated in its entirety for every path that is to be created. Consequently, some restriction on the types of paths to be considered is almost mandatory in order to avoid resorting to maze-solving techniques. Maze-solving techniques are essentially sequential algorithms and, as such, are much too slow for this application.

3. DEFINITIONS AND NOMENCLATURE

The purpose of this section is to present definitions and to establish uniquely the meaning of labels and names used in this report. The need for this rigorous defining of commonly used words is evident when, for example, several neighborhood relations with subtle differences have to be distinguished. Although the language is rich enough to permit this fine gradation of shades in meaning, the everyday use of these words has assigned to them almost synonymous implications.

In the next sections, the words defined below will be used exclusively within the meaning here indicated and, in many cases, a word of attention will be included to prevent misinterpretations.

A. Definitions: Neighborhood Relations:

Contiguous:	Meeting or touching on one side.
Adjoining:	Meeting or touching at least at some point. This concept includes that of contiguity.

Neighbor: Satisfying some empirical rule established as the "neighborhood relation," not necessarily implying contiguity.

As an example, the shaded modules of Fig. 8a and b are adjoining, but only the ones in Fig. 8b are simultaneously contiguous. This apparently excessive detailing is useful in cases like that of Fig. 8c: If we refer to the modules adjoining A, we refer to B, C, D, and E. But if we refer to the modules contiguous to A, only B and C qualify as such.

The neighborhood relation can assume any form and is not restricted to the common meaning of "immediate" neighbors. For example, the shaded modules of Fig. 9a and b represent the neighbors of module A under the conditions of the following rules: For 9a: Those modules having one side in common with A. For 9b: Those modules within a "Manhattan" distance of 3 from A.

B. Nomenclature:

Non-regressive path:	Any path traced in such a way that no two consecutive turns are in the same direction. Figure 10c and 10d.
Regressive path:	Any path with two or more consecutive turns in the same direction. Figure 11.
Normal path:	A distinguished set of the class of non-regressive paths characterized by having only one turn. Figure 10d.
Barrier:	A collection of paths dividing the network of modules into two disconnected regions.

C. Labels:

Labels of modules:	Modules are labeled with a pair of coordinates, as indicated in Fig. 12. The first term of the pair indicates the row, and the second the column to which the module belongs.
Labels of vertices:	The labels of vertices are derived from the ones of the corresponding modules according to the rule indicated in Fig. 13. The resulting coordinates are shown in Fig. 14.

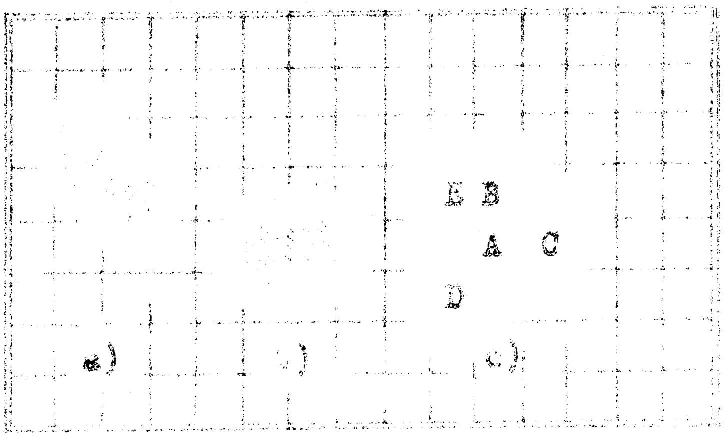


Fig. 8. Adjoining and contiguous modules.

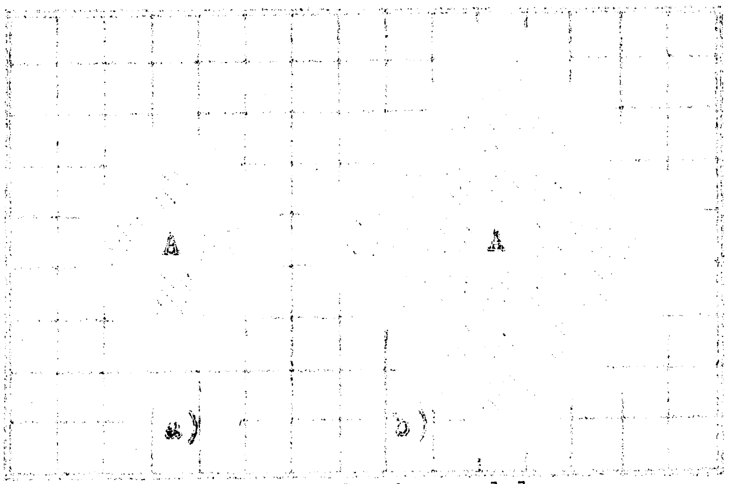


Fig. 9. Neighboring modules.

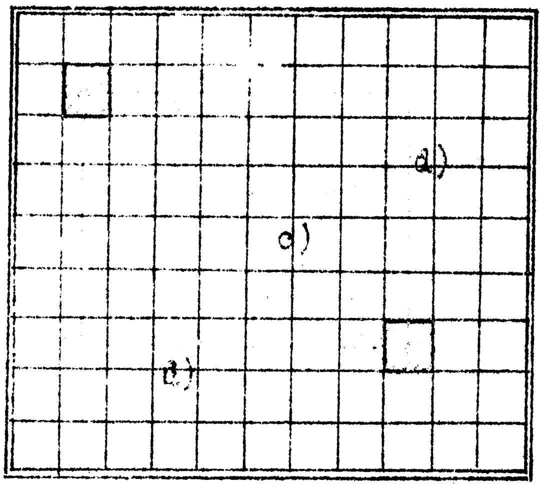


Fig. 10. Normal and non-regressive paths.

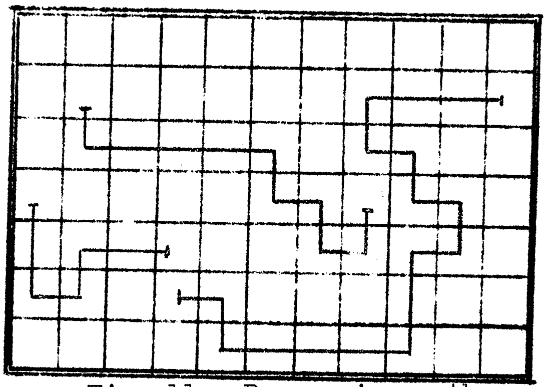


Fig. 11. Regressive paths.

(01;01)	(01;02)	- - - -	(01; n)
(02;01)	(02;02)	- - - -	(02; n)
(m;01)	(m;02)	- - - -	(m; n)

Fig. 12. Labeling of modules.

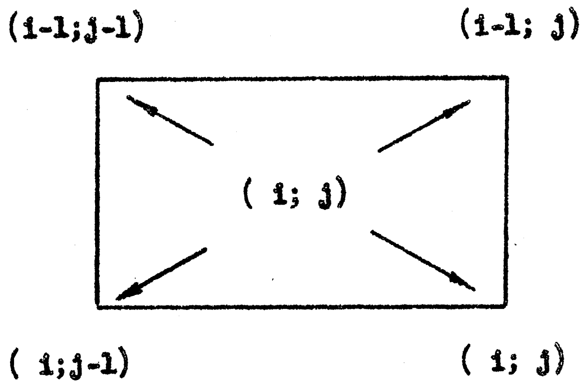


Fig. 13. Generating method for vertex coordinates.

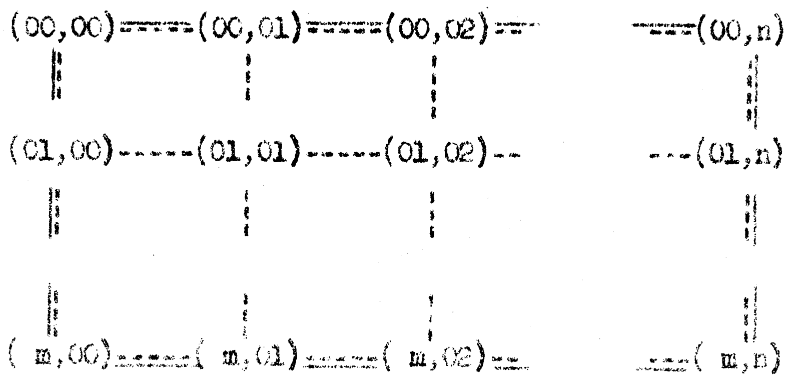


Fig. 14. Labeling of vertices.

4. DETECTION OF BARRIERS AND ISOLATED REGIONS

As explained in Section 2 a physical barrier is difficult to define because it entails the shape as well as a neighborhood relation between the paths contributing to the formation of the barrier. This difficulty is also present in the identification of barriers by means of an algorithm, because several tests have to be applied in sequence to ascertain whether a particular combination of path shapes and geometrical disposition constitutes a barrier.

Therefore, it seems logical to resort to some kind of mapping technique, such that when applied to the original configuration of paths it produces an image in which the desired property is easily identified.

At this point, we shall introduce some necessary definitions: Let us define the <u>dual segment</u> as the common side of a pair of contiguous modules connected by a segment of a path. Note that there is a one-to-one correspondence between dual segments and path segments associated with a particular path. Let the set of dual segments corresponding to a path be known as the <u>dual</u> of that path.

The <u>path lists</u> contain the labels of the modules through which the path is traced, and when the path is extended the list is updated by adding the label of the new module or modules. Simultaneously with the building of the path and the path list, the labels of the dual segments corresponding to the paths are ordered in lists. If the dual remains as a connected line then a single list is maintained, but when a disconnected dual segment is generated, it starts a new list. It can happen that while generating a new path, all the corresponding dual segments generate independent new lists. Further, if a dual segment S_1 belonging to a dual list D_1 is connected to some dual segment S_2 belonging to a dual list D_2 originally created by some other path, then the contents of the dual list containing S_1 are added to the list containing S_2. At any stage, a dual list contains only a connected sequence of dual segments, perhaps contributed by several paths.

Therefore, there is no correspondence between the path lists. The paths simply generate the dual segments and these, depending on their resulting connections, can add to the dual of their own path, generate other dual lists, or join some pre-existent dual lists. Consequently, each dual list contains the coordinates of dual segments such that they have one coordinate in common, identifying them as belonging to a connected tree.

Lemma 1: A list of dual segments containing either a closed sequence of coordinates or the coordinates of two points belonging to the border of the network defines a line which divides the grid into two disconnected regions. This line is called a _barrier_.

Proof: Since each dual segment arises from a corresponding path segment connecting two contiguous modules, clearly no new path can connect them. Hence, no new path can cross the dual segment. Therefore any continuous sequence of dual segments cannot be crossed anywhere by a new path. So if both ends belong to some border or close on themselves, a disconnected region is generated. Hence, by definition a barrier exists.

Corollary 1: Discontinuity of the set of dual segments arising from a single path indicates the possibility of crossing the path by a _new_ path at every point of discontinuity.

Example: It is desired to determine if points A, B and C can be connected two at a time, given that the four paths indicated in Fig. 15 are pre-existent.

First, the duals of the paths can be generated according to the definition of dual. In Fig. 16, the four duals corresponding to the paths are shown distinctly to emphasize their origin. However, we are really interested in

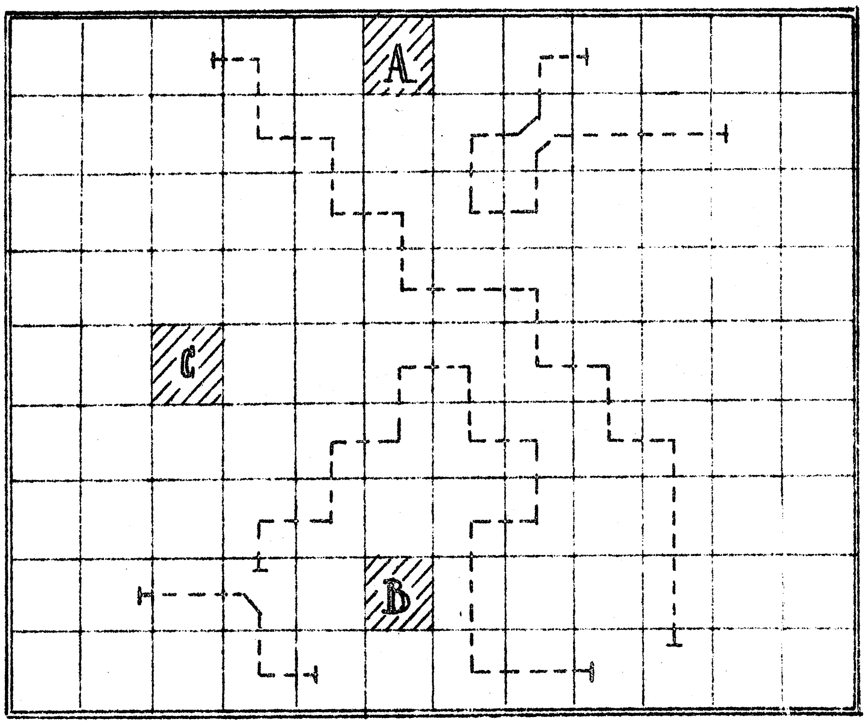


Fig. 15. Pre-existent paths.

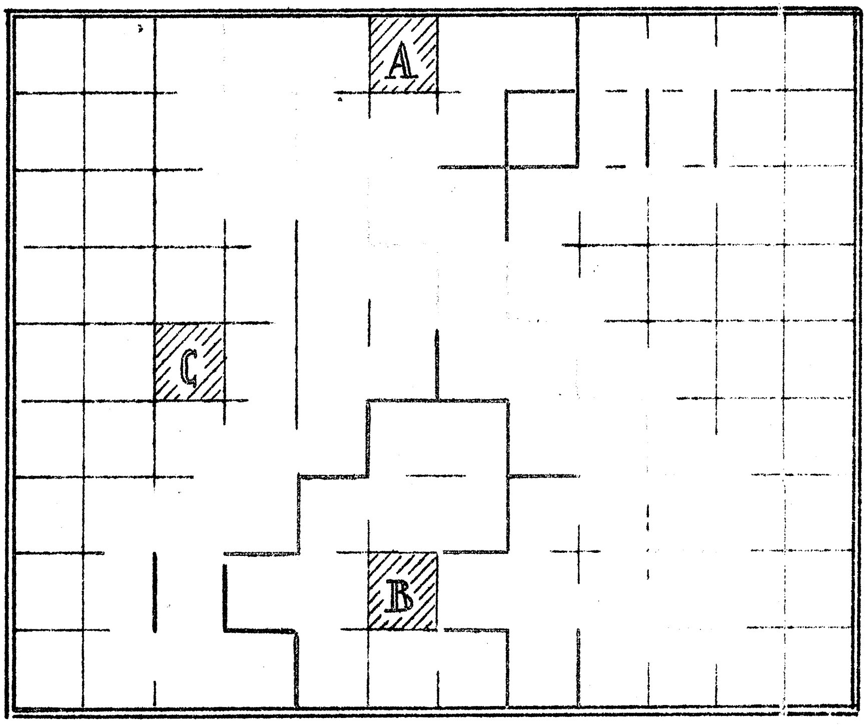


Fig. 16. Duals of the pre-existent paths.

connected sequences of dual segments independently of whether or not they belong to the same path. Therefore, at this stage, there should be nine dual lists as can be verified from Fig. 16 by counting the number of connected dual lines. Of all these nine lines, only one satisfies the requirements of Lemma 1; and this line is shown in Fig. 17. Because it is a continuous line formed by dual segments and with the ends on the border of the network, it is defined as a barrier and thus divides the network into two disconnected regions. As a result, it can be stated that module C is isolated from A and B, but A and B can be connected. If now all the duals are shown again, as in Fig. 18, it is easy to see how to trace the connection AB.

This suggests the possibility of solving the problem of connecting two points through the existent paths by treating it as a maze problem in which the walls are represented by the duals instead of by the actual paths. While this is perfectly possible when only a few of these connections are needed, it is completely impractical in this case when the procedure is to be repetitive, and one has to create a new path as soon as a connection has been made and some information has been transmitted between the terminal modules. Thus, a much simpler and faster, if less general, procedure is needed.

Programming the Detection of Barriers:

While the number of paths is increasing, the lists of paths and of dual segments are constantly being kept up to date, and thus they increase both in length and number. A new kind of list denoted as a "barrier list" is generated from the list of duals whenever either of the following cases occur:

(a) When one entry happens to have each of the two coordinates common to one coordinate of two previous entries, identifying it as a link closing a loop; and

(b) When the list contains two coordinates belonging to the borders of the network.

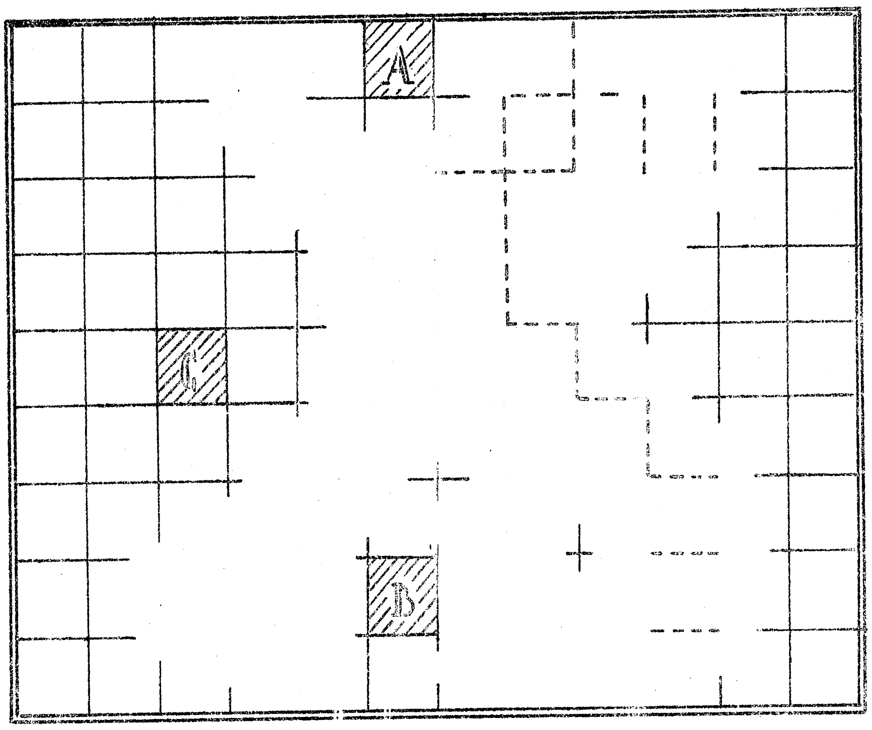


Fig. 17. Continuous barrier.

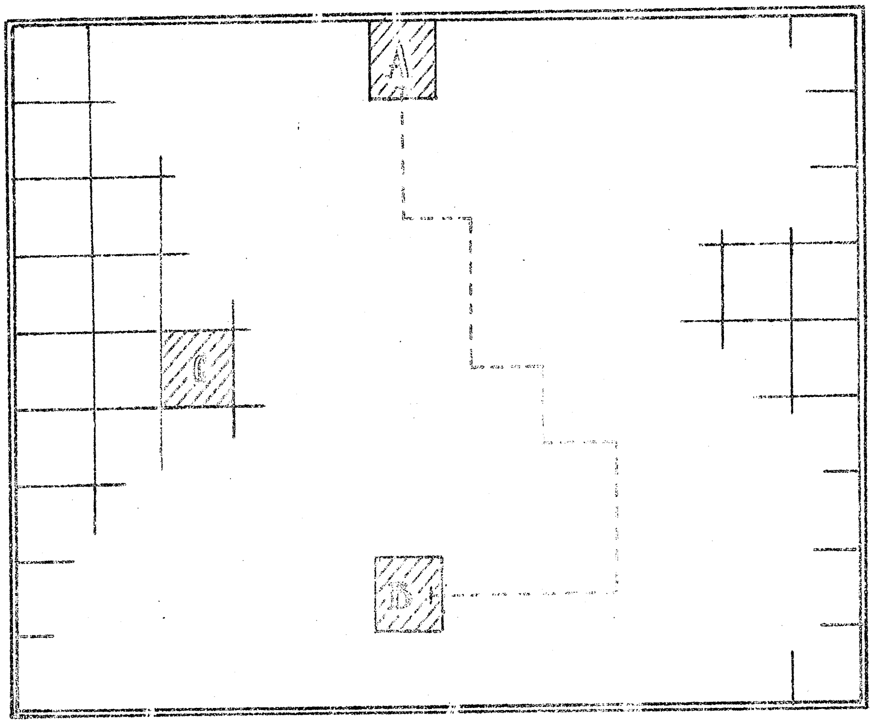


Fig. 18. Tracing path A-B.

The continuous updating of the three kinds of lists suffices to tell if a path is possible between any two given modules at any moment.

It is evident that if no barriers have as yet been completed and if no restrictions are placed upon the path as to the number of corners or its permissible length, then the connection is always possible. But even if one or more barriers are already present, it is still possible that the two modules to be connected lie in the same region with respect to the barrier or barriers. Two solutions for this problem are suggested:

(a) For each new barrier, two tables would be generated containing the labels of all modules belonging to each of the two disconnected regions. Any two new modules to be connected would be checked to see if their labels are contained in a single table, in which case both lie in the same region with respect to the barrier inducing the partition under consideration.

Obviously, this method implies the shifting of enormous amounts of data (the labels of all modules), and the whole procedure has to be repeated for every new barrier. Furthermore, the amount of data to be treated remains appreciably constant and does not diminish as could be expected, since the modules belonging to other previous barriers can now act again as originators of new paths.

(b) A normal path connecting the two modules is built on an assumed blank network, and its dual list is generated. Then Lemma 2 is applied.

Lemma 2: Given two modules, if the dual of any of the two normal paths connecting them contains an odd number of segments in common with the barrier proper, then the modules lie in disconnected regions and a path is not possible.

In Fig. 19, the black segments indicate the dual of the normal path connecting A with B. The green line is the barrier created by the red path. In Fig. 20a, the dual of the normal path (black) and the barrier (green) have one common segment, but Lemma 2 specifies with the barrier proper, so 20b is the appropriate representation of this case. Here, C, A and B are found to be in the same region because an even number, namely two, of segments are common to both the dual of the normal path and the barrier proper.

If the other normal path is traced it is found that it too has an even, namely zero, number of segments in common with the barrier proper.

5. FINAL CONSIDERATIONS FOR PATH BUILDING

A. Restriction on the Class of Paths Admissible

It has been mentioned in Section 2 that the path-building procedure has to be repeated for every instruction to be executed, and therefore it is imperative to employ a very simple and fast algorithm for path tracing. Furthermore, since some of the paths may be of considerable length, it will be helpful if the algorithm is amenable to parallel processing.

If one tries to resort to maze-solving techniques, it is found that these use a method of cell classification, assigning relative weights according to some neighborhood relation and in a monotonically varying sequence. Therefore, these methods are intrinsically sequential since the weight of a cell cannot be determined until the weight of its immediately preceding cell is specified.

In order to simplify the problem, we restrict the types of admissible paths to what is referred to as "non-regressive paths." A non-regressive path is defined as one traced following a set of priorities on the vertical and horizontal directions. The set of priorities establishes which of the

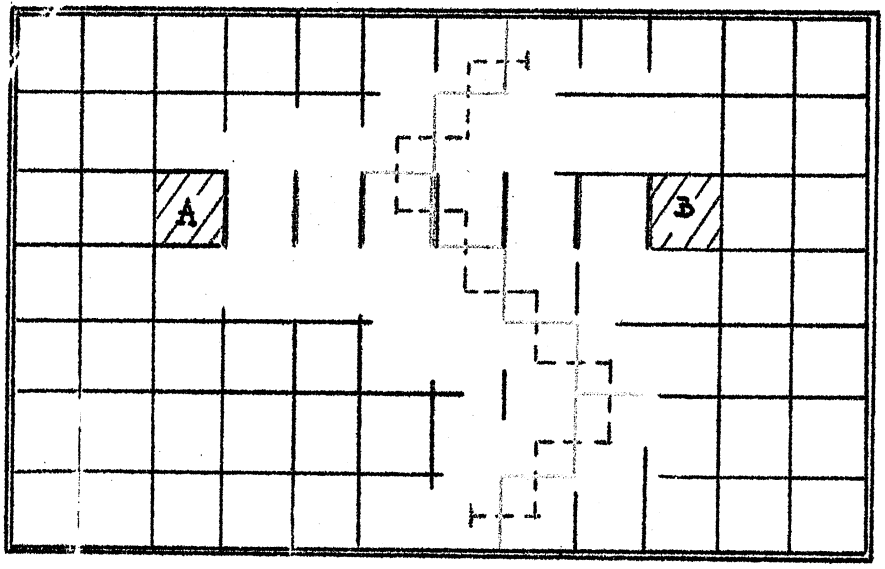


Fig. 19. Dual and barrier.

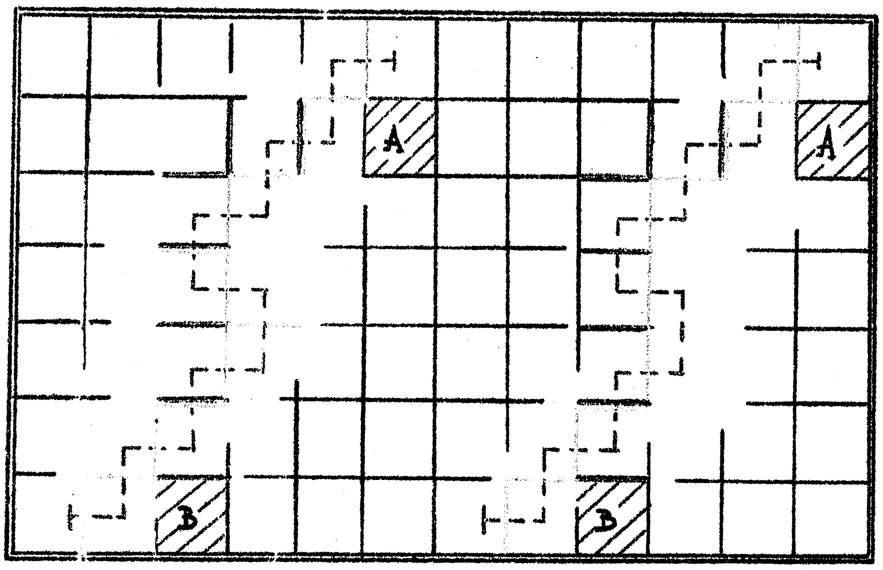


Fig. 20. Dual and barrier proper.

two senses are to be followed when tracing the segments in each of the horizontal and vertical directions.

This means that once the vertical and horizontal priorities are established, for example vertical down and horizontal to the right, all the path segments have to be traced in either of the two specified senses exclusively.

It is to be noted that this restriction on the allowable class of paths has the advantage of eliminating the need for an algorithm capable of tracing a minimum length path, since all non-regressive paths have the same length when measured in terms of the number of segments needed to connect the modules. This method of measuring distance has been called "Manhattan distance. And if each module is assigned a pair of coordinates, then this distance can be assimilated to the "Hamming distance."

It could be thought that this restriction on the types of paths would severely limit the possibility of connecting two modules through a set of obstacles, but it is easy to show that this is not the case even for networks of small size.

For an m x n network, the number of paths (non-regressive) connecting two opposite corners is:

$$\text{no. of n-r paths} \quad \binom{n+n-2}{n-1} = \frac{(m+n-2)!}{(m-1)!\,(n-1)!}$$

For a minimal iterative circuit computer, with a 40 x 40 network, the number of non-regressive paths is

$$\binom{m+n-2}{m-1} = \binom{78}{39} = \frac{1.13 \times 10^{115}}{(2.03 \times 10^{46})^2} = 280 \times 10^{20}$$

Even for the impractical case of a 10 x 10 network, the number of paths is still very high:

$$N = \binom{20-2}{9} = \frac{18!}{9!\ 9!} = 48{,}620$$

Within the class of non-regressive paths we distinguish the "normal paths" and the "broken paths," illustrated as a, b, and c respectively in Fig. 10.

B. Use of Redundant Paths

Since the number of available paths is so large, it is necessary to define some priorities as to the type of path most convenient to try to build first. Evidently the normal paths employ a very simple algorithm and therefore seem to be the natural choice for a first try. Furthermore, the length of all non-regressive paths, including the normal ones, is the same, and therefore there is no particular advantage with respect to propagation time through the path. But there is an advantage in time during the path-building phase, since only one "turn" instruction is needed in the case of the normal paths.

Since it is very probable that it may be possible to trace more than one path, one could very well use the extra paths to add redundancy to the transmission of information between the modules. If only one path is found, then it is used for the transmission with no checks. If two paths are available, the information received from them at the receiving module is checked for agreement and it is then used or discarded, stopping the program. If three or more paths happen to be available, a "majority vote" can be taken at the receiving end to determine the correct information.

The "majority vote" technique when used with three paths affords a high degree of reliability since the condition for acceptance of incorrect information is the occurrence of the same type of error at the same time in two of the three channels.

All the modules in the path except the end ones act only as transmission elements and therefore perform no logical function. It is safe then to assume that the only type of error that can be introduced is the failure of transmission, as opposed to the dropping of bits or the generation of erroneous "ones" filling the spaces of some original "zeros."

Under these conditions, the bounds for cumulative errors as treated in Reference 4 do not apply. The simpler rules that follow give an estimate of the increase in reliability that can be expected from a transmission line composed of m parallel paths.

Let's denote by R_i the reliability of the individual module, and by R_t the reliability of the total path. Similarly, $F_i = 1-R_i$ will indicate the individual "failability" of the modules.

It is necessary here to remark that reliability is understood as the probability that the element will perform correctly during a certain time interval; in this case, during the time it takes to execute the transmission phase. That is, a reliability of 0.95 indicates that the element functions correctly 95 out of 100 times that it is pressed into service. It does not indicate that the element performs correctly during 95% of the transmission phase in any one attempt since in this case only a very few special patterns would be transmitted with no errors.

In order to calculate the reliability of systems with elements having individual reliabilities R_i, one proceeds in the following way: The total reliability of a series of n elements with individual R_i's is the product of these R's.

$$R_t = \prod_{1}^{n} R_i$$

In this case, all the modules are physically alike, so that barring different environmental conditions all are supposed to possess the same $R_i = R$. Also, in this case, the total reliability of a path consisting of n elements in series is simply: $R_p = R^n$. See Fig. 21.

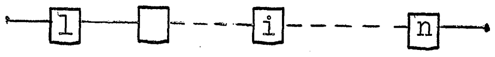


Fig. 21. Series connection of n modules.

When m such paths of individual reliabilities R_p each consisting of the same number n of elements are connected in parallel with no intermediate interconnections, as in Fig. 22, then the total "failability" is $F_T = F_p^m$, and therefore:

$$R_T = 1 - F_T = 1 - F_p^m = 1 - (1 - R_p)^m = 1 - (1 - R^n)^m.$$

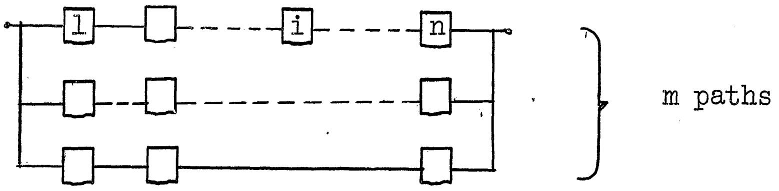


Fig. 22. m parallel paths of n modules each.

It is evident from the previous equality that the number m of parallel paths has a greater influence than the number n of elements in series in a path. This is especially true for values of R approaching unity and means that even for a long path there exists a possibility of compensating the effect of the large number of elements in series with a few parallel paths. For example:

For $R = 0.95$, $n = 10$, $m = 3$:

For a single path: $R_p = (0.95)^{10} = 0.5987$

For 3 paths in parallel: $R_T = 1 - (1 - R^n)^m = 0.9345$

It can be seen that the total reliability has not yet reached the original reliability of the individual module.

For $R = 0.99$, $n = 10$, $m = 3$:

For a single path: $R_p = (0.99)^{10} = 0.9045$

For 3 paths in parallel: $R_T = 1 - (1 - 0.9045)^3$

$$R_T = 1 - (0.0955)^3$$

$$R_T = 1 - 0.00087 = 0.99913$$

In this case, the total reliability of the network is greater than the reliability of the individual module.

C. Extra Requirements Introduced by the Redundant Mode of Operation

It is to be noted that this feature of built-in redundancy is obtained with almost no penalty in time or equipment. Time is not lost since all m paths can be traced at the same time and all are of the same length, and thus the procedures terminate simultaneously.

An extra requirement is the necessity of having extra hardware in the modules to implement the majority vote. This simply means that for a fair-sized machine there is a slight reduction in the number of modules available for the rest of the program.

D. Assignment of Priorities

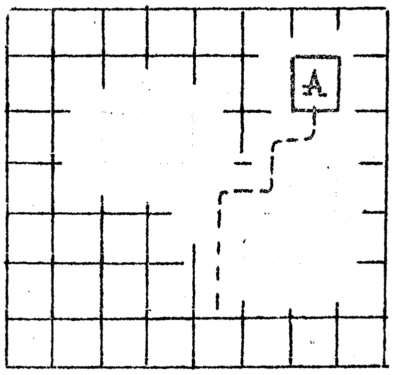
Given two modules to be connected we will consider the one with the lowest row coordinate as the starting point. Informally, we can say that the "uppermost" module is the starting point. The priority is established as a sequence of two directions, and is indicated as the pair consisting of V (for vertical) and H (for horizontal) in one of the two orders.

Thus, the pair (V,H) indicates that the vertical direction is followed as long as it is possible, whether or not a change to horizontal direction is possible. Only when an obstacle is met does the change to horizontal direction take place. But still the vertical direction has latent priority, and consequently even if the horizontal direction is clear the vertical one is resumed as soon as it becomes possible to do so.

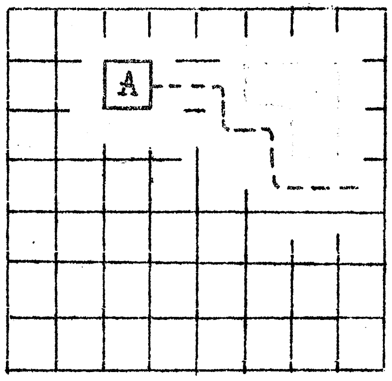
As the starting point has been defined as the uppermost module, the vertical direction needs no further qualification since it can only be in the "down" sense. The qualification for the horizontal direction is automatically given by the relative position column-wise of the "lower" module relative to the "upper" module.

Given two modules, the two combinations of (V,H) and (H,V) can, and generally do, give rise to completely different paths. In Fig. 23 the two different combinations of priorities are illustrated. In 23a the priority is (V,H) and the starting direction is vertical. Notice that when the path is proceeding in the horizontal direction, it resumes the vertical direction as soon as this becomes possible. This is notwithstanding that there are two or more modules available in the horizontal direction.

In some cases, the path starts from the uppermost module to be connected, following the low priority direction and thereby apparently violating the rules of direction precedence. What actually happens, as in Fig. 24, is that the immediate neighbor of the starting module in the direction specified by the priority is either an obstacle or is non-available because it lies in the shadow of some other obstacle. Therefore, as the high priority direction finds no available modules through which to trace the path, the secondary priority is followed, but only as long as is necessary to find an available module in the high priority direction. In Fig. 24a the obstacles are such that even with a (H,V) priority the resultant path has all the features of one traced according to a (V,H) priority. The same happens for a (V,H) priority, as seen in Fig. 24b.

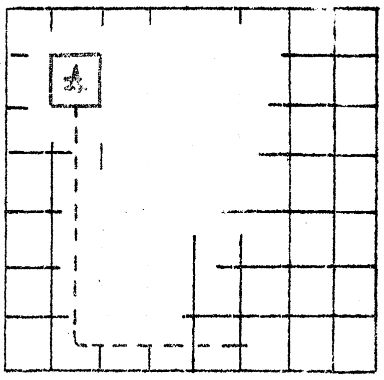


a) Priority: (V H)

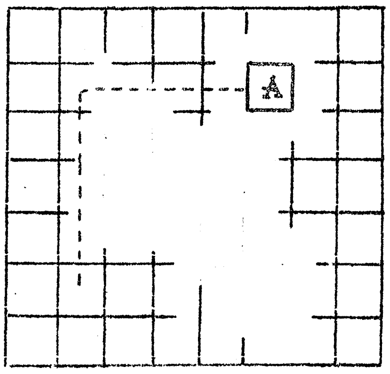


b) Priority: (H V)

Fig. 23. The two cases of priorities.



a) (H V) priority



b) (V H) priority

Fig. 24. Two cases starting with the low priority direction.

E. Path-Building Procedure

The path-building procedure involves two different steps: (i) Eliminating zones of modules non-acceptable as path components, and (ii) Tracing procedure.

(i) The elimination of module zones non-acceptable as possible path components is carried on by a "shadowing" technique. This procedure takes into account the priority pair and indicates which zones are forbidden for path penetration. This implies that if the path is allowed to enter one of these zones, then it becomes necessary to trace back part of the path in a direction opposite to one of the directions specified in the priority, thereby producing a regressive path which is not admissible.

The shadowing method operates in the following way:

(a) Determine the starting point and choose a priority pair.

(b) From the priority pair and the relative position of the two modules, determine the sequence of directions in which the path has to progress.

(c) From this sequence of directions, determine the two sides of the network which will serve to determine the starting obstacles for the shadowing procedure. If the directions are vertical down and horizontal left, then they can be indicated by a pair of arrows, like $\overset{\downarrow}{\leftarrow}$. From here we deduce that the right-hand border and the lower border are the starting places for the shadowing procedure.

(d) For every obstacle contiguous to the lower side, determine its highest point and project a horizontal line to the right until it intersects the right-hand side of the network, even if this means running over some obstacle. The set of modules limited by this line, plus the obstacle and the two sides of the network constitute the "shadow" of the obstacle, which is considered a zone forbidden to the path-tracing procedure.

(e) Repeat the same procedure of (d) for obstacles attached to the right-hand side, projecting the shadow vertically from the left-most point until it intersects the lower side of the network.

(f) Repeat the horizontal and vertical shadowing procedures as described in (d) and (e) for every obstacle now adjoining any of the shadowed zones. Adjoining means having at least one vertex in common.

(ii) When no more shadowing is possible because the rest of the obstacles are detached from the shadowed zones, the path is traced following the assigned priorities and considering both the obstacles and shadowed zones as obstacles.

In Fig. 25 the preceding method has been applied to the problem of connecting modules A and B through the set of obstacles indicated by C through S. The elimination and tracing procedures are explained below, with reference to steps (a) through (f):

(a) The starting module is determined by the lowest row coordinate. A priority pair is set arbitrarily: (V,H). Complete specification: $(V_S H)$.

(b) Module B is to the left of A, therefore the sequence of directions is vertical down and horizontal left.

(c) The sequence of directions can be represented as: $\leftarrow\!\!\downarrow$. Therefore the right-hand side and the lower borders are the starting places for the shadowing procedure.

(d) In Fig. 25, obstacles R and S are contiguous to the lower side. Thus their high points project shadows to the right indicated by (1) in Fig. 26, extending to the right side of the network.

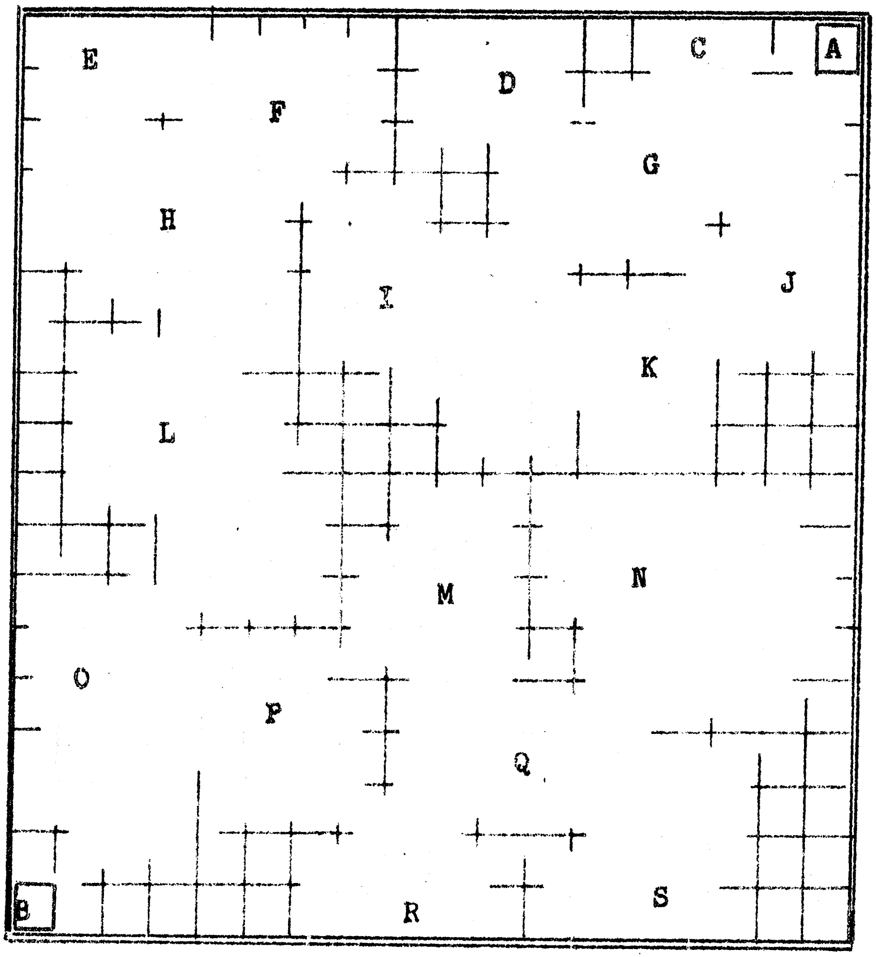


Fig. 25. Maze problem.

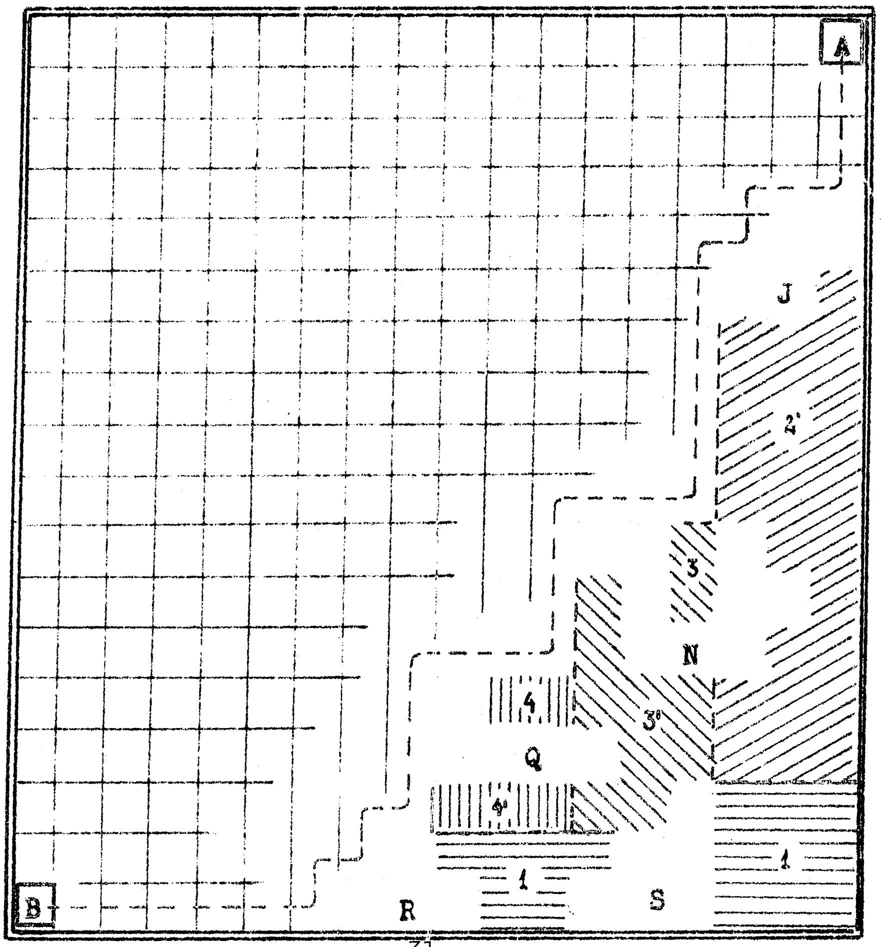


Fig. 26. First solution with priority V_sH.

(e) Obstacle J is contiguous to the right border, and therefore it projects shadow 2' vertically extending to the lower border. Here it is not necessary to cover up region 1 in the shadow of S because it is already eliminated as a possible region for path tracing.

(f) Obstacle N is now contiguous to a shaded zone, and so it projects a horizontal shadow 3 and a vertical shadow (3'). Q is now contiguous to a shaded zone. It projects shadow 4 and 4'. The shadowing possibilities are now exhausted.

The limitation to non-regressive paths makes the route pattern very sensitive to small changes in the shape of obstacles.

If all types of paths were admissible, a small change in the shape of one of the obstacles would probably imply only a small detour of the original path, but with non-regressive paths not only the shape but the adjacency of other obstacles to the modified shadow intervenes to modify radically the shape of the path. This means that a small variation in the configuration of one obstacle has a local influence plus a "long distance" effect according to whether the new shadow profile becomes contiguous or ceases to be contiguous to some other obstacle.

In order to illustrate the wide variation induced in the path route by minor changes in the shape of the obstacles, the same basic pattern of Fig. 25 has been used in Fig. 27 with the addition of a one-module obstacle G! The priority is still (V,H), but the exit from A is impossible in the vertical direction, and consequently the path is started in the direction of secondary priority, H.

If now a second one-module obstacle is added, like I' in Fig. 28, the path again is modified substantially; this time by the inclusion of obstacles L and O and their shadows in the forbidden zone. Figure 28. Here, the new

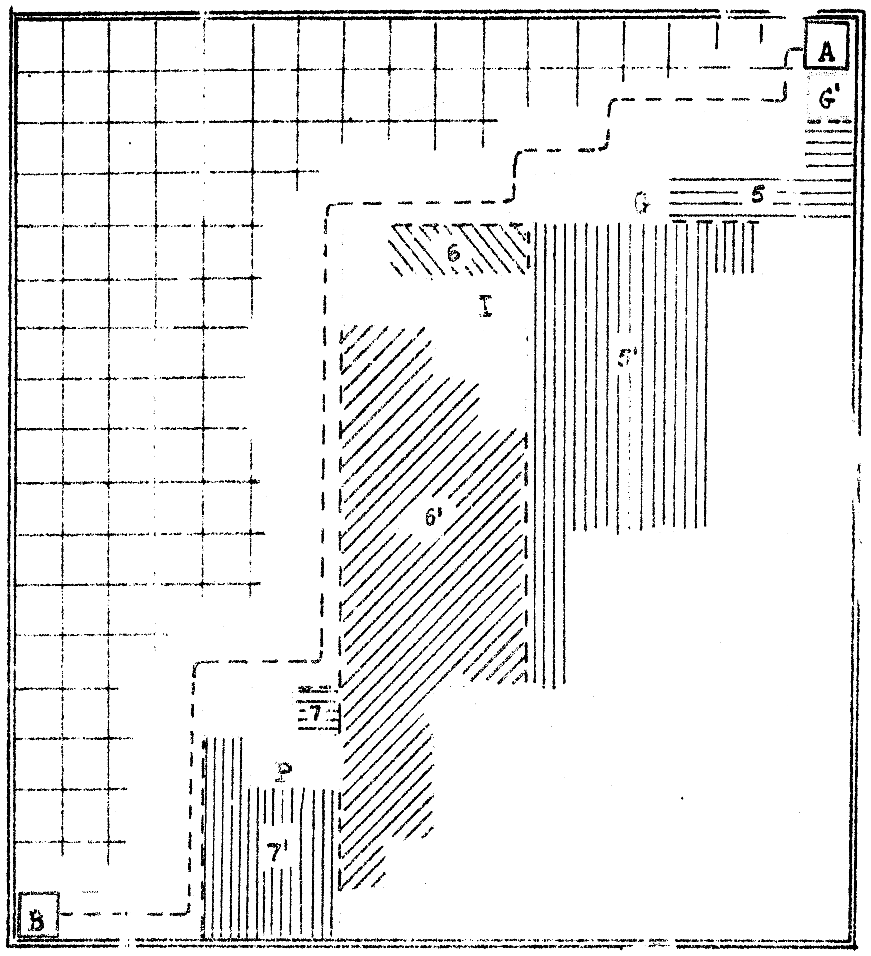


Fig. 27. Change in path route induced by the inclusion of G'.

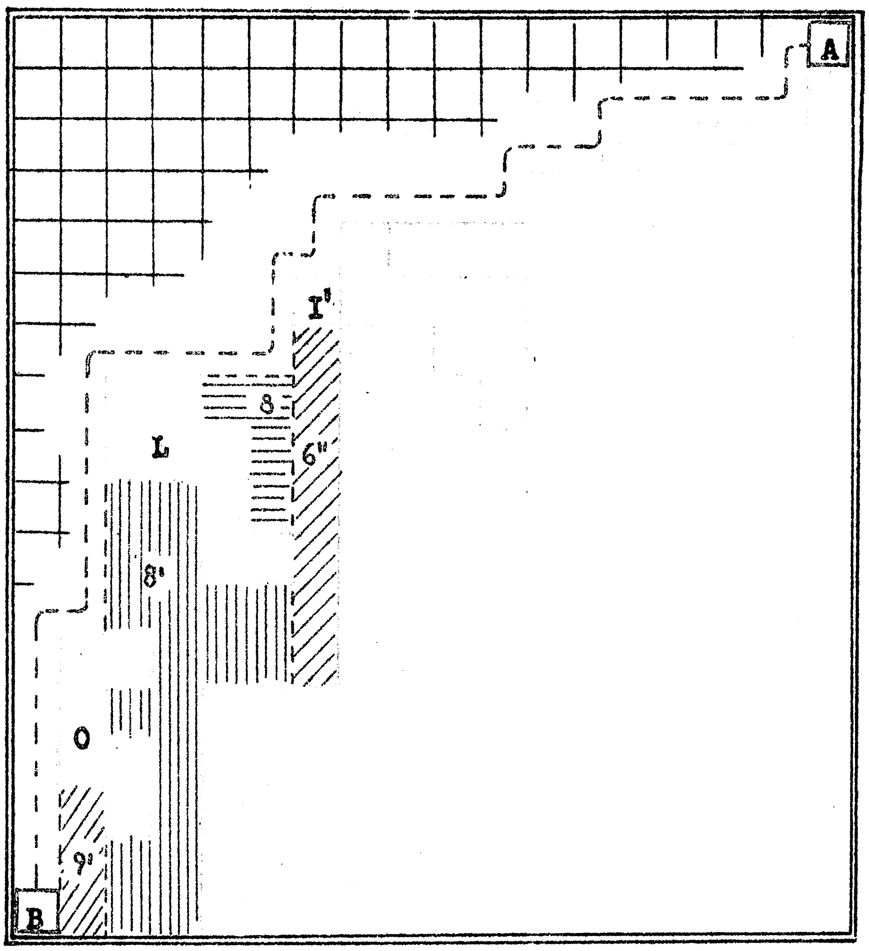


Fig. 28. Change in path route induced by the inclusion of I'.

additional shadow projected by I' is enough to cause obstacle L to become contiguous, which in turn brings O into the forbidden zone. The new path, as a result, is very different from the one in the original problem (Fig. 26).

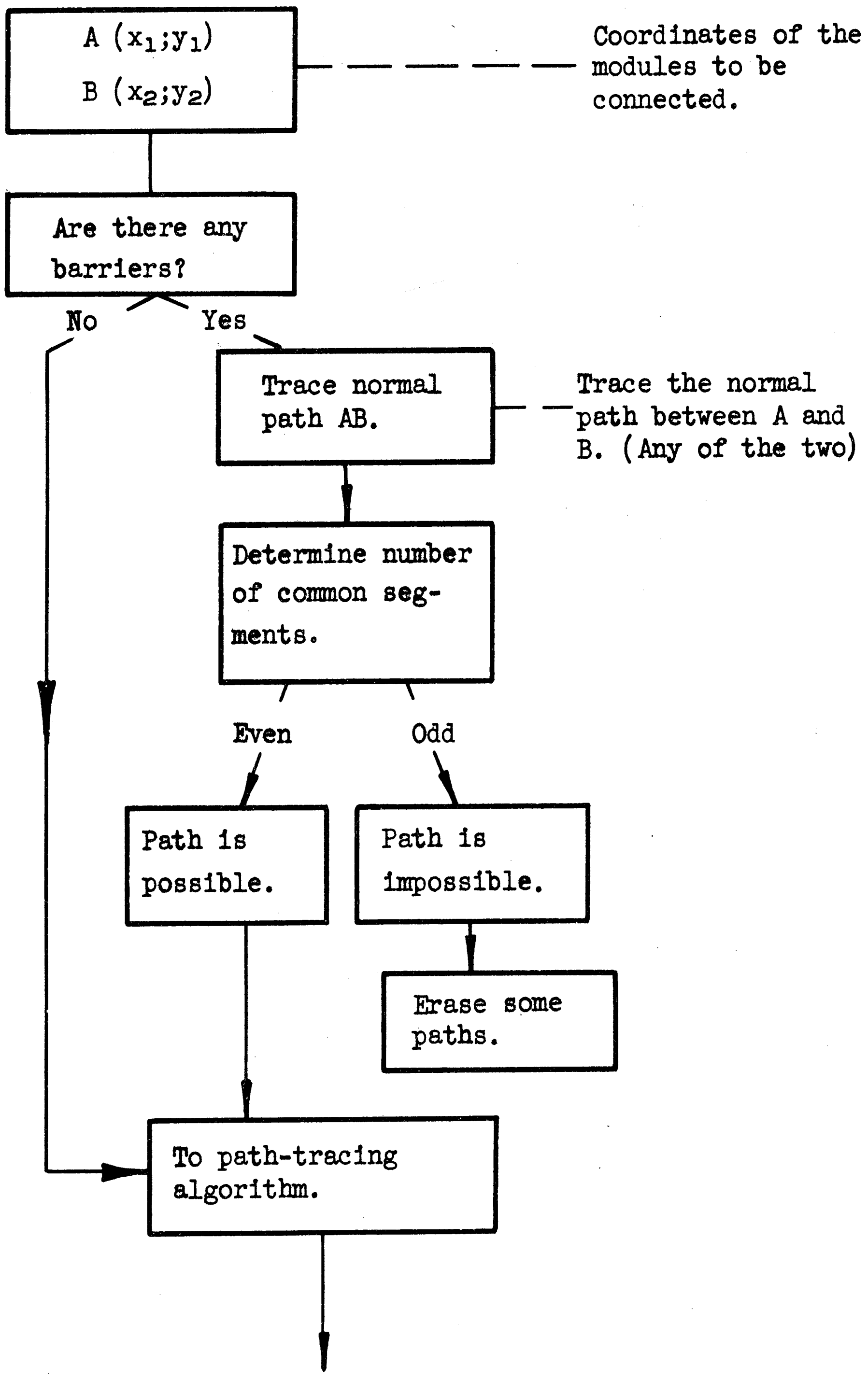
6. FLOW DIAGRAMS FOR THE IMPLEMENTATION OF THE BARRIER DETECTION AND PATH-TRACING PROCEDURES

A. Adapting the Procedures for Computer Solutions

The implementation of the barrier detection procedure does not present any difficulty. On the contrary, the implementation of the path-tracing procedure, as explained in Section 5, requires the use of pattern recognition techniques since it depends on the determination of such features as the left-most and uppermost corner of an irregular pattern of modules consituting an obstacle. In order to circumvent this difficulty, the obstacles are not treated as patterns of modules, but each individual module in the pattern is treated as a "unit obstacle." Therefore the obstacles referred to in previous sections are now conglomerates of "unit obstacles." These unit obstacles are also now treated individually as independent contiguous obstacles. The procedure is now applicable in the same way as before, since the contiguity of the unit obstacles assures the final treatment of the whole pattern by the shadowing method.

B. Flow Diagram for the Detection of Barriers

Figure 29 shows the flow diagram for the detection of barriers. It starts with the given coordinates of the two modules to be connected. Then the existence of any barrier is checked by reference to the barrier list. If no barriers are present, the whole algorithm is skipped and the path-tracing algorithm is entered directly. If there is a barrier, then the normal path connecting the two modules is traced, and the number of common segments be-



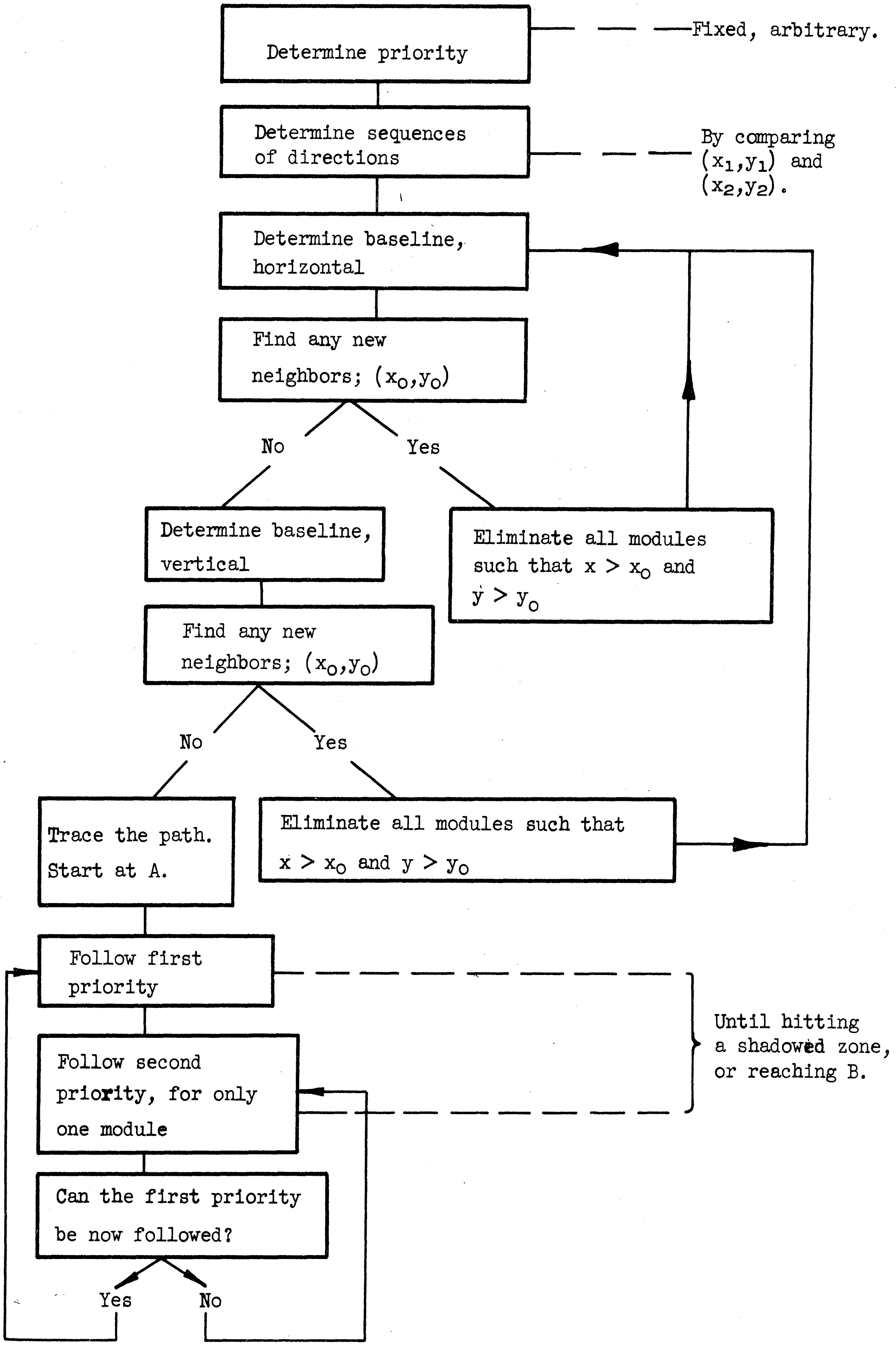
A $(x_1;y_1)$
B $(x_2;y_2)$
Coordinates of the modules to be connected.
Are there any barriers?
No
Yes
Trace normal path AB.
Trace the normal path between A and B. (Any of the two)
Determine number of common segments.
Even
Odd
Path is possible.
Path is impossible.
Erase some paths.
To path-tracing algorithm.

Fig. 29. Flow diagram for the detection of barriers.

tween the normal path and the barrier proper is determined. If there is an even number of common segments, then the two modules are in separate regions and there is no possibility of a path. If there is an even number including zero, of common segments then the two modules lie in the same region with respect to the barrier and a path is still possible. Therefore, the path-tracing algorithm is entered.

C. Flow Diagram for the Path-Tracing Algorithm

Figure 30 shows the flow diagram for the path-tracing algorithm. It starts with the determination of the sequence of directions. This is derived from the arbitrarily assigned priority and a comparison of the relative addresses of the two modules to be connected. Next, the baseline is determined. In the initial step, the baseline will be constituted by two sides of the original network, but later in the iteration it will be formed by the horizontal and vertical projection of the uppermost and leftmost (or rightmost) corner in the obstacle being considered. Any module adjoining (having at least one corner in common) the baseline is considered the new neighbor, and its coordinates are labeled (x_o, y_o). Then all modules whose $\underline{x}, \underline{y}$ coordinates are greater than x_o and y_o respectively, are eliminated. After the elimination or "shadowing" of these modules, a new baseline is determined and the procedure is repeated. If there are no new neighboring modules, the other half of the baseline is considered and a similar procedure is followed. If there are no neighbors to any of the two halves of the baseline, then the shadow procedure is terminated and the path can be traced.



Determine priority
Fixed, arbitrary.
Determine sequences of directions
By comparing (x_1,y_1) and (x_2,y_2).
Determine baseline, horizontal
Find any new neighbors; (x_0,y_0)
No
Yes
Determine baseline, vertical
Eliminate all modules such that $x > x_0$ and $y > y_0$
Find any new neighbors; (x_0,y_0)
No
Yes
Trace the path. Start at A.
Eliminate all modules such that $x > x_0$ and $y > y_0$
Follow first priority
Follow second priority, for only one module
Until hitting a shadowed zone, or reaching B.
Can the first priority be now followed?
Yes
No

Fig. 30. Flow diagram for path tracing.

REFERENCES

1. Moore, E. F., "Shortest Path Through a Maze," _Annals of the Computation Laboratory of Harvard University_, Harvard University Press, Cambridge, Mass., Vol. 30, pp. 285-292, 1959.

2. Lee, C. Y., "An Algorithm for Path Connections and Its Applications," _IRE Trans. on E.C._, Vol. EC-10, No. 3, pp. 346-365, September, 1961.

3. Loberman, H., and Weinberger, A., "Formal Procedures for Connecting Terminals with a Minimum Total Wire Length," _ACM Proc._, Vol. 4, 1957, p. 428.

4. Von Neumann, J., "Probabilistic Logics," _Automata Studies_, Princeton University Press, 1956.

5. Miller, R. E., and Selfridge, J. L., "Maximal Paths on Rectangular Boards," _IBM Journal_, Vol. 4, No. 5, p. 479, November, 1960.

6. Wilcox, R., and Mann, W., _Redundancy Techniques for Computing Systems_, Spartan Books, 1962.

7. Hald, A., _Statistical Theory with Engineering Applications_, Wiley and Sons, 1952.

8. Holland, J., "Iterative Circuit Computers," _Proc. W.J.C.C._, May, 1960, p. 259.

9. Ungar, S. H., "A Computer Oriented Towards Spatial Problems," _Proc._ IRE, Vol. 46, October, 1958, p. 1749.

10. Newell, A., "On Programming a Highly Parallel Machine to be an Intelligent Technician," _Proc. W.J.C.C._, May, 1960, p. 267.