An Algorithm for Path Connections and Its Applications*

C. Y. LEE†, MEMBER, IRE

Summary—The algorithm described in this paper is the outcome of an endeavor to answer the following question: Is it possible to find procedures which would enable a computer to solve efficiently path-connection problems inherent in logical drawing, wiring diagramming, and optimal route finding? The results are highly encouraging. Within our framework, we are able to solve the following types of problems:

1) To find a path between two points so that it crosses the least number of existing paths.
2) To find a path between two points so that it avoids as much as possible preset obstacles such as edges.
3) To find a path between two points so that the path is optimal with respect to several properties; for example, a path which is not only one of those which cross the fewest number of existing paths, but, among these, is also one of the shortest.

The minimal-distance solution has been programmed on an IBM 704 computer, and a number of illustrations are presented. The class of problems solvable by our algorithm is given in a theorem in Section III. A byproduct of this algorithm is a somewhat remote, but unexpected, relation to physical optics. This is discussed in Section VI.

I. Introduction

IN processing information consisting of patterns, rather than numbers or symbols, on a digital computer, we may wish to know how a computer, without sight and hearing, can be made to deal competently with situations which appear to require coordination, insight, and perhaps intuition. It is not our intention to consider the general problem of pattern detection and recognition by machines. We will consider rather the following simpler, and therefore perhaps more basic, problem in pattern processing by machines.

Let a pattern of some sort be presented to a machine. We then want the machine to construct some optimal path subject to various constraints imposed by the pattern. The problem is to find efficient procedures, which, if followed by the machine, would lead to an optimal solution.

Ideally, many situations would fall within this description. We might present to the machine a map of Manhattan and ask it to find the shortest-time route between, say, the United Nations and Yankee Stadium, using only public transportation. With sufficient care, it is possible to make a problem such as this unambiguous. In most cases, however, it would be too great a struggle just to present the problem in a way that is completely and consistently stated. For this reason, we have decided to present an abstract model in Section II. Based on this model, we will consider a class of well-defined optimal path problems. A general procedure for solving this class of problems will then be given in Section III.

Within this class of problems is the *shortest-route problem* on which there has been earlier definitive work. Algorithms for finding shortest paths have been given by Dantzig,[1] Ford and Fulkerson,[2] Moore[3] and Prim.[4] The minimal-distance illustrations (Section V) make use of one of Moore's algorithms, which is a specialization of algorithm A given in Section III. These experiments were tried out before the abstract model was completed. Once we have at our disposal the abstract model, it came as a pleasant surprise that problems such as the minimal-crossing and minimal edge-effect problems, which had appeared difficult to us previously, all yielded immediately to algorithm A. The possibility of joint minimization is also a direct consequence of algorithm A. A further outcome was the "diffraction" patterns. These experiments would not have been attempted if we had not noticed the patterns obtained in the minimal-distance experiments.

II. An Abstract Model and the Path Problem

A. $\mathfrak{C}$-Space, an Abstract Model

Let C be a set of elements called cells: $C = \{c^1, c^2, \cdots\}$. For each cell c^i in C, there is defined a subset of C called a *1-neighborhood* $N(c^i)$ of c^i: $N(c^i) = \{c_1{}^i, c_2{}^i, \cdots, c_n{}^i\}$. The following rules hold for 1 neighborhoods:

N1) Every 1-neighborhood has in it exactly n cells, where n, $n \geq 1$, is some predetermined number depending on the specific model involved.

N2) If $c^j \in N(c^i)$, then $c^i \in N(c^j)$. We will call the function N with domain C and range subsets of C the *1-neighborhood function.*

Together with the 1-neighborhood functions, there are n functions $d_1, d_2, \cdots, d_n$ on C to C defined as follows: If c^i is any cell in C and $N(c^i) = \{c_1{}^i, c_2{}^i, \cdots, c_n{}^i\}$, then

$$d_k(c^i) = c_k{}^i, \qquad k = 1, 2, \cdots, n.$$

* Received by the PGEC, December 2, 1960. This material was presented as part of the University of Michigan Engineering Summer Session, Ann Arbor, June 19–30, 1961.

† Bell Telephone Labs., Inc., Whippany, N. J.

[1] G. B. Dantzig, "Maximization of a Linear Function of Variables Subject to Linear Inequalities," Cowles Commission; 1951.

[2] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Can. J. Math.*, vol. 8, pp. 399–404; 1956.

[3] E. F. Moore, "Shortest path through a maze," in "Annals of the Computation Laboratory of Harvard University," Harvard University Press, Cambridge, Mass., vol. 30, pp. 285–292; 1959.

[4] R. C. Prim, "Shortest connection networks and some generalizations," *Bell Sys. Tech. J.*, vol. 36, pp. 1389–1401; November, 1957.

That is, $d_k(c^i)$ is the kth coordinate cell in the 1-neighborhood of c^i.

Let S be a finite set of symbols: $S=\{s^1, s^2, \cdots, s^m\}$. S is called the *alphabet* of space $\mathcal{C}$.

Let Γ be a map on C to $C\times S$. That is, for every $c^i\in C$, $\Gamma(c^i)=(c^i, s^i)$, $s^i\in S$. In general we will write $\Gamma(c^i)=(c^i, s(c^i))$. Therefore, to every cell $c^i\in C$ is associated some symbol $s(c^i)\in S$. The map Γ gives then the cell-symbol configuration for a particular path problem. Generally speaking, we may keep in mind the analogy that each function Γ corresponds to some sort of a street map for a particular city or town, and that the path problem is to find an optimal path from one point to another in that city or town.

Let c^i, c^j be two distinct cells in C. By a *path* $p(c^i, c^j)$ is meant a set of cells called a *chain:* $p(c^i, c^j)=\{c^0=c^i, c^1, c^2, \cdots, c^m=c^j\}$ such that $c^{i+1}\in N(c^i)$ for $i=0, 1, \cdots, m-1$. By $\pi(c^i, c^j)$ is meant the set of all paths $p(c^i, c^j)$ between cell c^i and cell c^j.

Let M be a map called an *admission map* with domain $\pi(c^i, c^j)$ and range the two-element set $\{0, 1\}$. Any path $p(c^i, c^j)$ such that $M(p(c^i, c^j))=1$ is said to be an *admissible path.* Otherwise, $p(c^i, c^j)$ is said to be *inadmissible.* The set of all admissible paths will be denoted by $\pi^*(c^i, c^j)$.

The quintuple (C, S, N, Γ, M) is called a $\mathcal{C}$*-space.*

B. The Path Problem

Let F be a vector of r functions $(f_1, f_2, \cdots, f_r)$ where each function f_i, $i=1, 2, \cdots, r$, is on $\pi^*(c^i, c^j)$, the set of admissible paths, to I, the set of non-negative integers. A path $p^1(c^i, c^j)$ of $\pi^*(c^i, c^j)$ is said to be *minimal with respect to* f_1 if

$$f_1(p^1(c^i, c^j)) \leq f_1(p(c^i, c^j))$$

for all $p(c^i, c^j)\in\pi^*(c^i, c^j)$. A path $p^{12}(c^i, c^j)$ is said to be *minimal with respect to* (f_1, f_2) if

$$\text{(i)}\quad p^{12}(c^i, c^j)\in P^1(c^i, c^j)$$

where $P^1(c^i, c^j)$ is the set of all paths in $\pi^*(c^i, c^j)$, which are minimal with respect to f_1; that is,

$$P^1(c^i, c^j)=\{p^1(c^i, c^j)\mid f_1(p^1(c^i, c^j))\leq f_1(p(c^i, c^j)) \text{ for all } p(c^i, c^j)\in\pi^*(c^i, c^j)\};$$

and

$$\text{(ii)}\quad f_2(p^{12}(c^i, c^j))\leq f_2(p(c^i, c^j)) \text{ for all } p(c^i, c^j)\in P^1(c^i, c^j).$$

Therefore, $p^{12}(c^i, c^j)$ is minimal with respect to (f_1, f_2) if among all paths minimal with respect to f_1, $p^{12}(c^i, c^j)$ is also minimal with respect to f_2. In a similar way, we may define a path $p^{12\cdots r}(c^i, c^j)$ which is minimal with respect to $(f_1, f_2, \cdots, f_r)$.

The path problem we are considering is the following:

P) Path problem: Given a $\mathcal{C}$-space (C, S, N, Γ, M), a vector $F=\{f_1, f_2, \cdots, f_r\}$, an initial cell c^* and a final cell c^{**}, find an admissible path $p^{12\cdots r}(c^*, c^{**})$ which is minimal with respect to $(f_1, f_2, \cdots, f_r)$.

In the following section we will show an algorithm which solves the path problem P for a certain set of vectors F.

III. A Search and Trace Procedure

A. Montone Functions and Monotone Vectors

Let a $\mathcal{C}$-space (C, S, N, Γ, M) be given. A function f on $\pi^*(c^i, c^j)$ to the set of non-negative integers I is said to be *monotone* if for every path $p(c^i, c^j)$, we have the inequality

$$f(p(c^i, c^k))\leq f(p(c^i, c^j)),$$

where $p(c^i, c^k)$ is any subpath of $p(c^i, c^j)$. A vector F of monotone functions $(f_1, f_2, \cdots, f_r)$ is said to be a *monotone vector.*

B. An Algorithm

Let c^* and c^{**} be the initial and final cells in a $\mathcal{C}$-space (C, S, N, Γ, M). The main procedure will be given in terms of a number of subprocedures.

D1) *Cell list:* A *cell list* L is an ordered list containing names of cells.

D2) *Cell mass:* With each cell in list L will be associated an r-tuple called a *cell mass* $(m_1, m_2, \cdots, m_r)$. The cell masses are ordered lexicographically. Thus, if $m=(m_1, m_2, \cdots, m_r)$, and $m'=(m_1', m_2', \cdots, m_r')$ are two cell masses, $(m_1, m_2, \cdots, m_r)<(m_1', m_2', \cdots, m_r')$ if $m_i=m_i'$, $i=0, 1, \cdots, k$, but $m_{k+1}<m'_{k+1}$; $0\leq k<r$.

D3) *Chain coordinate:* Associated with each cell in the list L is also a *chain coordinate* which is one of the 1-neighborhood coordinate functions d_k.

D4) *Auxiliary list* L_1*:* An auxiliary cell list L_1 is provided for momentary storage of names of cells.

R1) *Procedure for constructing auxiliary list* L_1: Let c be a cell in list L. By a path $p(c^*, c^i, c^{**})$ we mean a path $p(c^*, c^{**})$ of which $p(c^*, c^i)$ is a subpath. A cell c^i is said to be *admissible* if $p(c^*, c^i, c^{**})$ is an admissible path and if the cell mass for c^i has not yet been determined. Let $\{c^i\}\in N(c)$ be the set of all admissible cells in $N(c)$. Append to list L_1 the set $\{c^i\}$. L_1 is constructed by repeating this process for every entry c in L, under the condition that a cell should not be listed more than once. L_1 is therefore the set of all distinct cells c^i such that c^i is an admissible cell in $N(c)$ for some cell c in the list L.

R2) *Procedure for assigning cell masses and chain coordinates:* Let c^i be a cell in L_1. A *possible cell mass* for c^i is determined as follows: For each $c^j\in N(c^i)$ whose cell mass has been determined, construct an r-tuple,

$$(f_1(p(c^*, c^j, c^i)), \cdots, f_r(p(c^*, c^j, c^i))).$$

Now apply rule R3 below to find a $c^{j_0}\in N(c^i)$, for which this r-tuple is a minimum. A possible cell mass for c^i is then the r-tuple $(f_1(p(c^*, c^{j_0}, c^i)), \cdots, f_r(p(c^*, c^{j_0}, c^i)))$, and a possible chain coordinate for c^i is d_k where $d_k(c^i)=c^{j_0}$.

Next, find possible cell masses for all $c^i \in L_1$. Among the cells in L_1, let $\{c^l\}$ be the set of cells whose possible cell masses are a minimum. We then assign to each $c^l \in \{c^l\}$ the same cell mass and the same chain coordinate as its possible cell mass and its possible chain coordinate.

The cells whose possible cell masses are not minimal are no longer considered, and the list L_1 is cleared.

R3) *Selection rule:* In R2, a cell c^i and a selected subset $\{c^j\}$ of $N(c^i)$ were given. The rule which was invoked to select one c^{j_0} from the set $\{c^j\}$ is called a *selection rule.*

R4) *Procedure for updating list L:* Let $\{c^l\}$ be the set of cells whose cell masses had been determined by R2. Append first to list L the set $\{c^l\}$. Next examine each cell c in L to see if the cell masses of all its admissible 1 neighbors had been determined. If so, erase c from the list L.

R5) *Initialization:* Cell c^* is given the cell mass $(0, 0, \cdots, 0)$. The list L contains initially the single entry c^*. The list L_1 is initially cleared.

With the aid of rules R1–R5, we may now state the main procedure.

A: The Search and Trace Algorithms:

A1: Search algorithm: Apply rule R5 for initialization. Apply rules R1, R2, R4 to entries of L until either c^{**} appears in L or the list L has been exhausted. In the former case, we will proceed to the trace algorithm A2. In the latter case, there is no admissible path $p(c^*, c^{**})$ in the $\mathfrak{C}$ space.

A2: Trace algorithm: Begin at c^{**}, follow the chain coordinates until c^* is reached. This determines a unique path $p(c^*, c^{**})$.

C. The Procedure Applied to the Path Problem

Let a $\mathfrak{C}$ space (C, S, N, Γ, M), a vector $F = (f_1, f_2, \cdots, f_r)$ and an initial cell c^* be given. A cell c^i which is reached from c^* after exactly t application of rules R1 to R4 is said to have a *chain index* l (c^*, c^i) of t from c^*. We will begin with two lemmas.

Lemma 1: Let F be a monotone vector. Let c^i and c^j be two cells with cell masses $m(c^i)$ and $m(c^j)$ and chair indexes $l(c^*, c^i)$ and $l(c^*, c^j)$, respectively. If $m(c^i) < m(c^j)$, then $l(c^*, c^i) < l(c^*, c^j)$.

Proof: Assume $l(c^*, c^i) \geqq l(c^*, c^j)$. This means that after $l(c^*, c^j)$ applications of rules $R1$ to $R4$, the cell mass $m(c^j)$ has been determined, but the cell mass $m(c^i)$ has not. By the nature of rule $R2$, the cell masses are obtained by evaluating the vector F for the paths in question. Since F is monotone by hypothesis, $m(c^i) \geqq m(c^j)$ and the lemma follows.

Lemma 2: Let F be a monotone vector. Let $p(c^*, c^{**})$ be any admissible path from c^* to c^{**}. Then $m(c^{**}) \leqq F(c^*, c^{**})$.

Proof: Let $p(c^*, c^{**})$ consist of the chain of cells

$$p(c^*, c^{**}) = \{c^* = c^0, c^1, c^2, \cdots, c^q, c^{**} = c^{q+1}\}.$$

We will let the number of cells contained in a path, excluding the initial cell, be called the *path length* of that path. For the path $p(c^*, c^{**})$, therefore, the path length is $q+1$.

For admissible paths of length 1, it follows from rule $R2$ that $m(c^{**}) \leqq F(c^*, c^{**})$. Let us now assume valid the induction hypothesis that $m(c^{**}) \leqq F(c^*, c^{**})$ for all admissible paths of path length not greater than q.

Consider the subpath $p(c^*, c^q)$ of $p(c^*, c^{**})$. By the induction hypothesis,

$$m(c^q) \leqq F(c^*, c^q).$$

Let us now suppose $m(c^{**}) > F(c^*, c^{**})$. Since F is monotone, we then have the inequalities

$$m(c^q) \leqq F(c^*, c^q) \leqq F(c^*, c^{**}) < m(c^{**}).$$

Since now $m(c^q) < m(c^{**})$, it follows from Lemma 1 that $l(c^*, c^q) < l(c^*, c^{**})$. Therefore the cell mass $m(c^q)$ is determined before the cell mass $m(c^{**})$ is determined.

From rule $R2$ it follows that as soon as $m(c^q)$ is determined, the cell c^q becomes a member of cell list L. Since by rule $R2$,

$$m(c^{**}) = \begin{cases} \min \{F(p(c^*, c^i, c^{**})) \mid c^i \in N(c^{**})\} \\ \text{over all } c^i \text{ for which } m(c^i) \text{ is defined,} \end{cases}$$

and since c^q is one of such cells c_i, it follows that this minimum cannot be greater than $F(p(c^*, c^{**}))$. This contradicts our earlier supposition, and the lemma follows.

The basic result is embodied in the following:

Theorem: Let a $\mathfrak{C}$-space (C, S, N, Γ, M) be given. Let P be a path problem with respect to a vector F. If F is monotone, then algorithm A yields a path $\bar{p}(c^*, c^{**})$ satisfying P.

Proof: Let $p(c^*, c^{**})$ be any admissible path from c^* to c^{**}. Since F is monotone, we may apply Lemma 2 to get $m(c^{**}) \leqq F(p(c^*, c^{**}))$. It follows from rule R2 that

$$m(c^{**}) = (f_1(\bar{p}(c^*, c^{**})), \cdots, f_r(\bar{p}(c^*, c^{**}))).$$

Also, by definition,

$$F(p(c^*, c^{**})) = (f_1(p(c^*, c^{**})), \cdots, f_r(p(c^*, c^{**}))).$$

The theorem therefore follows by the lexicographic ordering of these r-tuples.

IV. Applications

In the applications to be discussed here, we will consider the set C to be a set of squares in the plane with the usual 1 neighborhoods as shown in Fig. 1. We will let the alphabet set S consist of the following:

1) Digits from 0 to 9.
2) All letters of the English alphabet.
3) The symbols: +, −, ·, ,, /, *, —, └, ┐, ┘, ┌, (,), →, ←, ↑, ↓, blank.

The 1-neighborhood function N and the coordinate functions d_1, d_2, d_3, and d_4 are defined as follows: Given a cell c^i, $d_1(c^i)$ is the cell to the right of c^i, $d_2(c^i)$ is the cell above c^i, $d_3(c^i)$ is the cell to the left of c^i and $d_4(c^i)$

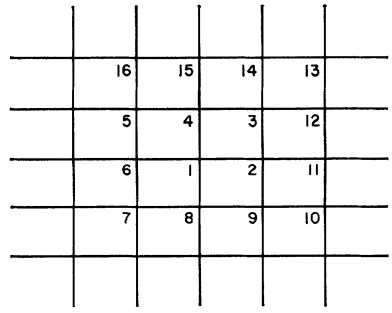

Fig. 1—The set C of squares in the plane.

is the cell below c^i. $N(c^i)$ is then the set $\{d_1(c^i), d_2(c^i), d_3(c^i), d_4(c^i)\}$.

Let a function Γ be given, so that to each cell $c^i \in C$ is associated a symbol $s(c^i) \in S$. The function M is given as follows:

Let $p(c^*, c^{**}) = \{c^0 = c^*, c^1, \cdots, c^{n-1}, c^n = c^{**}\}$ be a path from c^* to c^{**}. $p(c^*, c^{**})$ is admissible, *i.e.*, $M(p(c^*, c^{**})) = 1$, if

1) $s(c^i) =$ blank, —, or $|$ for $i = 1, 2, \cdots, n-1$.
2) $s(c^{i+1}) \neq$ — whenever $c^{i+1} = d_1(c^i)$ or $c^{i+1} = d_3(c^i)$, $i = 0, 1, \cdots, n-1$.
3) $s(c^{i+1}) \neq |$ whenever $c^{i+1} = d_2(c^i)$ or $c^{i+1} = d_4(c^i)$, $i = 0, 1, \cdots, n-1$.

Thus, except for the function Γ which depends on the application in question, the specialization of the $\mathfrak{C}$-space to our applications has been fully described.

A. A Minimal-Crossing Problem

Given a set of squares in the plane, the problem of finding a path $p(c^*, c^{**})$ from c^* to c^{**} such that $p(c^*, c^{**})$ crosses over the fewest number of existing paths is called the minimal-crossing problem. We will formulate the minimal-crossing problem as a path problem in some appropriate $\mathfrak{C}$ space and then solve it with the aid of algorithm A.

For this problem the vector F would consist of a single function f given as follows:

F1) $f(p(c^*, c^*)) = 0$.

F2) If $s(c^i) =$ blank, then

$$f(p(c^*, c^i)) = \begin{cases} \min \{f(p(c^*, c^i)) \mid c^i \in N(c^i)\} \text{ over all } c^i \\ \quad \text{for which } f(p(c^*, c^i)) \text{ has been defined;} \\ \text{undefined otherwise.} \end{cases}$$

F3) If $s(c^i) =$ —, then

$$f(p(c^*, c^i)) = \begin{cases} (\min\{f(p(c^*, d_2(c^i))), f(p(c^*, d_4(c^i)))\}) + 1 \\ \quad \text{if either one of the values of } f \text{ is defined;} \\ \text{undefined otherwise.} \end{cases}$$

F4) If $s(c^i) = |$, then

$$f(p(c^*, c^i)) = \begin{cases} (\min \{f(p(c^*, d_1(c^i))), f(p(c^*, d_3(c^i)))\}) + 1 \\ \quad \text{if either one of the values of } f \text{ is defined;} \\ \text{undefined otherwise.} \end{cases}$$

We assert first that f is monotone. This is so by the iterative nature of the definition of f; the value of f never decreases as a path increases in length. Hence, we may apply the basic theorem to get:

Corollary 1: Algorithm A solves the minimal-crossing problem.

Example 1: Consider the cell configuration given in Fig. 2. The path AA consisting of the chain of cells $\{6, 5, 16, 15, 14, 3, 2, 11, 28\}$ is already present. We wish to find a minimal-crossing path from c^* (cell 18), to c^{**} (cell 13).

Applying algorithm A, we find that list L consists of the single entry $\{18\}$ to begin with. List L_1 therefore has in it entries $\{5, 17, 19\}$. Note that cell 39 is not an admissible 1-neighbor of c^*. The possible cell masses for cells $\{5, 17, 19\}$ are 1, 0 and 0, respectively. Hence, by rule R2, cells 17 and 19 are assigned cell mass 0 and are appended to the list L. Moreover, the chain coordinates of cells 17 and 19 are respectively d_4 and d_2. The cell masses and chain coordinates for these cells are properly denoted in Fig. 2. Thus, in cell 17, we have the pair $(\downarrow, 0)$, meaning that the chain coordinate is d_4 (*i.e.*, downward) and the cell mass is 0.

Applying rules R1 to R4 again, we find that list L_1 has in it now the entries $\{5, 20\}$. This is so since cells 16, 36, 38, 39, 6 and 40 are all not admissible. The possible cell masses for cells 5 and 20 are respectively 1 and 0. Thus, the cell mass for cell 20 is 0, the chain coordinate for cell 20 is d_2, and cell 20 is appended to list L. Moreover, cells 17 and 19 are erased from list L.

Fig. 3 shows the cell mass and chain coordinates for all the cells reached by the application of Algorithm A. The trace algorithm then traces out a solution path $p(c^*, c^{**})$ as shown. The boundary cells have been omitted in Fig. 3.

Example 2: Consider again the cell configuration in Fig. 2. In this case, however, we stipulate that it costs 3 units to cross a —, but 1 unit to cross a $|$. That is, the definition F3 is changed to read:

F3′. If $s(c^i) =$ —, then

$$f(p(c^*, c^i)) = \begin{cases} (\min \{f(p(c^*, D_2(c^i))), f(p(c^*, D_4(c^i)))\}) + 3 \\ \quad \text{if either one of the values of } f \text{ is defined;} \\ \text{undefined otherwise.} \end{cases}$$

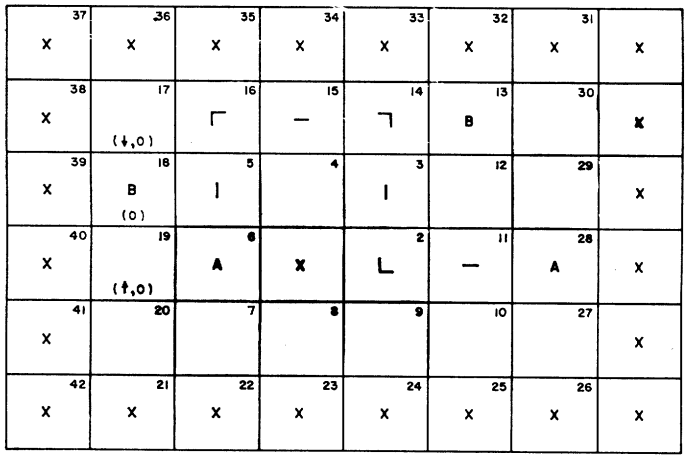

Fig. 2—A minimal-crossing example.

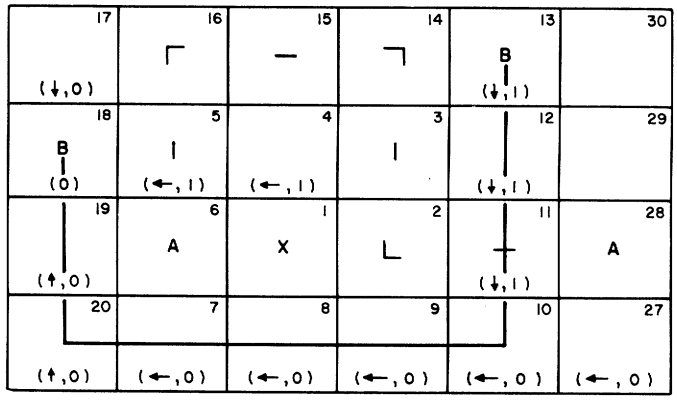

Fig. 3—Cell configuration after algorithm A has been applied to Example 1.

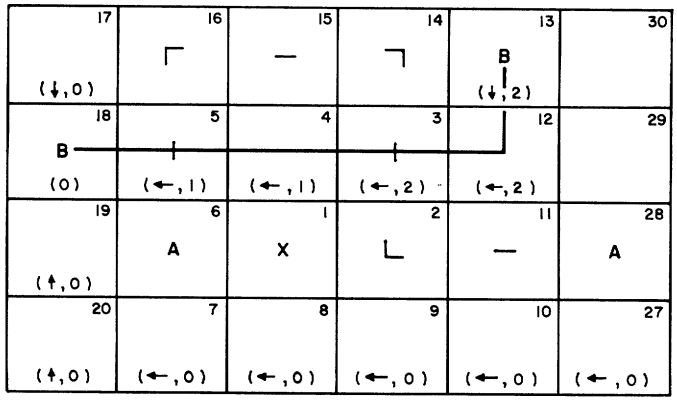

Fig. 4—Cell configuration after algorithm A has been applied to Example 2.

Since F is still monotone, algorithm A may be applied to solve this modified path problem.

In this case, after applying algorithm A, we have the cell configuration and the solution path shown in Fig. 4.

In Example 1, the function f is not only monotone, but grows a single unit at a time. For such functions and for vectors made up of such functions, it is possible to simplify algorithm A to solve the path problem. In Example 2 the function f no longer grows one unit at a time. The machinery of $\mathcal{C}$-space is needed to cope with this more general class of monotone functions.

B. A Minimal-Edge-Effect Problem

Let us begin with the $\mathcal{C}$-space consisting of squares in the plane described earlier. Let an initial cell c^* and a final cell c^{**} be given. We wish now to consider the problem of finding a path from c^* to c^{**} which avoids, as much as possible, any symbol other than —, | and the blank symbol. In other words, we want a path which does not tend to "cling to edges."

For this application, the vector F would again consist of a single function g given as follows:

G1) $g(p(c^*, c^*)) = 0$.

G2) If $s(c^i) =$ blank, then

$$g(p(c^*, c^i)) = \begin{cases} (\min \{g(p(c^*, c^i)) \mid C^i \in N(c^i)\}) + R(c^i) \\ \text{over all } C^i \text{ for which } g(p(c^*, c^i)) \text{ has been defined;} \\ \text{undefined otherwise,} \end{cases}$$

where $R(c^i) =$ the number of cells c^j in $N(c^i)$ in each of which the symbol is neither blank, nor —, nor |.

G3) If $s(c^i) =$ —, then

$$g(p(c^*, c^i)) = \begin{cases} (\min \{g(p(c^*, d_2(c^i))), g(p(c^*, d_4)c^i)))\}) + R(c^i) \\ \qquad \text{if either value of } g \text{ is defined;} \\ \text{undefined otherwise:} \end{cases}$$

G4) If $s(c^i) =$ |, then

$$g(p(c^*, c^i)) = \begin{cases} (\min \{g(p(c^*, d_1(c^i))), g(p(c^*, d_3(c^i)))\}) \\ \qquad + R(c^i) \text{ if either value of } g \text{ is defined;} \\ \text{undefined otherwise.} \end{cases}$$

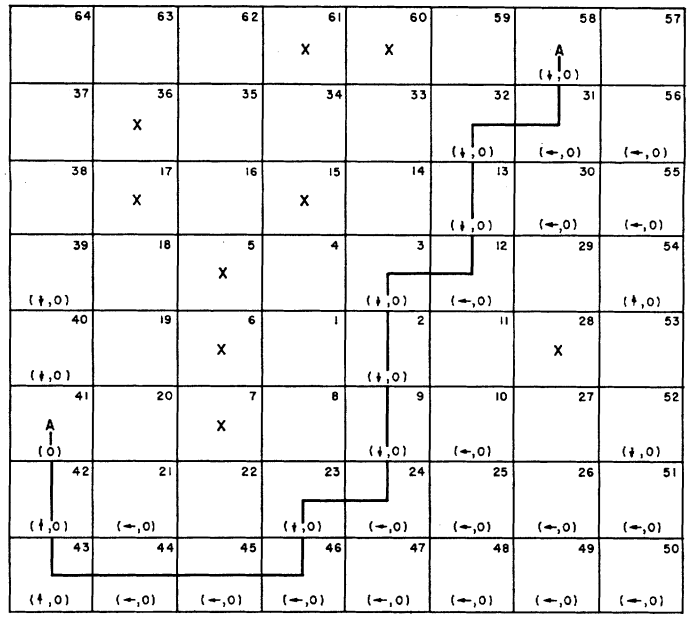

Fig. 5—A minimal edge-effect example.

From this it follows that F is again monotone. Therefore, we have:

Corollary 2: Algorithm A solves the minimal edge-effect problem.

Example 3: Consider the cell configuration given in Fig. 5. We wish to construct a path AA from cell 41 to cell 58 satisfying the path problem with respect to the vector F given by G1 to G4.

To begin with, the list L has in it the single entry $\{41\}$. The list L_1, therefore, has in it entries $\{20, 40, 42\}$. The possible cell masses for these cells are, respectively, 1, 0, 0. Therefore, by rule R2, $m(40) = m(42) = 0$. The chain coordinates for cells 40 and 42 are also determined and are respectively $\downarrow$ and $\uparrow$.

List L is now updated to contain cells $\{41, 40, 42\}$. From this, we get for list L_1 the cells $\{20, 19, 39, 21, 43\}$. Following rule R2, we get therefore $m(39) = m(43) = m(21) = 0$, and also their chain coordinates. By rule R4, cell 42 is erased from list L.

The new list L now has in it cells $\{41, 40, 39, 43, 21\}$. Therefore, cells $\{20, 19, 18, 38, 22, 44\}$ all belong to list L_1. Consider now cell 44. It is clear that $m(44) = 0$. We must, however, invoke the selection rule R3 to get its chain coordinate. We may assume, in this case, that the left neighbor is always preferred over the other 1-neighbors. The chain coordinate for cell 44 then becomes $\leftarrow$.

Continuing in this way, we arrive at the path shown in Fig. 5. We see that the path so constructed avoided all the $\times$'s which may be considered as edges.

C. Joint Minimization

In the last two applications, the vector F in each case consisted of a single function. This was so because we were looking for paths which were minimal with respect to a single property. In the first case, the property was the number of crossings, and in the second, the property was edge effect.

On the other hand, the C-space model was intended to solve joint minimization problems; *i.e.*, the vector F may have several component functions. In order to illustrate the possibility of joint minimization, let us consider a *minimal distance then edge-effect problem.* What we wish to find here is a path which is, first of all, one of the shortest, and secondly, among all shortest paths, this path is also minimal with respect to edge effect. The vector F for this problem has, therefore, two component functions $F = (h, g)$, where h is the distance function and g is the edge-effect function.

The edge-effect function g will be taken to be exactly the same as that defined previously in G1 to G4. The distance function h is given as follows:

H1) The function h satisfies F1, F3, F4.

H2) If $s(c^i) =$ blank, then

$$h(p(c^*, c^i)) = \begin{cases} (\min \{h(p(c^*, c^i)) \mid c^i \in N(c^i)\}) + 1 \text{ over all} \\ \quad c^i \text{ for which } h(p(c^*, c^i) \text{ has been defined;} \\ \text{undefined otherwise.} \end{cases}$$

From these definitions it again follows that both h and g are monotone functions. $F = (h, g)$ is therefore a monotone vector. Therefore, we have:

Corollary 3: Algorithm A solves the joint minimization problem with respect to first distance then edge effect.

Example 4: Let us consider again the original configuration shown in Fig. 5. We wish to find a path AA which solves the path problem with respect to the vector $F = (h, g)$ for this configuration.

Following rule R5, we begin with cell 41 in list L. By rule R1, cells 20, 40 and 42 therefore belong to list L_1. The possible cell masses for cells 20, 40 and 42 are

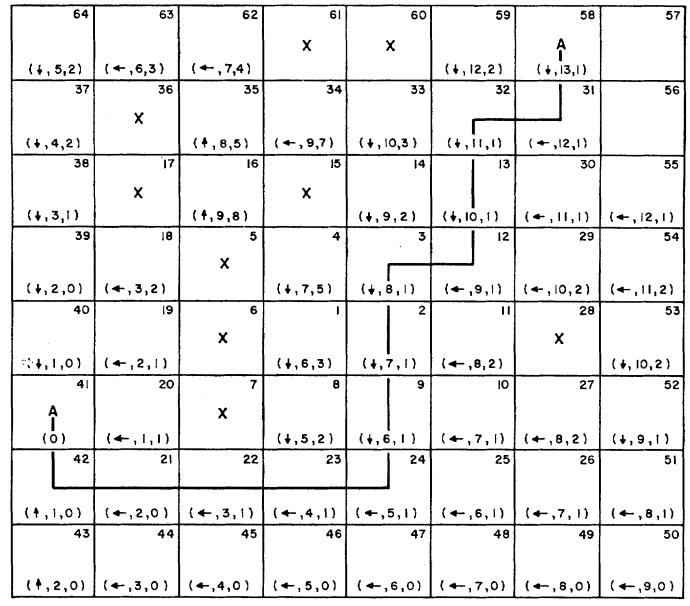

Fig. 6—A joint minimization example.

respectively (1, 1), (1, 0) and (1, 0). Therefore, $m(40) = (1, 0)$ and $m(42) = (1, 0)$. The chain coordinates for cells 40 and 42 are respectively ↓ and ↑.

From rule R4, the list L now contains cells 41, 40 and 42. Therefore, by R1, list L_1 has now in it cells $\{20, 19, 39, 43, 21\}$ with possible cell masses respectively: (1, 1), (2, 1), (2, 0), (2, 0), (2, 0). Thus $m(20) = (1, 1)$ and cell 20 has chain coordinate ←.

Continuing in this manner, and using again the selection rule R3 that the left direction is to be preferred over all other directions, we arrive at the path shown in Fig. 6. We see that in this case, since we are interested in the shortest path, the path AA makes contact once with an edge. We had seen before that there are paths which make no contact with any edge.

D. Generalizations

We should, perhaps, take a moment at this time to reflect on the following two questions. In Section II, we have set up an abstract model, our ℭ-space, in such a way that algorithm A can be used to solve path problems formulated within this model. The first question we will ask is whether algorithm A can be applied to still more general situations—that is, whether our abstract model of ℭ-space may be further generalized. In a similar way, we may wish to relax the monotone condition on vectors F. The second question is, therefore, whether our basic theorem may be generalized to include also perhaps a subclass of non-monotone vectors.

The answer to the first question can be given in the affirmative, and may appear a bit surprising. Specifically, we may redefine the 1-neighborhood function N such that N needs to satisfy neither rule $N1$ nor rule $N2$ of Section II. The only condition that N must satisfy is a *finiteness* condition:

NO. Every 1-neighborhood is finite.

Let us call a space (C, S, N^*, Γ, M) a ℭ-*space if the 1-neighborhood funciton N^* satisfies the finiteness condition NO rather than conditions $N1$ and $N2$ of Section II. Accordingly, we must also make minor changes to algorithm A. For instance, we must redefine the coordinate functions d_i, and modify our process of assigning chain coordinates. Let us agree to call the modified algorithm A^*. Then our basic theorem would read:

Let a ℭ*-space (C, S, N^*, Γ, M) be given. Let P be a path problem with respect to a vector F. If F is monotone, then algorithm A^* yields a path $\bar{p}(c^*, c^{**})$ satisfying P.

In regard to the second question, our knowledge is very meager. It is quite possible that an essentially different algorithm is needed to deal with non-monotone vectors.

Coming back to our basic theorem, we ought to make it clear that even when a vector F is monotone, it may be so pathological that the process of applying algorithm A could become extremely tedious. To be specific, let $p(c^1, c^n)$ be an admissible path consisting of the chain of cells:

$$p(c^1, c^n) = \{c^1, c^2, \cdots, c^n\}.$$

A monotone function f is said to be *1-hereditary* if $f(p(c^1, c^n))$ depends only on $f(p(c^1, c^{n-1}))$ and on cells c^{n-1} and c^n. A monotone function f is said to be *2-hereditary* if $f(p(c^1, c^n))$ depends only on $f(p(c^1, c^{n-2}))$ and $f(p(c^1, c^{n-1}))$ and cells c^{n-2}, c^{n-1} and c^n. In a similar manner, we may define p-hereditary functions for $p \geqq 1$.

All of the examples of monotone vectors given in this section happen to be 1-hereditary. In such cases the process of applying algorithm A is much simplified. We may also apply algorithm A to solve the *minimal corner problem*; that is, to find a path with the least number of corners. This problem presents an interesting twist, since the corner function is monotone but 2-hereditary. In the same way, one may construct p-

hereditary functions for arbitrary p. Indeed, one may construct monotone functions which are not finitely hereditary.

V. Minimal-Distance Solutions—A Maze and Other Illustrations

As the reader can see, the statement of algorithm A lends itself quite directly to computer programming. Such a step for the minimal distance problem has been carried out. A natural cell configuration is determined by the printer associated with the computer. The printer can print 120 characters in one line, and usually prints 60 lines to a page. The cell configuration is therefore a rectangular array of 120 cells by 60 cells. The sets of symbols are, in this case, the set of characters on the printer.

In Fig. 7 is shown an input into the computer. The boundary cells and the obstacle cells are all marked $\times$. In this illustration, we wish to find a minimal-distance path between cell A and cell B.

Fig. 8 shows the result of applying algorithm A1 (the search algorithm) to the cell configuration given in Fig. 7. Since it is not possible to print more than one character in each cell, we have chosen to print out the least significant octal digit of the cell masses. The reader can see that the immediate neighbors of cell A has cell mass 1. The neighbors of these have cell mass 2, etc. This search-expansion process continues until cell B is reached.

Fig. 9 shows the result of applying algorithm A2 (the trace algorithm) to the cell configuration. The selection rule used here is to order the admissible neighbors according to the following list of preference: right, up, left, down. The path shown is the machine's solution to the original path problem.

Fig. 10 depicts a three-stage adder circuit; each box designates one of the circuit stages. We wish to apply algorithm A to establish all appropriate connections.

Fig. 11 shows the result of applying algorithm A. One may note that in these paths, there are many more corners than necessary. The appearance of these extra corners is due to our simple selection rule. To a large extent, the appearance of paths can be controlled by incorporating appropriate selection rules into algorithm A.

Shortly after the program was written, we realized that this program, without change, can be used to also solve maze problems. In this sense then, and with proper modifications, the program may serve in particular as a 704 version of Shannon's maze-solving machine.[5] An illustration is the Hampton court maze shown in Fig. 12. Fig. 13 shows the result of applying the search algorithm to the maze configuration. The machine's solution is given by Fig. 14.

VI. The Search Algorithm and Huygens' Principle

In the course of programming algorithm A for the minimal-distance problem, it occurred to us that the search algorithm is, in a remote sense, a computer model of waves expanding from a source under a form of straight-line geometry. Those cells having the same cell mass may be thought of as the locations of the wavefront at the mth unit of time. Fig. 8, for example, may be taken to represent the wavefront originating from source A as it expanded.

Following this line of thought, we proceeded to run a few simple experiments suggested by optics. In Fig. 15, the obstacle consisted of a vertical barrier with a slit in the middle. The source is at the left-hand end of the diagram. The blank spaces may serve as an indication of the propagation of the wavefront. The reader may see that the wave pattern to the right of the obstacle is the same as that which would have been produced by a source located at the slit.

In the same way, Fig. 16 shows the effect of having a 2-slit obstacle and Figs. 17 and 18 are patterns created by having a number of multiple-slit obstacles.

There are several differences between the model shown here and the usual diffraction patterns in optics. The geometry assumed in this model is, in the first place, not Euclidean. Since distances in this geometry are measured roughly as distances are understood by taxi drivers on the island of Manhattan, this geometry is sometimes called *Manhattan geometry*.

In addition to the difference in geometry, and the fact that we are dealing with a discrete space, we have also not taken into account the phenomenon of interference. The computer instead has a built-in first-come-first-served rule. That is, where there are several sources present, the amplitude of the wave at any point is determined by the source closest to it.

Although there is only a remote resemblance between this model and optics, these experiments seem to suggest the possibility of microsimulation of physical phenomena on a computer and the possibility of looking for effects in this way if the laws of nature were modified. In our case, it would be possible to include also interference. The model would then be a reasonably faithful representation of elementary wave phenomena under discrete Manhattan geometry.

VII. Acknowledgment

In the course of this work, the writer has had the benefit of stimulating conversations with a number of people. Among these are E. F. Moore, T. H. Crowley, D. H. Evans, S. H. Washburn, R. W. Hamming and C. A. Lovell. The "diffraction" experiments were an outgrowth of a conversation with D. H. Evans. The writer also wishes to acknowledge the untiring assistance for Sections V and VI from S. O. Four of IBM.

[5] C. E. Shannon, "Presentation of the maze-solving machine," *Trans of the 8th Cybernetics Conf.*, Josiah Macy Jr. Foundation, New York, N. Y., pp. 173–180; 1952.

(*See Figs. 7–18, on following pp. 354–365.*)

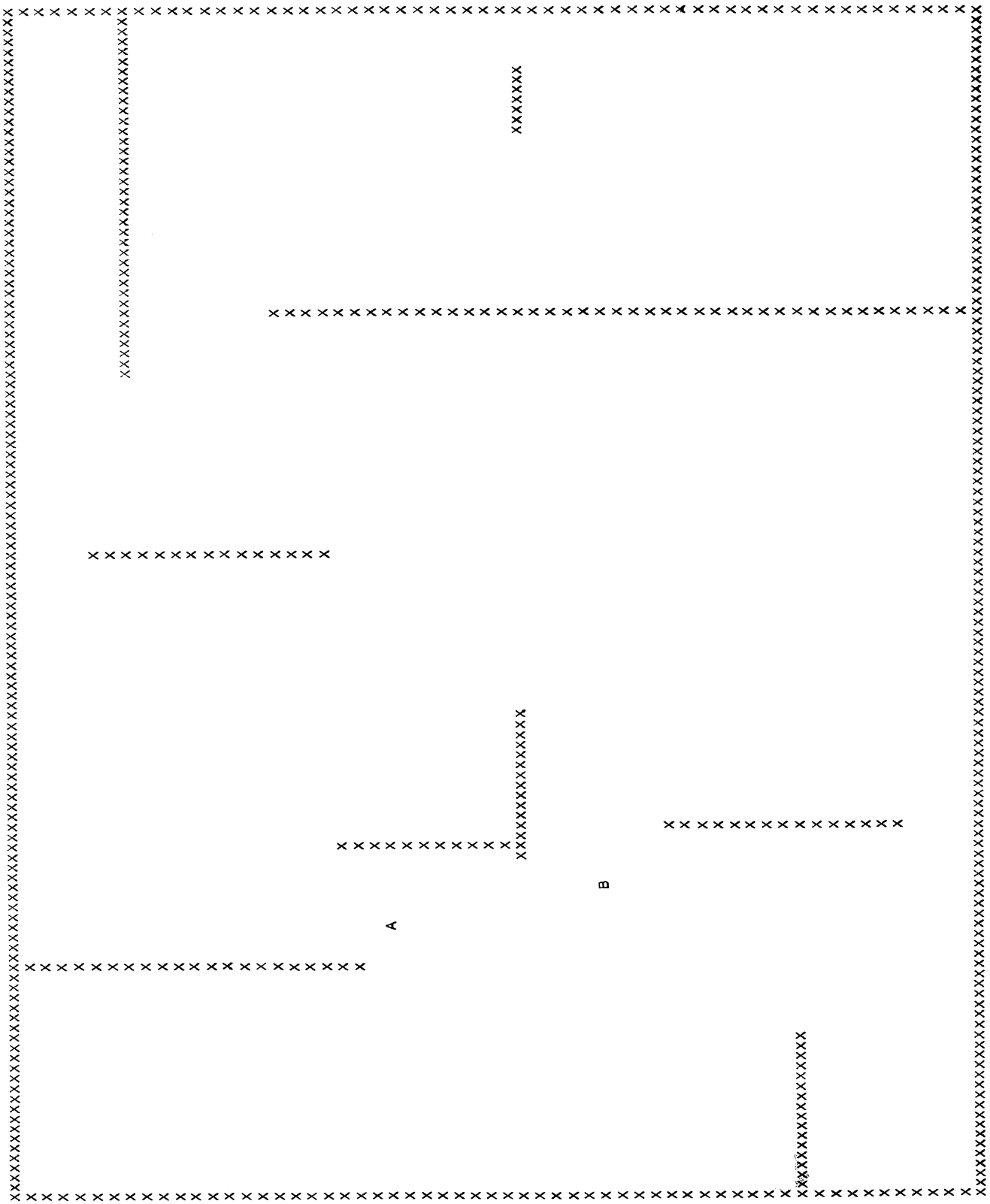

Fig. 7—Input into computer for minimal-distance solutions.

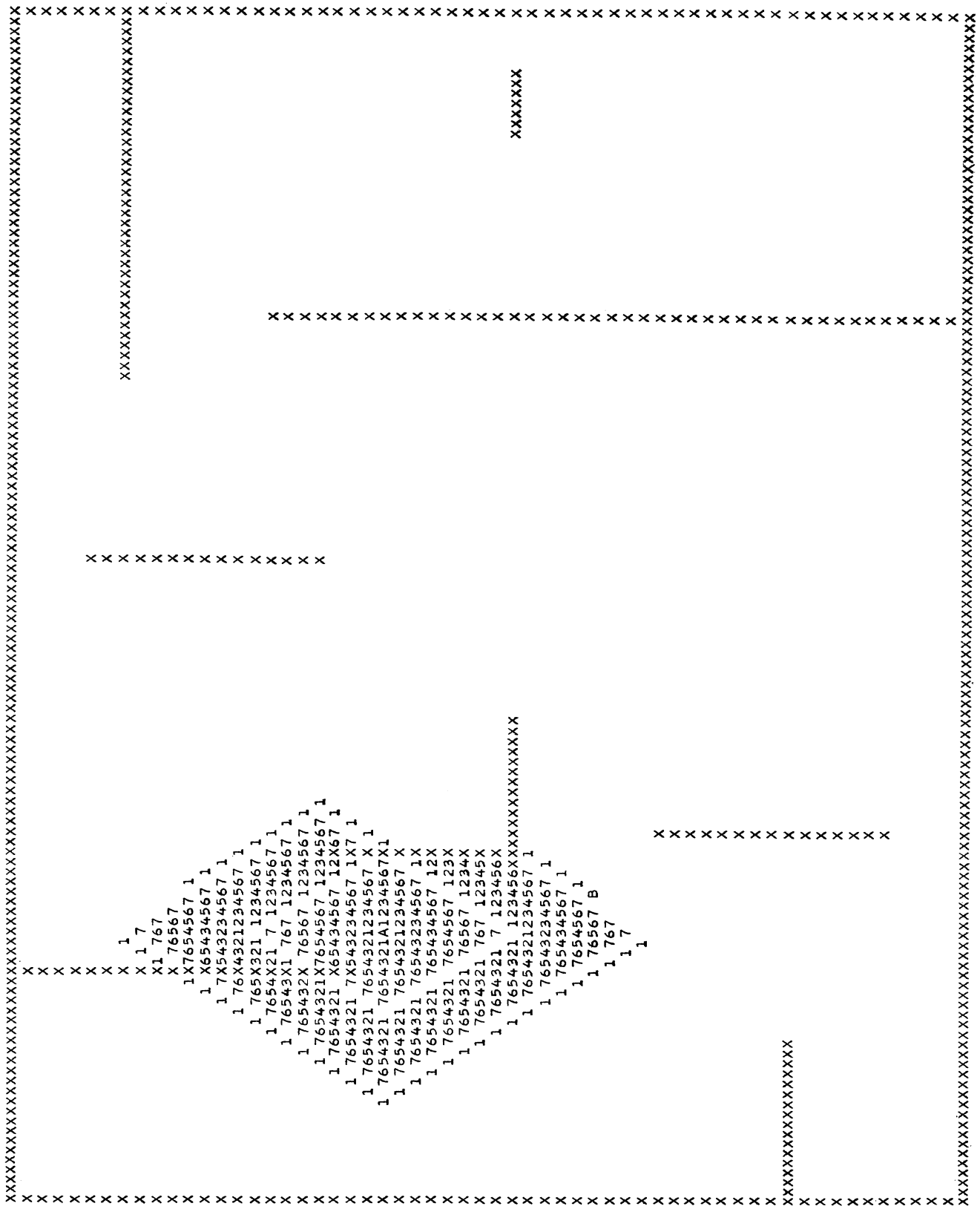

Fig. 8—Result of applying search algorithm to Fig. 7.

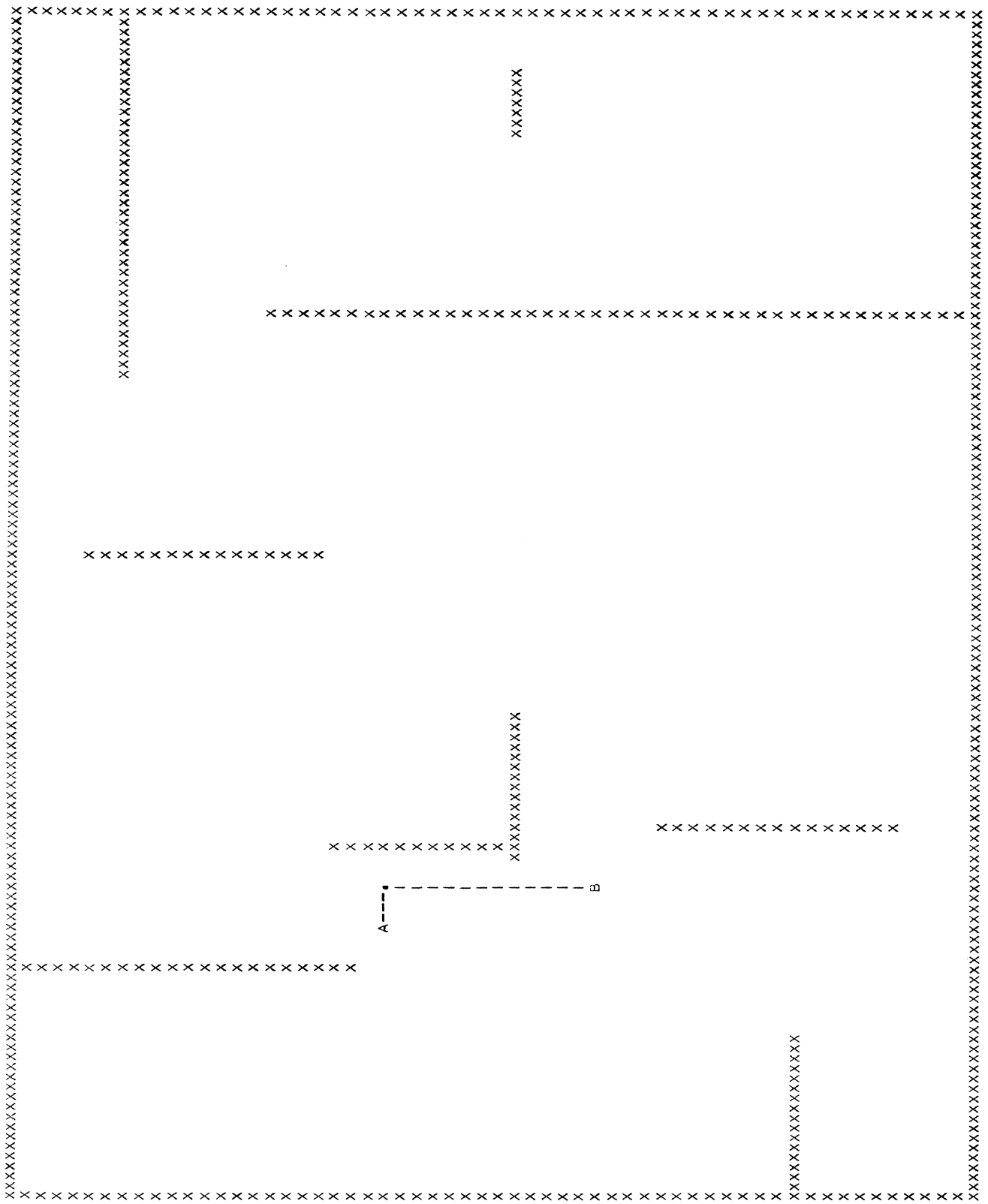

Fig. 9—One solution to the minimal-distance problem.

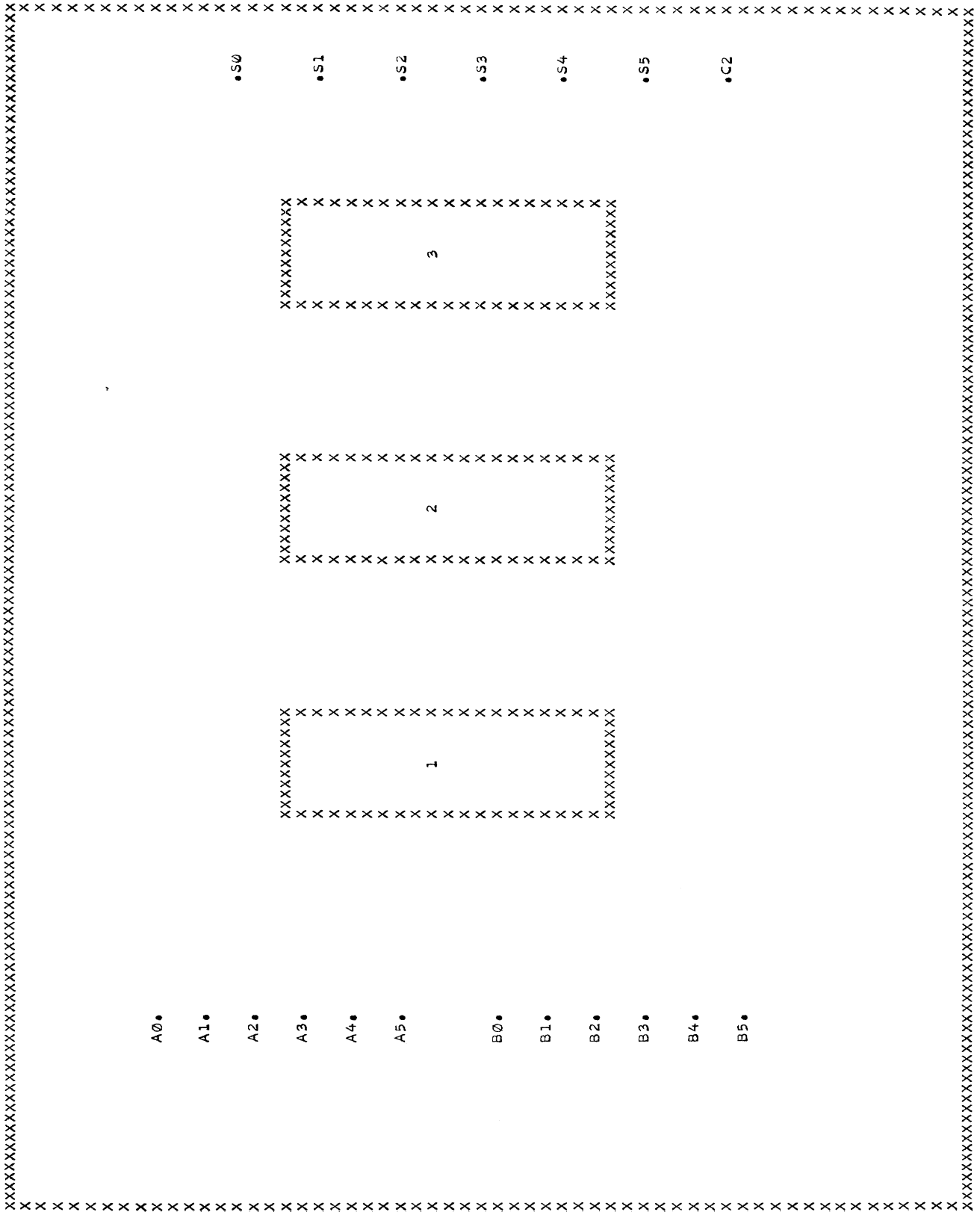

Fig. 10—A circuit block diagram as input into computer.

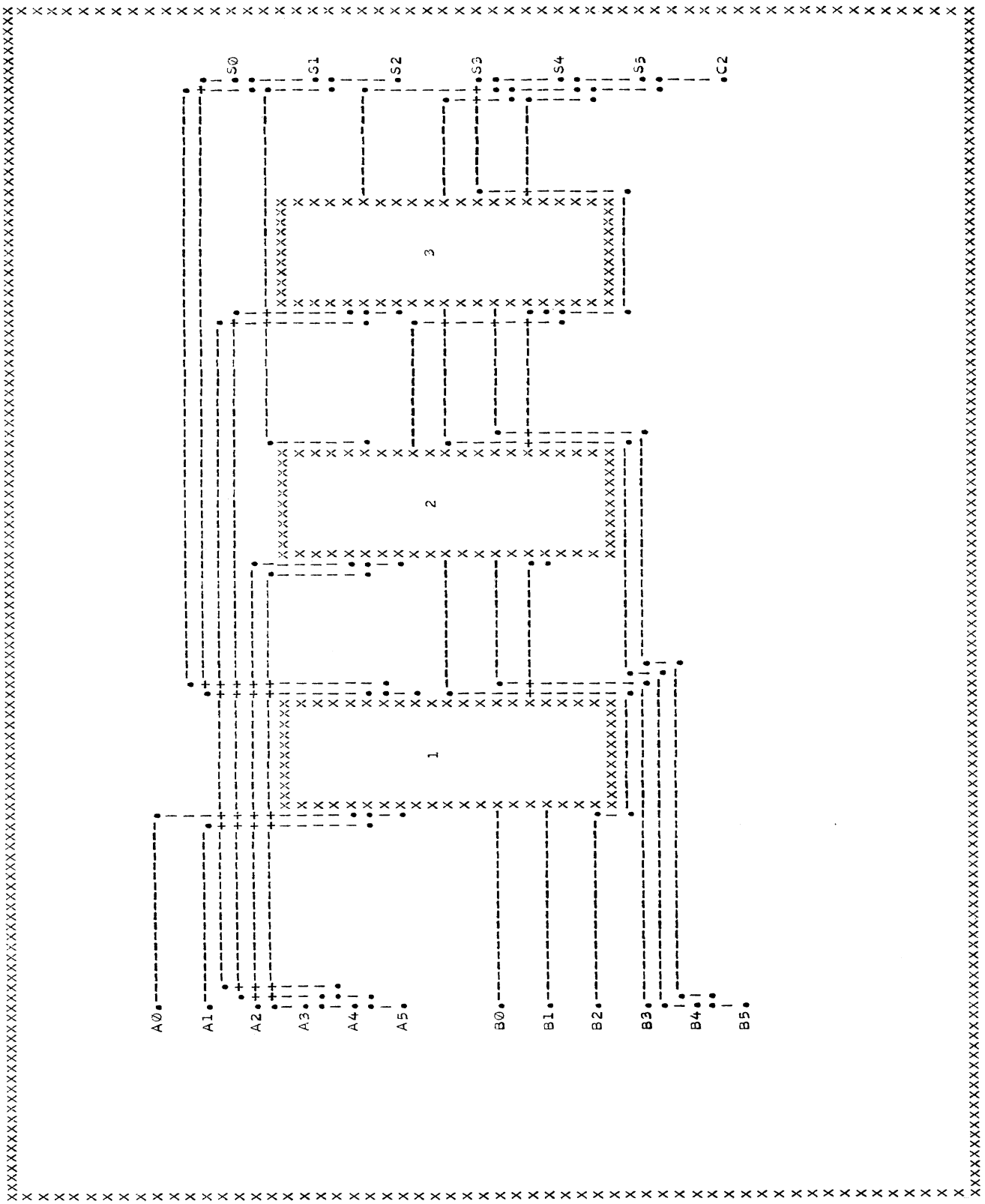

Fig. 11—Result of applying algorithm A to Fig. 10.

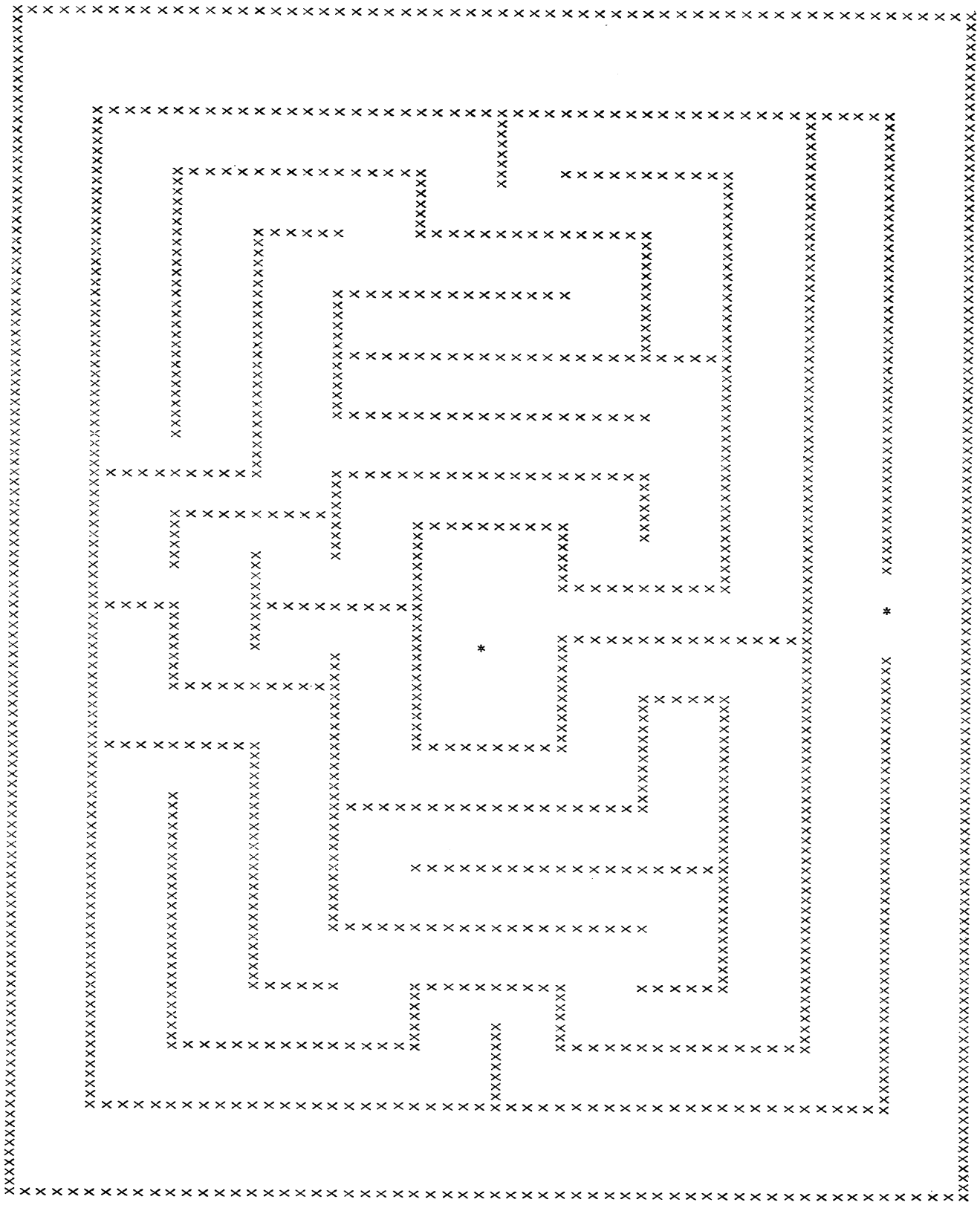

Fig. 12—A maze input into the computer.

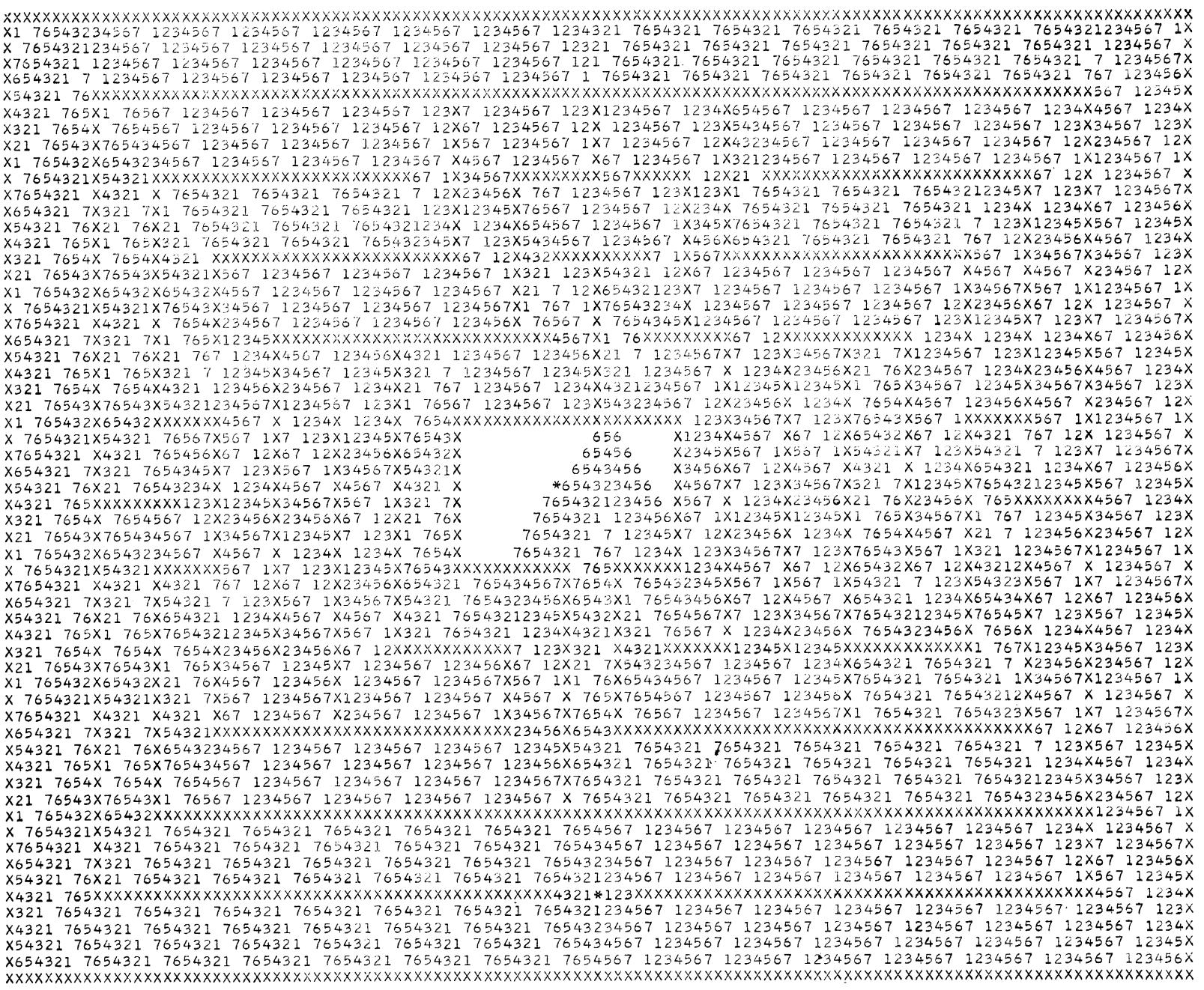

Fig. 13—Result of applying search algorithm to Fig. 12.

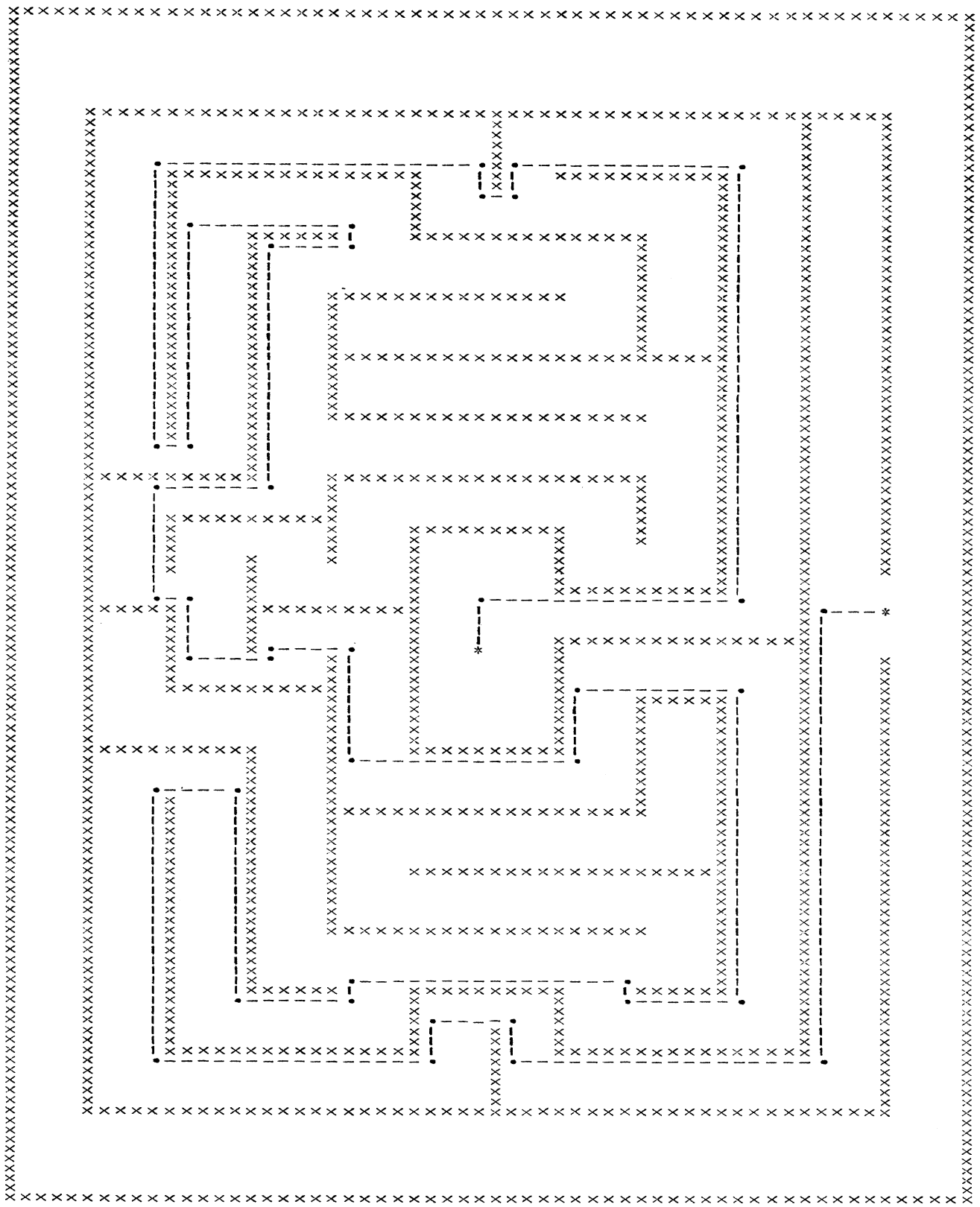

FIG. 14—Machine's solution to the maze problem.

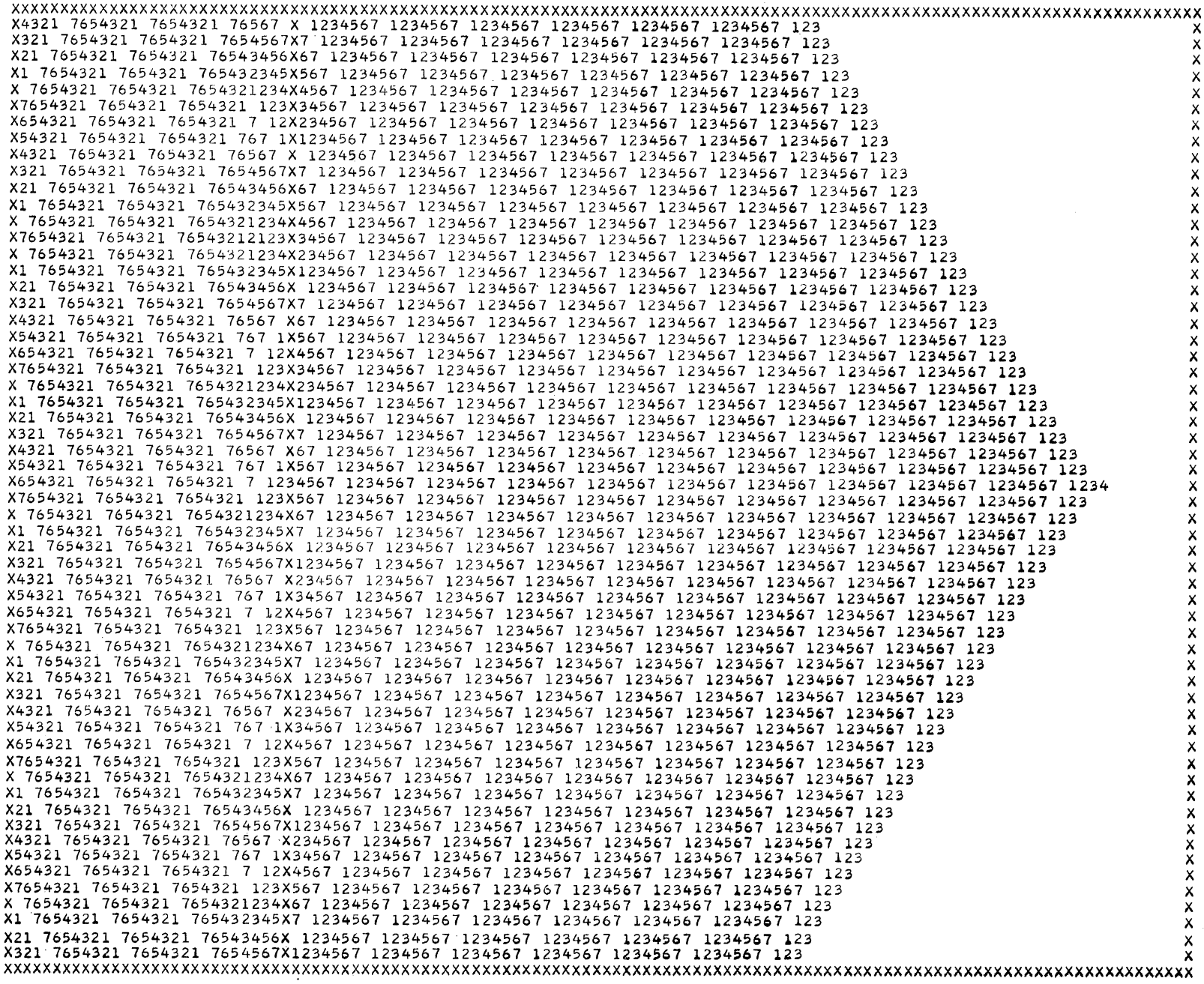

Fig. 15—A single-slit experiment.

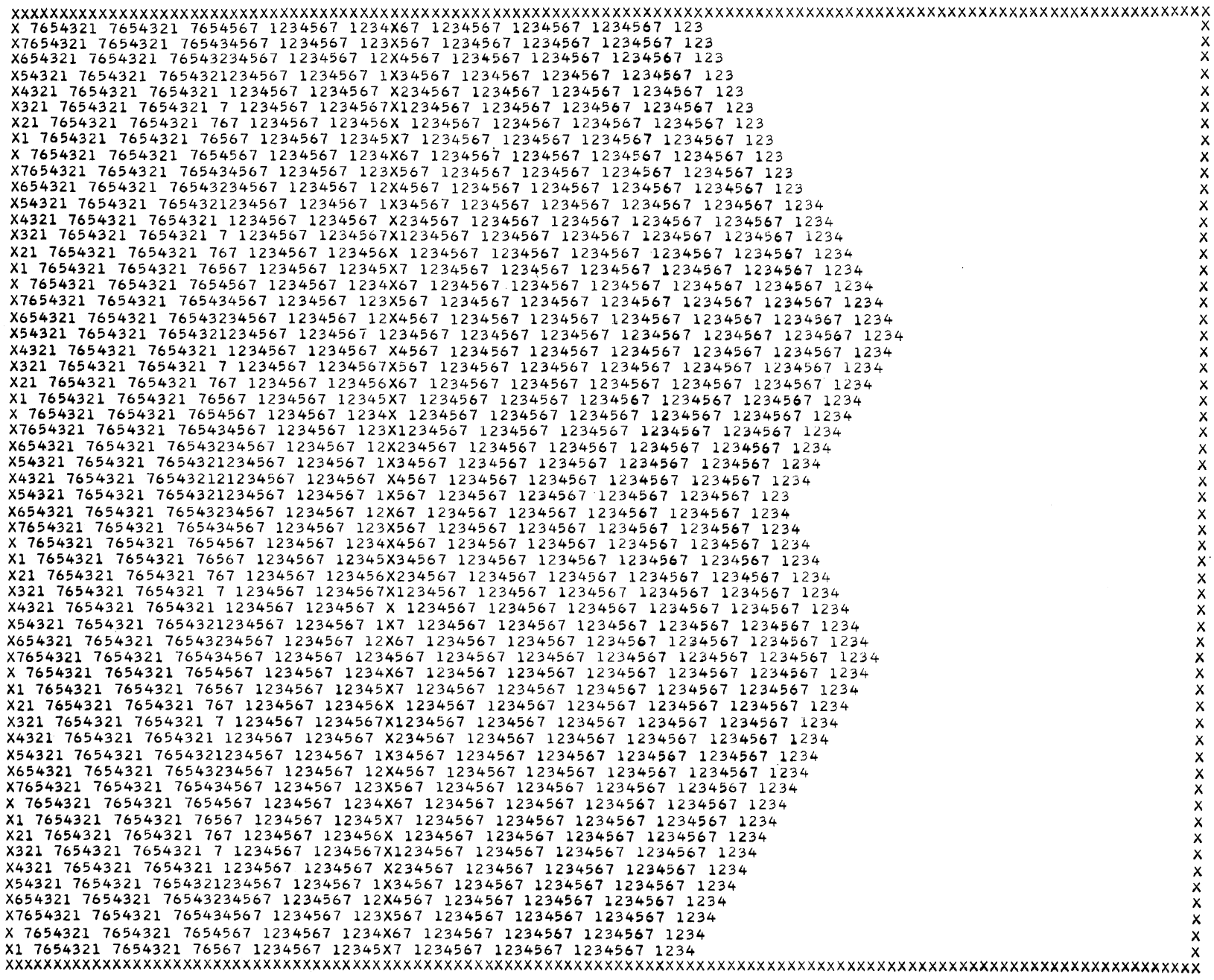

Fig. 16—A double-slit experiment.

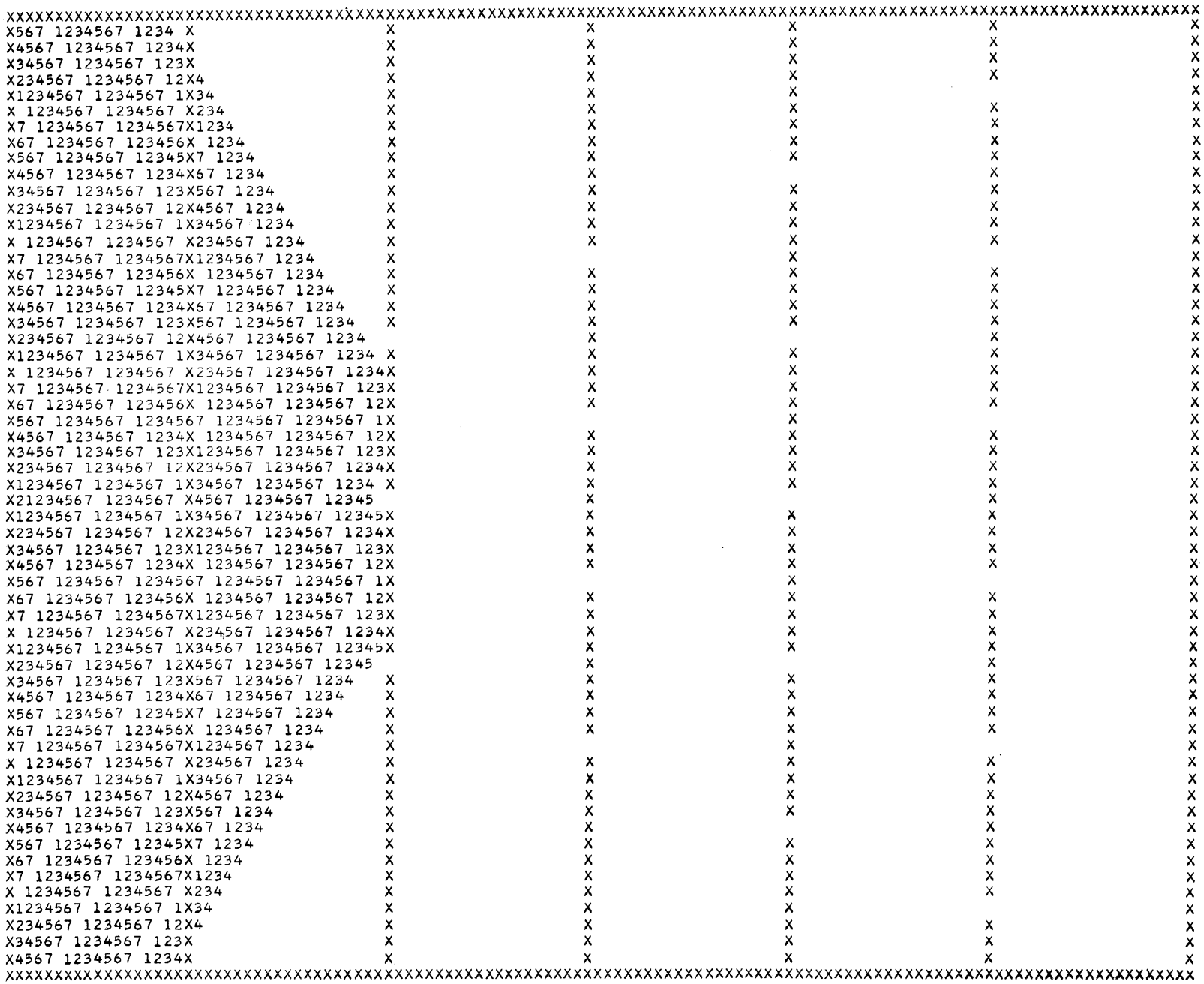

Fig. 17—Beginning of a multiple-slit experiment.

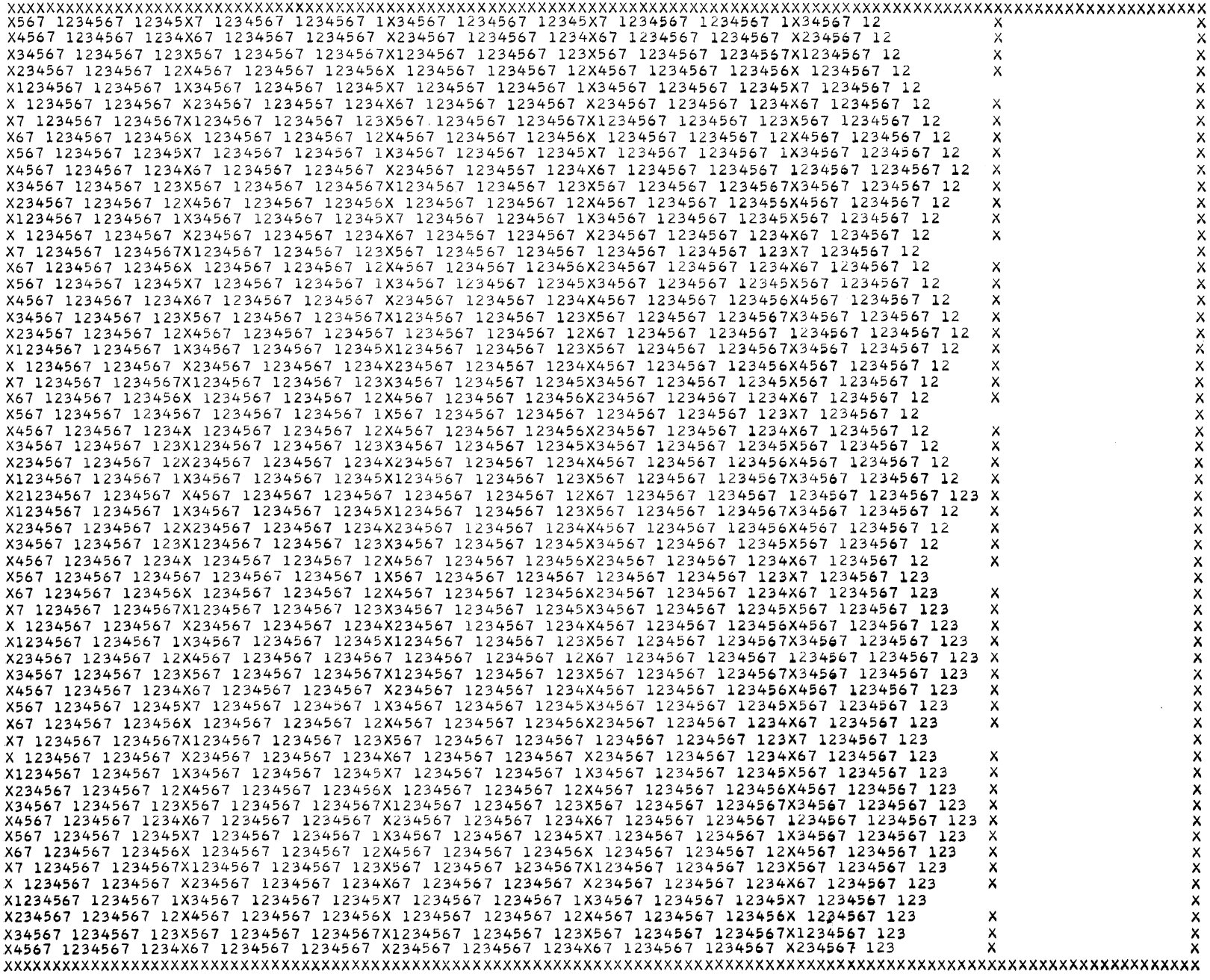

Fig. 18—Near the end of a multiple-slit experiment.