PROCEEDINGS OF AN INTERNATIONAL SYMPOSIUM ON THE THEORY OF SWITCHING

2-5 April 1957

PART II

CAMBRIDGE, MASSACHUSETTS
HARVARD UNIVERSITY PRESS
1959

SEVENTH SESSION

Friday, April 5, 1957
9:00 A.M. to 12:00 M.

SWITCHING SYSTEMS

Presiding

George W. Brown

Professor of Engineering
Director of Western Data Processing Center
University of California at Los Angeles
Senior Staff Engineer
International Telemeter Corporation

The Shortest Path Through a Maze

EDWARD F. MOORE

Member of the Technical Staff
Bell Telephone Laboratories, Incorporated

Suppose that it is desired to find the shortest path through a maze. For a very simple maze this can be done just by observing it and trying a few paths that look reasonable. But this trial-and-error procedure does not apply to a complicated case, as can be seen by looking at Fig. 1. Persons not familiar with the methods given in this paper usually take

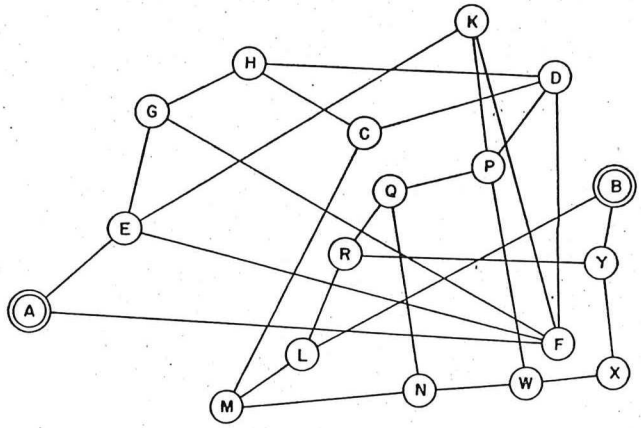

Fig. 1. Can you find the path of length six from A to B?

five or ten minutes to solve the problem of Fig. 1. The solution to this problem is given in Fig. 2, which is printed near the end of this paper.

The methods given in this paper require no foresight or ingenuity, and hence deserve to be called algorithms. They would be especially suited for use in a machine, either a special-purpose or a general-purpose digital computer.

One place in which a problem involving paths through mazes arises is in the routing of toll telephone traffic. If it is desired to place a call from city A to city B of Fig. 1, and if a trunk circuit going directly between these cities (which would be represented in Fig. 1 by a straight line directly between the two cities) is available, it would certainly be used. However, between any pair of cities no direct trunk circuit is available at certain times, because all the trunks provided are "busy"; that is, they are all in use by other customers. But if such a case occurs, and there is a path going by way of one other city (I shall call this a path of length two) which is known to be available, such a path would probably be used. The traffic is so congested in Fig. 1 that the shortest path available is of length six, going by way of five cities (in addition to A and B) before it reaches its destination. In such a situation, the best path to use would not necessarily involve the least number of steps, but

might be one involving less cost, less distance, or giving a better quality of telephone transmission. Before showing how to include such features, we shall consider the simple case in which it is desired to minimize the number of cities visited.

In addition to communication problems such as the one above, there are situations involving mazes which arise in transportation. For instance, suppose that we have an air-traffic or a railroad-traffic control system. Let us assume that a certain number of airplanes (or trains) have already been routed, and it is desired to route one more, without changing any of the previous routings, so as to arrive at the destination as early as possible without any near collisions.

One way of approaching this problem is to divide the space available into blocks. A block in a railroad system might be a few miles of track. A block of air space might be a region several miles long and 1000 feet thick. We shall also divide time into blocks such that it takes one block of time to traverse one block of space. The restriction might be that no two vehicles would be allowed within two blocks of each other. At any rate, the vertices of the maze (corresponding to the cities of the example above) would be blocks of space-time. In other words, if space number 13 were occupied at time number 7, the point having the coordinates (13,7) should be deleted from the maze, so that no more vehicles will be routed through this block. To keep a distance of two blocks between vehicles, we should also delete the blocks (12,7), (14,7), (13,6), and (13,8) from the maze. The edges of the maze will be between any two blocks of space-time where it is possible to travel without violating safety conditions. The shortest path through the maze would give the preferred routing for the next vehicle.

The foregoing two examples have shown that there are problems in the efficient use of congested communication or transportation systems which might be solved by the use of mazes. The present paper does not attempt to list all such possible applications, but a few more will be mentioned later.

Now let us actually carry out the steps of an algorithm for solving the problem given in Fig. 1. We shall call this algorithm A to distinguish it from some other algorithms to be discussed later. On the *first* step write the number 0 on city A. (It is not necessary to actually write on Fig. 1, since the results of this are given in Fig. 2.) On the *second* step write the number 1 on the cities E and F; on the *third* step write the number 2 on cities D, G, and K. On each step the next larger number will be written on all cities which have not yet been written on, but which are adjacent to cities which have. It can easily be proved that the number being written will always be the length of the shortest path from city A to the city on which the number is written. On the *fourth* step write the number 3 on the cities C, H, and P; on the *fifth* step write the number 4 on the cities M, Q, and W; on the *sixth* step write the number 5 on the cities L, N, R, and X; on the *seventh* step write the number 6 on the cities B and Y. We have now reached the destination of the path, and (even if not all the cities are yet reached) we return to the starting point step by step. On the *eighth* step draw an arrow from B to L; on the *ninth* step draw an arrow from L to M.

At each step we now draw an arrow (in a telephone mechanization of this we might at this point actually be setting up the connection over which talking will take place) from the city last reached to any city numbered one lower, that is, any city one step nearer to the beginning point. On the *tenth* step draw an arrow from M to C; on the *eleventh* step draw an arrow from C to D; on the *twelfth* step draw an arrow from D to F; on the *thirteenth* step draw an arrow from F to A; on the *fourteenth* step erase all of the numbers which have been written on the cities. The shortest path has now been found to be $AFDCMLB$.

In general, if the length of the shortest path is n, algorithm A will require $2n + 2$ steps. If this is carried out in parallel with separate circuits corresponding to each city, it would require only $2n + 2$ pulse times in a synchronous machine. If it were done as a program on a digital computer, the steps given as single steps above would be done serially, with a few operations of the computer for each city of the maze; but, in the case of complicated mazes, the algorithm would still be quite fast compared with trial-and-error methods.

In algorithm A it would be necessary to have a memory capable of remembering an arbitrary integer (up to the length of the longest path which could occur). If it were desired to do the same problem in a way which requires less memory, another method, which will be called algorithm B, could be used. Algorithm B would be somewhat more confusing to the circuit designers and maintenance men, but it would find the same path in the same number of steps as algorithm A, using less memory to do so. The integers modulo 3 are used. At each step only the numbers 0, 1, or 2 could be written, and this number would always be the remainder obtained from dividing the distance from city A by 3. This remainder would not be obtained by actually dividing, but instead by adding 1 mod 3 to the number on the adjacent cities. Instead of using the integers mod 3, the integers mod n could be used, for any $n > 3$. The use of the integers mod n depends on the fact that, if Fig. 2 were calculated for any set of cities, each city must have an integral label which is equal to, one less than, or one more than the label of any adjacent city. Integers mod 3 are sufficient to distinguish among these three cases, but the integers mod 2 are not.

In order to carry out algorithm B we need only two bits of memory for each city of the maze. There are only four possible states in which the sequential circuit can be at each city: the blank state (if no number has yet been written at the city) and the states 0, 1, and 2. Similarly, if algorithm B is carried out by programming operations on the memory of a computer, only two bits of memory are needed for each city. The operations done while performing each step of algorithm B would be done in exactly the same order as those done in algorithm A, the only difference being that the numbers recorded are integers modulo 3 instead of ordinary integers. Of course, in an actual mechanization of algorithm B, a few more states would be required to handle synchronization, writing in the arrows, and so on, but the circuit required at each city would have a fixed complexity, regardless of how many cities were added to other parts of the network or how long the longest possible path became.

There is still another algorithm for solving the same problem, which will be called

algorithm C. Algorithm C requires only one bit of memory per city. To carry out this algorithm, we first write 0's in all of the cities. Next, we write a 1 in city A. On each later step we write 1's in all those cities adjacent to cities in which 1's were written the previous time. After n steps city L is reached, and then an arrow is written from B to L. As the next step, all 1's are changed to 0's. Next the writing of 1's is repeated. After $n - 1$ steps the city M is reached, and then an arrow is written from L to M. The procedure is continued until the arrows finally reach city A. The total number of steps required is

$$(n + 1) + n + (n - 1) + \cdots + 1 = \tfrac{1}{2}(n + 1)(n + 2),$$

where n is the length of the shortest path from A to B. Algorithm C thus uses less memory than either of the previous algorithms but it requires considerably more time to solve the problem. The choice of which algorithm to use in a machine would depend on whether speed or the cost of memory were the more critical consideration in the contemplated application.

All of the algorithms discussed solved exactly the same problem, but the actual maze problems of practical importance in communication and transportation are not this simple. Usually what is wanted is not the path going through the least number of intermediate cities, but the one having the least elapsed time or requiring the least fuel (in the case of transportation) or the best quality of transmission or least expensive circuits (in the case of communication). One example of a problem of this more complicated nature would be to have a number called the *cost* associated with each edge of the maze. With each path through the maze we shall associate the sum of all of the costs of all of the edges traversed by this path. We shall assume that it is desired to find the path which minimizes this total cost. Figure 3 illustrates such a problem, using a maze topologically equivalent to Fig. 1, but having the cost written next to each edge. The procedure we will use to solve this problem will be called algorithm D.

On the *first* step of algorithm D, we shall write the number 0 on city A; on the *second* step we shall write the number 3 on city E and the number 11 on city F; on the *third* step we shall write the number 11 on city K, the number 9 on city G, the number 17 on city D, and the number 10 on city F (erasing the number 11 which had previously been there). After the $(n + 1)$th step the number written on each city will be the total cost of reaching that city by the shortest path of length n or less. The number to write on a given city J at each step is calculated as follows: take each of the numbers written during the last step on neighboring cities, and add to it the cost of traversing the one remaining edge to reach J, and then write on J the smallest number in the set whose members consist of the number (if any) already written on J and of each of the totals just calculated.

To illustrate this rule, on the *fourth* step we write the number 19 on city H, the number 40 on city C, the number 18 on city P, and the number 16 on city D (erasing the number 17 which had previously been written there). Skipping the details of the next few steps, after the *ninth* step we arrive at Fig. 4. We then terminate the procedure, because taking further

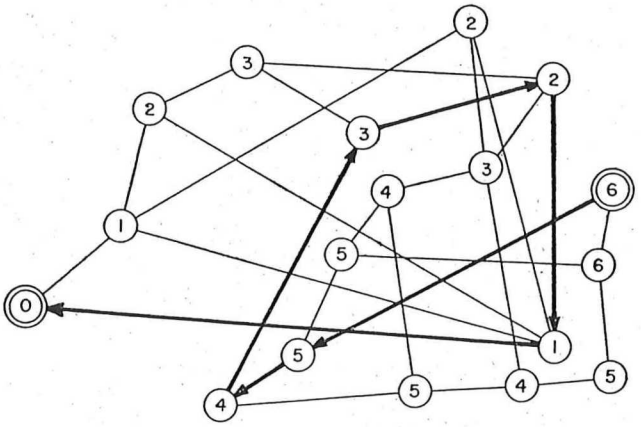

FIG. 2. The result of applying algorithm A to Fig. 1.

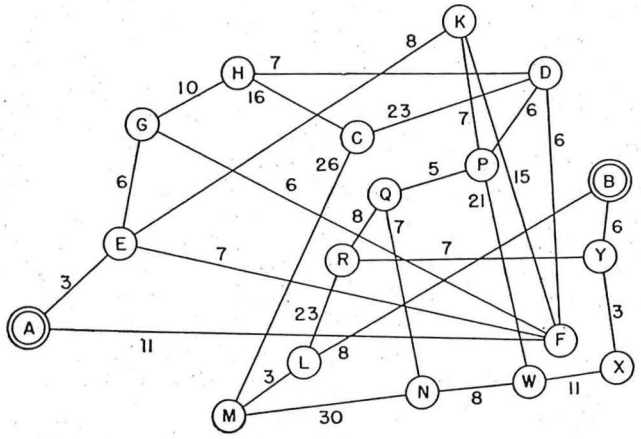

FIG. 3. Which path from A to B involves the least total cost?

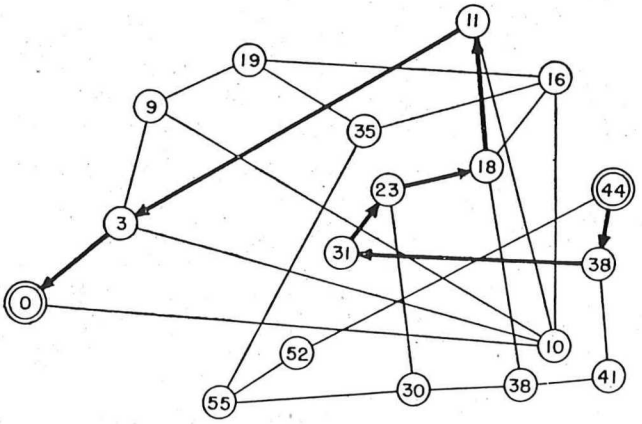

FIG. 4. The result of applying algorithm D to Fig. 3.

steps does not change any of the numbers written on the cities. Then we draw arrows from B to Y, Y to R, and so on, marking the path $BYRQPKEA$, which has a total cost of 44.

In the process of marking this path, at each step we draw from the city last reached an arrow which has the property that the two numbers marked at the two ends of the arrow differ by the cost of traversing that step. The arrowhead always points toward the smaller of the two numbers. Since the numbers written on each city at this time are just the cost of the shortest path from that city to city A, following the arrows drawn in this way just insures that we will go by the least expensive path.

But what happens if there are two solutions to the problem? Then, as we follow these arrows we shall come to some city which has two arrows pointing out of it. Regardless of which one we follow, we shall be on a path of minimum cost. The fact that we reach such a point will serve as an indication that there is more than one path having the minimum cost. In fact, each of the algorithms A, B, C, and D has the property that a correct solution will be obtained even when the problem has no unique solution. Furthermore, if the solution is not unique, all the information is there with which to calculate all minimal paths.

In the mechanization of these algorithms, it should be noticed that algorithms B and C are not suitable for use in asynchronous circuits. All of the operations of step n must be completed throughout the maze before the operations of step $n + 1$ begin. That is, each step should be done during the nth clock pulse. If this were not done, and some of the circuits operated faster than others, we could have situations where the numbers were written in the wrong places or in the wrong order, and could find a path which was not a solution to the problem.

However, algorithm D is ideally suited for either synchronous or asynchronous circuit operation. Since the numbers that are written on the cities can be changed if other paths turn out to be less expensive, the machine could be designed to change these numbers as soon as the information about a lower value arrives from a neighboring city. Then as soon as the numbers stop changing a solution will have been reached. At any intermediate step in the process the numbers will not necessarily represent the least expensive path of length n or less, but will represent the least expensive path which has had time for all of the circuits to operate. And since algorithm A can be treated as a special case of algorithm D (in which all of the costs of individual steps are 1) it can also be operated either asynchronously or synchronously.

These four algorithms do not exhaust all of the variations on this method of solving mazes. As a further example, which will only be described roughly (and hence does not deserve the name algorithm), suppose that you are in a certain city at a certain time and wish to arrive at your destination by train as soon as possible. Then by consulting your timetable you can write down the earliest time at which you can arrive at each city which it is possible for you to reach without changing trains. Then, on the next step, you can write down the earliest time at which you can reach each city which can be reached by changing trains only once. It will be necessary to make some assumptions about how long

it takes to change trains in each city. You can continue this process, always keeping track of the earliest time that you have found for reaching each city, until you reach a point where none of these times can be improved. Then you pick out the route you could follow. This would be possible even though the timetable contained many arbitrary restrictions (trains that stop at Plainfield only for passengers to points beyond Philadelphia, trains that do not run on Sundays or holidays, and all of the other footnotes on the timetable). Thus, it is not necessary to have a simple algebraic operation such as summation which describes the restrictions on the paths. Similar modifications might be made to give better solutions to the traffic control problem than the method mentioned at the beginning of this paper. For instance, it was assumed that all vehicles travel at the same speed, and that time and space can be quantized without loss of efficiency. By making up something corresponding to a timetable in complexity it should be possible to approximate more nearly the complexities of the real physical situation.

There have been at least two previous attempts to solve problems of this maze variety by the use of explicit procedures on machines. The case of automatic switching of toll telephone calls is more fully described by the papers listed in reference 1. The switching points (that is, the vertices of the maze) were arranged in a hierarchy, or chain-of-command structure, with a national center, regional centers, sectional centers, primary outlets, and toll centers. Each of these levels is given a different strategy for dealing with calls, and, in case the direct routes are all busy, a call is sent up and down this chain of command. The method of routing calls is such that one step is made before the availability of all further parts of the route is checked. Thus, in case of a temporary overload on the system it is possible for it to find a route which is not quite the most economical one possible. Also in the case of an extreme disruption of the system, such as flood or earthquake damage could provide, it is possible for it to fail to find a route even though an indirect one is available. On the other hand, algorithm *D* can be rigorously proved to give always the most economical route which is available.

However, the present method of toll telephone switching has the advantage that it can be used to set up different calls simultaneously without interference. If several calls originated simultaneously at one city, the proposed algorithms could be modified to handle them simultaneously. But since actual communication systems have traffic originating independently throughout the system, the proposed algorithms could only be used with control circuits having higher speeds of operation than the present circuits, if they are to be capable of handling the same volume of traffic.

In the case of finding the fastest route for a truck through congested city traffic,[2] a program was written for a digital computer such that "by trial and error a sequence of paths which minimizes the objective function as a function of time has been determined." Algorithm *D* has the advantage over the method used in the program in that it proceeds directly to the solution without trial and error or iterations, and it is so much simpler that it would be very easy to program on a computer.

All four algorithms given in this paper are applicable to mazes having paths which can be traversed in only one direction, such as one-way streets. The only modification required is that, in making the steps which depend on the adjacency of two cities, only adjacencies consistent with the directions of the arrows of the directed graph are to be used.

The origin of the present methods provides an interesting illustration of the value of basic research on puzzles and games. Although such research is often frowned upon as being frivolous, it seems plausible that these algorithms might eventually lead to savings of very large sums of money by permitting more efficient use of congested transportation or communication systems. The actual problems in communication and transportation are so much complicated by timetables, safety requirements, signal-to-noise ratios, and economic requirements that in the past those seeking to solve them have not seen the basic simplicity of the problem, and have continued to use trial-and-error procedures which do not always give the true shortest path. However, in the case of a simple geometrical maze, the absence of these confusing factors permitted algorithms A, B, and C to be obtained, and from them a large number of extensions, elaborations, and modifications are obvious.

The problem was first solved in connection with Claude Shannon's maze-solving machine. When this machine was used with a maze which had more than one solution, a visitor asked why it had not been built to always find the shortest path. Shannon and I each attempted to find economical methods of doing this by machine. He found several methods suitable for analog computation, and I obtained these algorithms. Months later the applicability of these ideas to practical problems in communication and transportation systems was suggested.

REFERENCES

1. R. I. Wilkinson, "Abridged bibliography of articles on toll alternate routing," *Bell System Tech. J.* **35** (1956), 507.

2. L. Rosenfeld, "The determination of an optimal trucking route through a given traffic congestion pattern," *Proc. Western Joint Computer Conference* (1956), 80.