Probability in the Engineering and Informational Sciences, 4, 1990, 311–318. Printed in the U.S.A.

SEARCH IN A MAZE

S. GAL
IBM Israel Scientific Center
Technion City, Haifa, Israel 32000

E.J. ANDERSON
Institute of Management Studies
University of Cambridge
Cambridge, England CB2 1RX

Suppose that you find yourself trapped in a maze about which you know nothing except that it has an exit point. We present an optimal strategy that will lead you to the exit point in minimum expected time. This strategy ensures that the expected total length of the arcs you traverse will not exceed the sum of the lengths of the arcs in the maze.

1. INTRODUCTION

In this paper, we consider the problem of finding the exit in a maze. A maze is a connected finite network with two special nodes: the entrance, denoted by O, and the exit, denoted by E. We suppose that the searcher, who wishes to find the exit, moves on the network with unit speed and starts at the entrance. The searcher only has information about that part of the maze he or she has already traversed. Thus, we may think of the searcher carrying with him a map of the maze showing only the lengths of those arcs that have already been traversed, together with the number of untried arcs leading off each of the nodes that have been visited. Actually, the strategy we use requires only local information. Thus, it is sufficient for the searcher to leave marks at each node he visits, indicating the arcs that he traversed in the direction away from this node. This type of locally determined strategy is similar to some policies that could be employed in

 0269-9648/90 $5.00 + .00

a distributed computing environment to deal with incoming messages or queries directed to an unknown node of the computer network. Such a model was used, for example, by Golumbic [7], and our fixed permutation strategy might provide an alternative "local" path-finding mechanism.

Only graph-theoretic information is of relevance here. The nodes of the maze are not thought of as having any particular position in space, nor do we assume that the network is planar.

Efficient algorithms for searching mazes were already known in the 19th century (see Lucas [8] and Tarry [9]). An algorithm that can yield a better result in some cases but has the same worst-case performance is described by Fraenkel [3,4]. In Chapter 3 of [2], Even describes in an attractive fashion depth-first search algorithms that are well known in computer science; he notes that the depth-first search algorithm is actually a special case of Tarry's algorithm. Using this algorithm to find the exit point of the maze guarantees reaching it in a time not exceeding twice the sum of the arc lengths.

In [1] Anderson introduced the normal strategy and claimed, mistakenly, that it guarantees reaching the exit point in an expected time no more than the sum of the arc lengths. However, we show in Section 2 that the expected search time under the normal strategy can be as long as twice the sum of lengths of the arcs in the maze.

Throughout this paper, we will use expectations taken over the possible realizations of the search strategy, with the maze regarded as a fixed network, and the search strategy can use only the information about that part of the maze already traversed. In Section 3 we present a strategy, called the fixed permutation strategy, that ensures the expected total traversal length will not exceed the sum of the lengths of the arcs in the maze. We show that this strategy is optimal in the minimax sense.

The related problem of searching for an object in a known network was considered by Gal [5,6]. Except for two special cases, a Eulerian network or a tree, the optimal search strategy in a network is an open problem. It is therefore interesting that we have been able to obtain the optimal search strategy in an unknown network without any restriction on the type of network to be searched.

We now present some notation that will be used in this paper: The maze is denoted by M and we suppose that there are m arcs in M. The nodes, other than the entrance (but including E), are indexed by i, $i = 1,2,\ldots,I$. The length of an arc a is denoted by l_a. The total length of the arcs in the maze is denoted by L.

2. THE NORMAL STRATEGY AND A COUNTEREXAMPLE

The normal strategy for the searcher can be described as follows. At a node where there are untraversed arcs, choose one of these at random. At a node at which there are no untraversed arcs, choose the most recently traversed arc of those that have been traversed only once. (It can be shown [1] that the normal

strategy will always specify an arc to traverse until the searcher either reaches the exit or returns to the entrance after traversing each arc exactly twice.) This is a natural policy and probably corresponds to the strategy that would be followed by most people in practice. The normal strategy certainly ensures that the maze is explored in an orderly fashion. Nevertheless, it is a bad strategy to use. Indeed, the example we give below shows that it can lead to expected search times as much as twice as long as for the fixed permutation strategy.

First, we give a simple counterexample to Theorem 1 of Anderson [1]. Consider the maze shown in Figure 1 where

$$l_a = l_b = l_d = l_e = 1, \qquad l_c = y, \qquad \text{and} \qquad y \gg 1.$$

Then under the normal strategy, the expected time to reach the exit exceeds

$$(1/3) \times (2y) + (2/3)(1/2) \times (3 + 2y) > (4/3) \times y > y + 4 = L.$$

In fact, the expected time to find the exit in a maze using the normal strategy can be as close as desired to $2L$. Consider the maze shown in Figure 2, where the arc joining the entrance O to the node $n + 1$ is of length $y \gg 1$ and the other arcs are all of length $1/n$. Then the probability that the long arc (of length y) is traversed is given by

$$(1/3) + (2/3)(1/3) + (2/3)(2/3)(1/3) + \cdots + (2/3)^{n-1}(1/3)$$

which is arbitrarily close to 1 for n large enough. Since the long arc will be traversed twice if at all, and it constitutes as large a proportion of L as we like, this establishes the required property.

3. THE FIXED PERMUTATION STRATEGY, *S*

The fixed permutation strategy can be described as follows: On starting at O, choose a random permutation of the arcs that touch O and go along the first arc. Then on any subsequent return to O, the next arc from this fixed permutation will be used (even if, by that time, it has already been traversed in the direction that leads to O). Each time a node i is reached that has not previously been visited, choose a random permutation P_i of the unvisited arcs touching i,

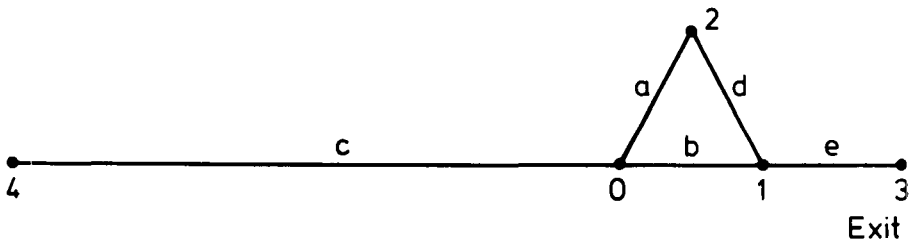

FIGURE 1.

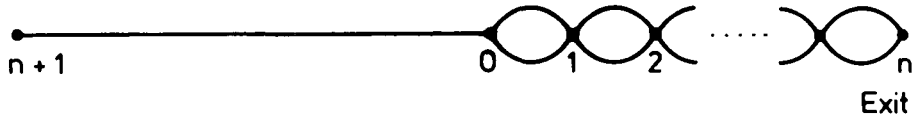

FIGURE 2.

independently of previous choices, and follow this (fixed) random permutation on each subsequent return to i (even if the arc that is scheduled to be used has already been traversed in the direction that leads to i). If when you reach a node i (other than O), all the arcs participating in P_i have been visited, go along the arc by which you first came to i.

An alternative and equivalent way of describing the strategy S is as follows: For any node i, write a_i for the arc by which the node i is first visited. Then observe that on each visit to a node i, the strategy effectively chooses at random from among those arcs that have not been traversed in the direction away from i, excluding a_i. If this set is empty, then the strategy chooses the arc a_i. It is not hard to show that if the degree of all the nodes in the maze is 3 or less and the degree of O does not exceed 2, then the normal and fixed permutation strategies coincide. Note that the fixed permutation strategy is a randomized version of Tarry's algorithm (see Fraenkel [3]).

Each traversal of an arc is called a step. We shall be interested in the behavior of the trajectories obtained by the realizations of S that do not stop at E but continue using S until they finish. We say that a trajectory finishes at a node i if, when node i is reached, all the arcs incident on i have already been traversed in the direction that leaves i. We shall see that all trajectories finish after $2m$ steps and that at this stage, the searcher is back at O and all the arcs of the maze have been visited twice.

For example, if the maze is given by Figure 1 above, then a possible $2m$-step realization of S is as follows: Starting at O, choose a permutation of the arcs a, b, and c, say, (b,a,c). Continue along b and when reaching node 1, choose a random permutation of the arcs d and e, say, (d,e). Then this $2m$-realization of S that does not stop at node 3 (the exit) is the trajectory T_1 that visits the following nodes in order:

$$O,1,2,O,2,1,3,1,O,4,O.$$

Notice that in this example on the second visit to O, the trajectory chooses an arc that has already been traversed, when an untraversed arc is available at that node. At first sight, this appears to be a foolish choice, since from the information available to the searcher, each untraversed arc is equally likely to lead to the exit and should be treated on equal footing. There appears to be no good reason for traveling to the untraversed arc at node 1 when there is an untraversed arc available at the current node. This type of consideration makes it

counterintuitive that, as we show below, the fixed permutation strategy is better than the normal strategy.

The proof of our main result on the expected time to reach the exit using the fixed permutation strategy requires two preliminary lemmas.

LEMMA 1: *Each realization using S and not stopping at E generates a trajectory T that, after 2m steps, finishes at O after visiting each arc of M exactly twice.*

PROOF: Note that such a realization uses Tarry's algorithm for visiting all arcs. Thus, Lemma 1 is simply a reformulation of Tarry's theorem (see Theorem 1 of Fraenkel [3]). ■

A trajectory that satisfies the conditions of Lemma 1 is called a *tour*. Obviously, the length of each tour is $2L$. The corollary below follows immediately from Lemma 1.

COROLLARY 1: *Using S guarantees that the time to reach the exit is smaller than* $2L$.

Let d_O be the degree of O and d_i the degree of node i.

LEMMA 2: *The number of different tours* K_t *satisfies*

$$K_t = d_O! \prod_{i=1}^{I} (d_i - 1)!$$

Each tour is obtained with an equal probability of $1/K_t$.

PROOF: When we begin at O, there are $d_O!$ equiprobable ways to start a trajectory by choosing different permutations of the arcs that touch O. On reaching any node i, which has not been visited before, there are $(d_i - 1)!$ ways to continue the trajectory because there are $d_i - 1$ unvisited arcs incident on i. All these continuations lead to different trajectories and are equiprobable and independent of the previous choices. Since each tour visits all the nodes of M, we have K_t equiprobable tours. ■

Each tour T corresponds to an event in the $I + 1$ permutation probability space. Denote this event by F_T. Denote the time to reach E going along tour T by r_T.

The proof of our main result below relies on an observation concerning the inverse of a tour. We denote the "inverse" tour for T (i.e., the trajectory that visits the arcs of T in the opposite direction) by T'. For example, the inverse trajectory T_1' of the tour T_1 given in Section 3 visits the following nodes in order:

$$O,4,O,1,3,1,2,O,2,1,O.$$

Obviously, if $T \neq T_1$, then $T' \neq T_1'$ and $(T')' = T$. Moreover, the inverse tour T' can also be obtained under the strategy S (to see that this is the case, notice that the first occasion a node is visited in T' corresponds to the last visit in T

and hence takes place along the arc on which the node is first visited in T). So by Lemma 2, the probability of T is equal to the probability of T' $(= 1/K_t)$. The essence of the proof is to observe that since the searcher is equally likely to choose T or T', the expected time to the first visit to the exit cannot be more than half the tour length.

THEOREM 1: *Using S, we determine that the expected time to reach E is no more than L.*

PROOF: We begin by dividing the set of events E_T into three disjoint sets Q, Q', and R where each $E_{T'}$ in Q' is an event that generates the inverse trajectory T' of a trajectory T with E_T in Q, and R consists of the events generating trajectories that are self-inverse. Hence, $\Pr(Q) = \Pr(Q')$.

The expected time to reach E is

$$\sum_{E_T} r_T \times \Pr(E_T)$$

$$= \sum_{T \text{ in } Q} r_T \times \Pr(T) + \sum_{T \text{ in } Q'} r_T \times \Pr(T) + \sum_{T \text{ in } R} r_T \times \Pr(T) \qquad (1)$$

$$= \sum_{T \text{ in } Q} (r_T + r_{T'}) \times \Pr(T) + \sum_{T \text{ in } R} r_T \times \Pr(T).$$

Here we have used the tour T as shorthand for the event E_T that generates it. Note that for each trajectory T, $r_T + r_{T'} \leq 2L$ because T' is actually the same trajectory in the opposite direction. Thus, for T in R, $r_T \leq L$. So the expected time to reach E is

$$\leq (2L) \times \Pr(Q) + L \times \Pr(R) = L. \qquad \blacksquare$$

COROLLARY 2: *If the exit node has a degree* 1, *then the expected time to reach E, if we use S, is L. If the degree of the exit node is greater than* 1, *then the expected time to reach E is smaller than L.*

PROOF: If the degree of the exit E is 1, then for each tour T, $r_T + r_{T'} = 2L$ because E is visited only once. Thus, in this case, the value of expression (1) is L. On the other hand, if the degree of E is greater than 1, then each tour visits E more than once. Thus, in this case, $r_T + r_{T'} < 2L$, which implies that the value of (1) is strictly smaller than L. $\blacksquare$

The corollary below follows easily from a similar consideration of tours and their inverses.

COROLLARY 3: *For each edge in the maze, the expected number of times it is traversed before exit is no more than* 1.

As we have seen already, when the normal strategy is used, the number of times that some specific arcs are traversed can be made arbitrarily close to 2 in expectation.

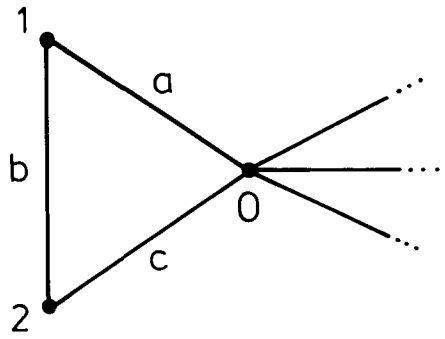

FIGURE 3.

Anderson [1] has defined a two-person game associated with this maze problem. We suppose that one player, the setter, chooses a network with a finite number of arcs and nodes having total length L and determines which nodes are the entrance and exit. The other player, the searcher, operates under the constraints described in Section 1 and wishes to reach the exit in minimal expected time, whereas the setter wishes to choose a network that will maximize the expected time for the solver to reach the exit. Any strategy for the searcher may require at least expected time L to reach E (take, for example, the trivial maze that consists of the two nodes O and E and a single arc connecting them). Since S guarantees an expected time not exceeding L, then it is an optimal strategy for the searcher.

It sometimes happens that strategies which are optimal in the minimax sense can be improved on by some strategies that have the same minimax value but, in some instances, produce better results. This is the case for our strategy S and two possible improvements for it can be outlined as follows:

1. If, at some stage, in a certain part of the maze all the arcs have been visited, then these arcs can be ignored later. For example, in the maze shown in Figure 3, if the trajectory starts with the nodes O, 1, 2, O, . . . , then the arcs a, b, and c can be ignored later.
2. Suppose that at some stage, being at node A, the searcher can determine that the trajectory dictates moving to another node B along arcs already traversed. If an alternative shorter route from A to B exists, then this should be taken, but in future decisions, the searcher should act as though the original longer route had been followed.

References

1. Anderson, E.J. (1981). Mazes: search games on unknown networks. *Networks* 11: 393–397.
2. Even, S. (1979). *Graph algorithms.* Rockville, MD: Computer Science Press.

3. Fraenkel, A.S. (1970). Economic traversal of labyrinths. *Mathematics Magazine* 43: 125–130.
4. Fraenkel, A.S. (1971). Economic traversal of labyrinths (correction). *Mathematics Magazine* 44: 12.
5. Gal, S. (1980). *Search games*. New York: Academic Press.
6. Gal. S. (1989). Continuous search games. In D.V. Chudnovsky & G.V. Chudnousky (eds.), *Search theory: Some recent developments*. New York: Marcel Dekker, pp. 33–53.
7. Golumbic, M.C. (1987). A general method for avoiding cycling in a network. *Information Processing Letters* 24: 251–253.
8. Lucas, E. (1882). *Recreations mathematiques*: Paris.
9. Tarry, G. (1895). Le problem des labyrinths. *Nouvelles Annales de Mathématiques* 14: 187.