An Example of Deterministic Walks on Random Environments

Rushil Mallarapu

November 16, 2019

Abstract

We study an example of a deterministic walk on a random environment. This environment consists of an infinite cylinder with bridges independently determined to be open or closed in each realization. We simulate this environment using Python. We seek answers to two basic questions relating to the periodicity of paths within this environment. The simulation is explained, and data from the program is tabulated. The dynamics of this problem for a simple case are then studied in further depth.

Introduction

We introduce a problem which is an example of a deterministic walk in a random environments. An infinite number of circles are stacked one above the other. Neighboring circles are connected by k edges or bridges. This is shown in Figure 1, with two bridges connecting each level.

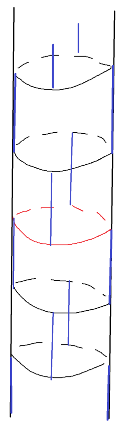

Figure 1: An example of our environment where $k = 2$. An arbitrary circle is chosen as the zeroth circle, shown here in red. The bridges are in blue.

The state of each bridge is either open or closed. The environment is random because the state of each bridge is a random variable. More precisely, in each realization of the environment, each bridge is open with probability p, independently of the state of the other bridges. In the case of $k = 2$, a possible realization of this environment is illustrated in Figure 2.

We place a particle in one of the circles. Given a realization of the environment, this particle travels along a deterministic path as follows: The particle moves in the counterclockwise direction, and each time it reaches an open bridge, it takes that bridge and moves to the corresponding neighboring circle. In the example of Figure 3, the particle starts at the red circle, and follows the path indicated by the arrows. Note that the particle eventually returns to the starting point (the red circle) and starts the same path again. It repeats the same path an infinite number of times. We will refer to such path as periodic paths. Note also that the particle goes around the cylinder 4 times in this example before it gets back to its original starting point.

Note that, while in this example the path is periodic, some paths are not periodic and may be unbounded, such as a path that has a single open bridge in every level.

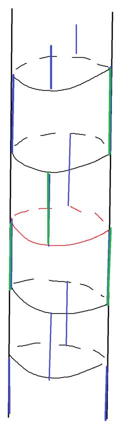

Figure 2: An example of bridges being open and closed on our environment where $k = 2$. In this realization, the green bridges are open and the blue ones remain closed.

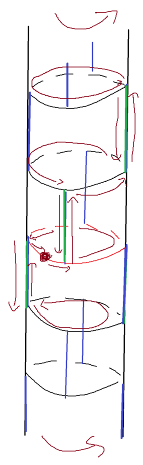

Figure 3: An example of a walk along our environment where $k = 2$. The dark red represents the path of the point. In this realization, the number of loops the point makes is 4.

Our goal is to understand the dynamics of such a system. More precisely, we seek to answer the questions listed below, and how these answers depend on k, the number of bridges connecting neighboring circles:

1. What is the probability that a path is periodic?
2. Given that the path is periodic, what is the expected number of times the particle goes around the cylinder before it returns to its starting point?

Now, to answer the first point, given that the environment is infinite, it is a nontrivial question as to whether the path will be periodic. However, if the probability p that any random bridge is open is constant, this will always be the case. The proof is shown below.

Proof that $\mathbb{P}(pathisperiodic) = 1$

Consider that for the path to be infinitely long, it is necessary that - without loss of generality - every positive level has at least one open bridge. Therefore, $\mathbb{P}(pathinfinite) \subset \mathbb{P}(everylevelhasatleastoneopenbridge)$. Now, the probability that all the k bridges at any given level are closed is p^k, where p is between zero and one. Therefore, $\mathbb{P}(nthlevelhasatleastoneopenbridge)$ is $1 - p^k$. Now, the probability that every level has one open bridge is the infinite product $\prod_{n=0}^{\infty} \mathbb{P}(\text{nthlevelhasatleastoneopenbridge})$, which is just $\prod_{n=0}^{\infty} 1 - p^k$. As $p^k < 1$, $1 - p^k < 1$, and is constant with respect to n. Therefore, this product evaluates to zero. Thus, $\mathbb{P}(\text{everylevelhasatleastoneopenbridge}) = 0$, which implies that $\mathbb{P}(\text{pathisinfinite})$ is zero. Ergo, $\mathbb{P}(\text{pathisperiodic}) = 1$, as desired.

Simulation of Problem

Python Simulator Code

```python
import random
import math

def next_bridge_3(p, s):
    n, k = p[0], p[1]
    i = k % (2*s)
    a = int(n - (1/2) + (1/2)*math.pow(-1, n + k))
    b = (i//2) % s
    return (a,b)

def find_bridge_prob_2(nb, Po):
    if nb in b:
        return b[nb]
    b[nb] = (random.randrange(100) < (Po*100))
    return b[nb]

def move_2(p, s, P):
    nb = next_bridge_3(p, s)
    if find_bridge_prob_2(nb, P):
        if p[0] == nb[0]:
            np = (p[0]+1, (p[1]+1)%(2*s))
        elif p[0] == nb[0]+1:
            np = (p[0]-1, (p[1]+1)%(2*s))
    if not find_bridge_prob_2(nb, P):
        np = (p[0], (p[1]+1)%(2*s))
    return np

N = 100000 #number of iterations
S = 2 #labyrinth size
P = 0.5 #probability of bridge being open

E = 0

for i in range(N):
    b = {}
    pv = []
    pi = (0,0)
    p = (0,0)
    while pi not in pv:
        p = move_2(p, S, P)
        pv.append(p)
    E += (len(pv))

print("The average number of loops is {}".format(E/(2*S*N)))
```

Explanation of Code

The simulation code consists of three functions and one main running block. We will first explain the broad method of operation of the program, and then delve into how this is coded for in Python.

The program follows the path of a particle along the environment by determining its positions along the cylinder, where positions are given as coordinate pairs where the first number is an integer indicating which circle of the labyrinth the particle is at, and the second is a positive number modulo $2k$ which represents what position along the circle the point is, where these positions are defined as the sectors between the intersections of bridges and the circle itself. The program starts with a counter at the origin point, $(0, 0)$, and in each iteration, it moves the point forward by one step. If the particle encounters a bridge which it has not come to before, it randomly assigns it a value of open or closed. If, however, the particle has encountered that bridge, it recalls its state from a dictionary generated as the program runs each iteration. Then, the particle moves across or through the bridge, and this continues until the list of points the particle crosses includes the origin point, indicating that the particle has arrived back to where it started. The size of the aforementioned list is then converted to the number of loops traversed in that iteration, which is averaged across all the iterations of the program.

The first function, "`next_bridge_3`", takes as an input a given point, defined as a tuple (n, k), and uses self-consistent modular arithmetic to assign this point a bridge, also represented by a coordinate pair of the level of the lower circle and its position around the circle, that a particle at this point would immediately encounter, regardless of the bridge being open or closed.

The next function, "`find_bridge_prob_2`", takes as an input a bridge, represented by a tuple, and checks whether it is open or not. It references a dictionary, b, which is generated for each iteration as the particle travels along, recording all the bridges it comes to. If the bridge is in b, the function returns a boolean, where "true" indicates the bridge is open. However, if the bridge is not in b, the function uses a random counter to assign a boolean to the bridge, which is then stored in b. This probability of the bridge being open is scalable. The benefit of this method is that only the bridges important to the path of the particle will be "realized" in order to save computational resources.

The final function, "`move_2`", simply takes in a point, finds its corresponding bridge, determines whether that bridge is open, and determines the next point a particle travelling along would move accordingly. The second coordinate always increments by one, as the particle makes one counterclockwise turn for every move it makes. If the bridge is closed, the first coordinate will stay the same. If, however, the bridge is open, the first coordinate will change, going down if the bridge connects below, and up if the bridge connects above.

The main block of code simply runs the program by initializing all the variables and going through the move sequence until the path of the particle comes back to where it starts and terminates the iteration. It also contains important control variables. N determines the number of iterations, which for testing purposes was set to 10^5. P determines the probability that any bridge will be open in a given realization. S determines the number of bridges connecting each level in the environment.

Table of Expected Values

The table below shows the expected value of loops for the labyrinth which has k bridges and a probability p that the bridge is open. For all instances, the program was run through 10^5 iterations, and the resulting number is shown to two decimal places.

k\p	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1	1.22	1.50	1.86	2.33	3.00	4.01	5.66	8.99	19.05
2	1.44	1.99	2.60	3.40	4.44	5.97	8.49	13.47	28.61
3	1.66	2.40	3.27	3.45	5.81	7.89	11.23	18.00	37.98
4	1.86	2.81	3.91	5.28	7.13	9.76	14.05	22.51	47.44
5	2.06	3.20	4.53	6.18	8.39	11.62	16.81	26.92	56.96
6	2.26	3.57	5.12	7.05	9.70	13.41	19.46	31.27	66.43
7	2.44	3.94	5.74	7.92	10.89	15.23	22.06	35.77	75.65
8	2.62	4.30	6.26	8.73	12.11	16.91	24.89	39.96	85.71

The Special Case of $k = 1$

The primary difficulty in directly determining the expected number of loops for an environment of connectedness k is in the fact that the number of adjacent open levels - the number of circles all connected to the starting circle by one or more bridge - is simply an upper bound on the path length, but does not define it.

However, for the case $k = 1$, this difficulty disappears. In this case, the number of adjacent open levels is exactly equivalent to the number of levels the particle will travel. Moreover, because the particle must travel once fully around every circle it reaches to take the bridge it came from and make it back to its starting point, and we are counting the length of the particles journey around circle 0, the number of loops of the particle for case $k = 1$ is exactly equal to the number of adjacent open levels to the starting circle.

With this information in hand, we can show why having the probability of a bridge being open set to one half results in the expected number of loops around the environment for $k = 1$ being exactly 3.

We need to determine the expected number of adjacent open levels. Note that the probability of having n adjacent open levels is $\frac{1}{2}^n$. Then, the "value" of having n adjacent open levels is n itself. Therefore, we have that the expected value of adjacent open levels is as follows:

$$\mathbb{E}(adjacentopenlevels) = \sum_{n=1}^{\infty} n \cdot \frac{1}{2}^n$$

This sum evaluates to 2. As mentioned above, the expected number of loops is one more than the expected number of adjacent open levels. Thus, the expected number of loops for a bridge probability of one half and $k = 1$ is 3, as it should.