Deterministic walks in a random cylinder

Tanav Choudhary,

A high school student at Singapore International School, Mumbai, India

Abstract:

In this paper, we consider the deterministic paths traversed by a particle in a randomly configured infinite cylinder. We derive a formula for the expected value of the number of loops that a particle takes around the cylinder in one period of the path given that the path is periodic.

Introduction:

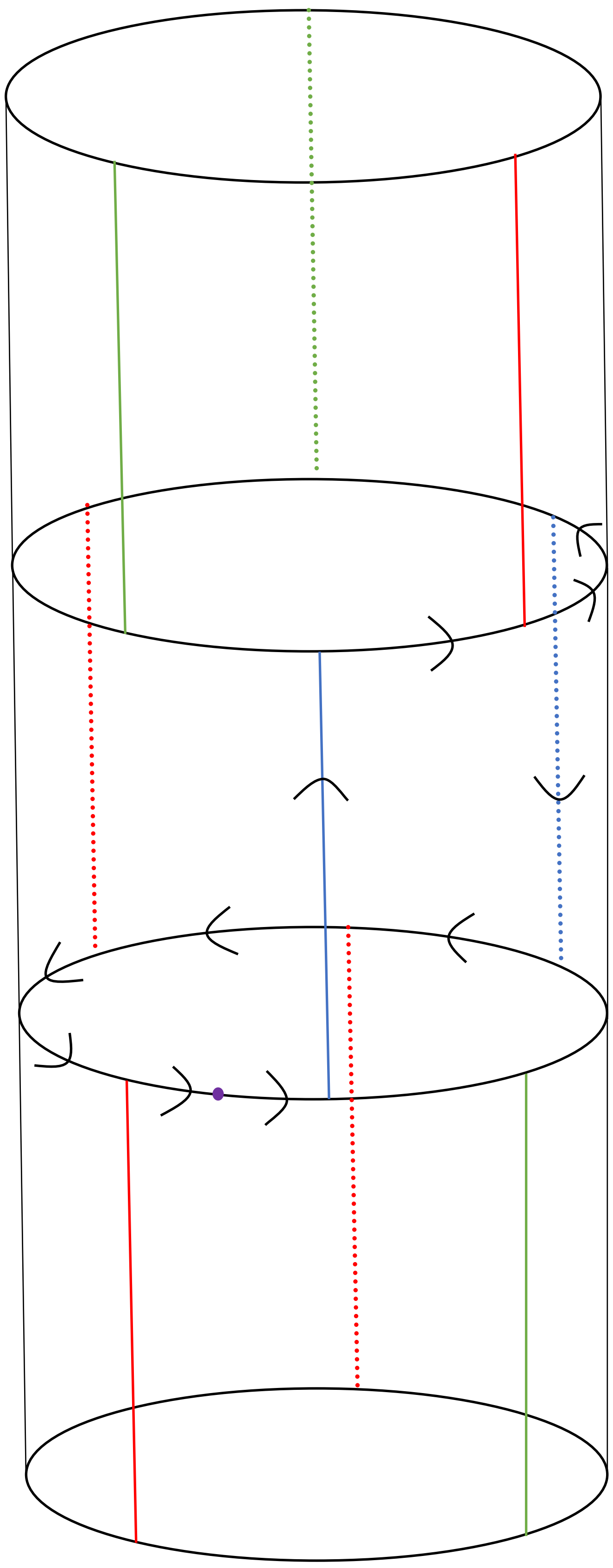
In this paper, we consider the problem of finding the expected number of loops a particle takes around a randomly configured infinite cylinder in one period of a periodic path. A detailed description of this problem will be given later. We are not aware of any other work that considers this problem. We consider this problem due to our interest in a particular class of problems known as deterministic walks in random environments [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. Some of the most prominent open problems in this area of research are in a 2 – dimensional context, and the techniques used in 1-dimensional results do not directly generalize to problems in 2 dimensions. The problem we solve in this paper also belongs to deterministic walks in random environments, though it is slightly different from those considered in other works. This problem shares features with 2-dimensional problems, but it has proven to be more amenable to analysis than them. We hope that the techniques and ideas we introduce in this paper will spark new strategies to solve the outstanding open problems in this field that have eluded researchers for several decades.

Deterministic Walks in random environments (DWRE) occupy an intermediate position between purely random and purely deterministic processes. These models combine random and deterministic properties in a very special way, making the study of this topic immensely interesting. In a DWRE, the initial configuration of a particular system is determined in a completely random manner using a given probability distribution; however, once the initial configuration is determined, the system follows a specific deterministic rule. These models have several important applications such as in analyzing the Lorentz Lattice Gas (LLG) [1, 2, 3, 6]. 1-dimensional DWRE models have proven to be quite amenable to analysis. Define Rigidity as an integer-valued function $r(z)$ where z is a vertex of the underlying graph of the DWRE. $r(z)$ represents the number of times a particle must visit a vertex in order to change the types of the edges (i.e. open or closed) connected to the vertex z. In fact, 1-dimensional models with constant rigidity have been found to be completely solvable [6]. Furthermore, 1-dimensional models with non-constant rigidity have also been attacked in the past, and several results are known about them such as Theorem 5 in [6]. However, as mentioned earlier, these 1-dimensional results do not directly generalize to problems in two dimensions, such as DWRE models on a regular 2 – dimensional lattice. The model we consider, an infinite cylinder, is a 2-dimensional model, but it is different from a regular lattice in the sense that in a 2D lattice, both of the dimensions extend infinitely but in the infinite cylinder, the vertical dimension extends infinitely whereas the horizontal dimension is a finite compact dimension. Hence, our model is closer to the other 2-dimensional models compared to the 1-dimensional models, due to which we believe that our techniques would generalize to other 2-dimensional DWRE problems more directly.

Now we give a detailed description of the problem considered in our paper. Please refer to figure 1 while reading the description of the problem. We shall begin by describing the infinite graph which forms the base of the model studied in this paper. Firstly one takes an infinite sequence of identical cycles $\ldots, C_{2c}^{(i-1)}, C_{2c}^{(i)}, C_{2c}^{(i+1)}, \ldots$ on $2c$ vertices, given by a parameter c. For odd i all odd vertices in the cycle $C_{2c}^{(i)}$ are connected to the corresponding vertex of $C_{2c}^{(i+1)}$(i.e. there is an edge between the first vertex in $C_{2c}^{(i)}$ and the first vertex of $C_{2c}^{(i+1)}$, the 3rd vertex of $C_{2c}^{(i)}$....). Likewise if i is even, then even vertices in the cycles $C_{2c}^{(i)}$ and $C_{2c}^{(i+1)}$ are connected. Bond percolation is then performed only on the edges connecting the cycles, each such edge is retained with probability 1/2. A deterministic random walk is then run on the percolated cylinder, the rule this walk follows is to rotate around the cycles counter-clockwise alternately trying to take edges going "up" or "down" between cycles. Each of the cycles $C_{2c}^{(i)}$ are referred to as “levels” in the rest of the paper. The edges connecting the cycles in the underlying infinite graph prior to bond percolation are referred to as “connections” in the rest of this paper. A connection is said to be open if it is retained after the bond percolation. Similarly, a connection is said to be closed if it is not retained after the bond percolation. Moreover, we refer to a deterministic random walk on the percolated cylinder as a path. Note that our system has infinite rigidity, which means that once the state of a connection is determined, the state of the connection will remain the same for the rest of the walk i.e. an open connection will remain open for the entire path and a closed connection will remain closed for the entire path. Furthermore, note that vertices on each cycle are equally spaced from each other. Also note that the particle that will trace a path on this percolated cylinder starts on a level such that there are infinitely many levels both above and below it. The problem considered in our paper is to find an exact formula for the expected number of loops taken by a particle in one period of a periodic path on this percolated cylinder in terms of c. We have given an illustrative example of a path on the infinite percolated cylinder described above in the next page.

It is not very difficult to see that a path in this environment is periodic with probability 1. First, notice that if the space which is accessible to the particle is bounded, then the path of the particle is periodic, since the particle can visit only finitely many connections and the state of the connections remain the same. The space accessible to a particle is guaranteed to be bounded if there exists a level above it such that all of the connections to that level from the level below it are closed, and if there exists a level below it such that all of the connections to that level from the level above it are closed. Evidently, such levels exist with probability 1 because if such levels did not exist, then there would have to be at least one open connection between every pair of adjacent levels, which has probability 0, as we shall show below.

Fig 1.



This is an example of a path on the infinite cylinder described earlier, where c = 3. Note that there are 3 connections between each pair of adjacent levels in this example (the black lines are a part of the border of the cylinder; they are not connections). Green lines depict unvisited connections. Blue lines depict connections that the particle visits which are open. Red lines depict connections that the particle visits which are closed. The purple dot on the cylinder represents the starting point of the particle. The arrows represent the path of the particle. Note that this is a periodic path, and the number of loops the particle takes in each period of the path is equal to 1. This is because in one period of the path, the particle goes around the cylinder exactly once, as can be seen in the diagram. Note that the difference between the solid lines and the dashed lines is simply that the solid lines depict the lines in the foreground of the cylinder and the dashed lines depict the lines in the background of the cylinder.

Now we explain why the probability that there is at least one connection between each pair of adjacent levels equal to 0. Note that the probability that all connections between two adjacent levels is closed is $\frac{1}{2^c}$, where c is the number of connections. So the probability that at least one connection is open is $\left(1-\frac{1}{2^c}\right)$, since at least one path being open is the complement of all paths being closed. So, if there are n pairs of adjacent levels, the probability that there is at least one connection between each of these pairs of levels is $\left(1-\frac{1}{2^c}\right)^n$, since the parity of different connections are independent of each other. Recall that we are considering a cylinder with infinitely many levels, so n tends to infinity. Although $\left(1-\frac{1}{2^c}\right)$ can get very close to 1, it is still less than 1 since there are a finite number of connections. Since it is less than 1, raising it to the power of n where n tends to infinity will give us 0, as desired.

Numerical Simulations:

We have carried out numerical simulations for walks to give the reader a taste of what are the expected values of the number of loops approximately equal to, before we jump to the main result. The results of the numerical simulations are given in the table below. Note that to compute an approximation for the expected value of the number of loops, we took an average of the number of loops for 100,000 random simulations.

Number of connections (c)	Approximation for the expected value of the number of loops
1	3.00194
2	4.43459
3	5.81465
4	7.11357
5	8.40251
6	9.65118

Now we will explain how we carried out the numerical simulations. Note that since we are dealing with an infinite cylinder, we cannot determine at once whether all connections in the system are open or closed. Hence, we determine whether a connection is open or closed as we pass it. Before each time a particle passes a connection, we check whether the particle has already passed that connection with the help of a dictionary of the coordinates of all the connections passed. If the connection was previously traversed, then we need to ensure that the parity of the connection is the same as before i.e. if the connection was closed, then it will remain closed and if it was open, it will remain open. Note that this is an important distinction between a completely random walk and a deterministic walk in a random environment. In a completely random walk, the parity of the connection will be randomly determined using the given probability distribution each time we pass the connection. However, in a deterministic walk in a random environment, only the initial parity of the connection (i.e. whether the connection is open or closed) is randomly determined using the given probability distribution. After that, the parity of the connection is governed by a deterministic rule, which in this case is that the parity of the connection remains the same as its initial state even if the particle traverses it again.

Coming back to the numerical simulation, if the connection was not previously traversed, then we randomly determine the parity of the connection such that the probability of it being closed is ½ and the probability of it being open is also ½. So, now we know the parity of the connection being traversed regardless of whether it was traversed earlier or not, hence now we can determine the coordinates of the next connection that we will traverse, and if the connection we just passed was not previously traversed, we would add it to the dictionary mentioned above. If the connection is closed, we would stay on the same level and move in the counter-clockwise (the direction was chosen arbitrarily) direction to the next connection.

Preliminary Definitions:

Adjacency Matrix – The adjacency matrix is a square matrix used to represent a finite graph. The number of rows and the number of columns of the matrix are each equal to the number of vertices of the finite graph. Each element of the adjacency matrix indicates whether there is an edge between a pair of vertices in a graph: The element is equal to 1 if such an edge exists but the element is equal to 0 if such an edge does not exist. Note that the adjacency matrix is a symmetric matrix, because if there is an edge connecting the second vertex to the third vertex, the same edge also connects the third vertex to the second vertex. However, this is not the case for the adjacency matrix of a directed graph, where each edge is assigned an orientation. In these kinds of graphs, each element of the adjacency matrix indicates whether there is an edge **from** a particular vertex **to** another vertex.

Example:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Consider this adjacency matrix of a directed graph. Note that the corresponding directed graph has exactly 3 vertices, since the number of rows and columns are 3 each. $A_{1,3}$ = 0, so there is no edge from the first vertex to the third vertex. However, $A_{3,1}$ = 1, so there is an edge from the third vertex to the first vertex. One property of adjacency matrices is that higher powers of the adjacency matrix gives the number of paths of a particular length from one vertex to another. Considering our previous example, $A^2_{1,3}$ is equal to the number of paths of length 2 from the first vertex to the third vertex. Similarly, $A^3_{1,3}$ is equal to the number of paths of length 3 from the first vertex to the third vertex, and so on. This property is pivotal for our solution to the problem being addressed by this research paper.

De Bruijn Graph – For the purpose of this paper, we are restricting the definition of De Bruijn Graphs to only those which concern binary strings. In this case the n – dimensional De Bruijn Graph is a directed graph having 2^n vertices, where each vertex represents a binary string of length n. There exists an edge from vertex A to vertex B if and only if the last n – 1 digits of the binary string represented by vertex A are the same as the first n – 1 digits of the binary string represented by vertex B. Concretely, what this means is that the second digit of the binary string of vertex A is equal to the first digit of the binary string of vertex B, the third digit of the binary string of vertex A is equal to the second digit of vertex B, and so on. This property of a De Bruijn graph allows us to form any binary string by traversing a path on the graph. For instance, in the 4 – dimensional De Bruijn graph, the binary string 101101 can be formed by starting at the 1011 vertex, then going to the 0110 vertex, and then ending the path on the 1101 vertex. Furthermore, each vertex is labelled according to the base 10 value of the binary string. For instance, 0010 has a base 10 value of 2, so it's the third vertex in the 4 – dimensional De Bruijn Graph (0000 is the first vertex). Now let's consider the adjacency matrix of a De Bruijn Graph. Observe that for an n – dimensional De Bruijn Graph, the binary string with base 10 value of x will only lead to binary strings whose base 10 value is either 2x + 1 (mod 2^n) or 2x (mod 2^n). This is because multiplying by 2 causes all of the binary digits to shift one place to the left, and we can choose the last digit to be either 1 or 0, (hence corresponding to 2x + 1 and 2x respectively). Now, if we perform mod 2^n, we basically remove the leftmost digit of the new binary string, so the last n – 1 digits become the first n – 1 digits, as desired. Hence the adjacency matrix of the n – dimensional De Bruijn Graph is such that for a given value of i, the value of the element is 1 if and only if $j \equiv 2i \ (mod\ 2^n)$ or $j \equiv 2i - 1 \ (mod\ 2^n)$. This is because the ith vertex corresponds to the binary string whose base 10 value is i – 1. So, by the result previously established, the base 10 values of the binary strings which this vertex leads to are 2i – 2 (mod 2^n) and 2i – 1 (mod 2^n). Hence, these correspond to the (2i)th and the (2i – 1)th vertices, as desired.

Example:

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Above is the adjacency matrix of the 4 – dimensional De Bruijn Graph.

Main Result:

In our paper, we show that the expected number of loops the particle takes around the random cylinder in one period of a periodic path is given by the following formula:

$$\frac{\binom{2c-1}{c-1}}{2^{2c}} + \sum_{n=2}^{\infty} n \frac{f_c\left(M_{n,c}^{2c}\right) + \sum_k (2^k - 1) h_k\left(P_{n,c}\right)}{2^{2nc}}$$

We shall now describe all of the terms in the formula above. Over here, the parameter c is the number of connections between two adjacent levels. To define the other entities used in this formula, we must define certain auxiliary objects. Given a binary string, let the *r-value* of each digit *d* be $d \times (-1)^m$, where *m* is the number of 0s in the binary string which are to the left of the digit d. Furthermore, let a binary string be called *good* if the sum of r-values of its digits is 0.

Now, we define a sequence of matrices $A_1, A_2 \ldots$ such that A_n is a $2^{2n} \times 2^{2n}$ matrix where $(A_n)_{ij}$ = 1 if and only if $j \equiv 2i \ (mod\ 2^{2n})$ or $j \equiv 2i - 1 \ (mod\ 2^{2n})$. After that, we define a sequence of matrices $T_1, T_2 \ldots$ such that to obtain T_n, you need to take A_n and change $(A_n)_{ij}$ to 0 if i or j correspond to a *good* binary string.

Now, we shall define an operation we call the skewed product, which is denoted by the symbol $\times^{Tc}$. Given two $2^{2nc} \times 2^{2nc}$ matrices A and B, consider the elements A_{ij} and B_{ij} such that $(i-1) \cdot 2^{2c} \ (mod\ 2^{2nc}) < j \leq i \cdot 2^{2c} \ (mod\ 2^{2nc})$. If $i \cdot 2^{2c} \ (mod\ 2^{2nc})$ = 0, then change it to 2^{2nc} in the inequality . It is evident that there are $2^{2(n+1)c}$ such ordered pairs (i, j). We call these ordered pairs *skewed.* Then, C = A $\times^{Tc}$ B is a $2^{2(n+1)c} \times 2^{2(n+1)c}$ matrix such that C_{ij} = $A_{i_1 j_1} \cdot B_{i_2 j_2}$, where (i_1, j_1) is the i^{th} *skewed* ordered pair in dictionary order and (i_2, j_2) is the j^{th} *skewed* ordered pair in dictionary order. Now we are ready to define the matrices $M_{n,c}$. The matrix $M_{n,c}$ can be defined recursively by:

$$M_{2,c} = T_c \text{ and } M_{(n+1),c} = (M_{n,c}^{2c} \times^{Tc} M_{n,c}^{2c}) \circ T_{nc}$$

where $\circ$ denotes the Hadamard product or element-wise product of two matrices and $\times^{Tc}$ denotes the skewed product which was defined above.

Now, let X_c be the set of all ordered pairs (i, j) such that if we concatenate the first 2c digits of the binary string corresponding to i with the binary string corresponding to j, the resulting binary string is *good.* Then, we define the function $f_c(M)$, where M is a matrix, such that $f_c(M) = \sum_{(i,j) \in X_c} M_{ij}$.

For a positive integer k and a matrix M, the function h_k is such that $h_k(\mathrm{M})$ is equal to the number of entries M_{ij} in the matrix which are equal to k and such that concatenating the binary strings corresponding to i and j gives us a good binary string. Thus, in the main formula, k takes on all positive integer values q such that at least one entry in the matrix $P_{n,c}$ is equal to q.

Call a binary string as *interesting* if its length is a multiple of 2c and its sum of r-values is 1. Then, we define $P_{n,c}$ to be the $2^{2(n-1)c} \times 2^{2(n-1)c}$ matrix such that $(P_{n,c})_{ij}$ is the number of interesting proper substrings of the binary string of length 2nc which results from concatenating the binary strings corresponding to i and j. If it is not possible to concatenate these binary strings or if the resulting binary string contains a good proper substring, then the entry of the matrix at (i, j) will be 0. Note that we will derive a recursive method to compute the matrices $P_{n,c}$ later in the paper.

Derivation of the result:

Let c be the number of connections between two adjacent levels. The particle starts at a point between two connections, i.e. the particle cannot start at a connection. Let a *move* be a movement of the particle by a distance of 1/2c times the circumference of the cylinder around the cylinder counter-clockwise (so a loop is basically 2c moves). Hence, the particle passes exactly one connection each move. First, note that given a position p of the particle which is between two connections, there is only one position q from which the particle can reach the position p in exactly one move, given that we know whether the connection adjacent to p in the clockwise direction is open or closed. Therefore, the path of the particle is periodic if and only if the particle revisits its initial point.

The Case with One Connection:

It is easy to see from the numerical simulations that when the number of connections is 1, the expected value of the number of loops seems to be very close to 3. In fact, the expected value of the number of loops is exactly equal to 3, which will be shown in this section. Notice that the particle must take an entire loop of a level if it visits that level. This is not very hard to see. Firstly, when a particle visits a level, it automatically traverses half of that level since each level is only adjacent to two connections. Then if the next connection is closed, then the particle traverses the other half of the level as well, completing the loop. If the next connection is open, then the particle will move one level away from the initial level, and since there is exactly one connection between two adjacent levels, the particle must revisit that connection since it needs to revisit the initial point (recall the observation made earlier – the path of the particle is periodic if and only if it revisits its initial point). When the particle traverses that connection for the second time, it traverses the other half of the level, completing the loop, as desired. Since the particle takes a loop of every level it visits, the expected value of the number of loops is equal to the expected number of levels visited by the particle, as desired.

Notice that the particle must take a loop of the initial level in order to revisit its initial point. Now we will find the expected number of levels the particle visits above the initial level, and using symmetry this number will also be equal to the expected number of levels the particle visits below the initial level. Recall that each connection is open with a probability of 1/2. So, the particle visits no level above the initial level if the first connection is closed, which has probability 1/2. If the particle visits exactly one level, the first connection must be open and the second connection must be closed, which has a probability of $(1/2)^2 = 1/4$. If the particle visits exactly two levels, the first 2 connections must be open but the third connection must be closed, which has a probability of 1/8, and so on. So, the expected value of levels visited above the initial level is:

$$\sum_{n=1}^{\infty} \frac{n-1}{2^n} = \sum_{n=2}^{\infty} \frac{n-1}{2^n} = \sum_{n=1}^{\infty} \frac{n}{2^{n+1}}$$

Thus, let $S = \sum_{n=1}^{\infty} \frac{n-1}{2^n} = \sum_{n=1}^{\infty} \frac{n}{2^{n+1}}$. Note that:

$$S = \sum_{n=1}^{\infty} \frac{n-1}{2^n} = \sum_{n=1}^{\infty} \frac{n}{2^n} - \sum_{n=1}^{\infty} \frac{1}{2^n} = 2\sum_{n=1}^{\infty} \frac{n}{2^{n+1}} - 1 = 2S - 1$$

This implies that $S = 1$.

Thus, we have shown that the expected value of levels visited above the initial level is 1. Using symmetry, the expected value of levels visited below the initial level is also equal to 1. So, the expected number of levels visited is 1 + 2(1) = 3 (recall that the particle also takes a loop around the initial level in order to revisit its initial point, as mentioned earlier). So, the expected value of the number of loops of the particle is equal to 3 when there is one connection, as desired.

The General Case:

Now, we establish a bijection between a path of the particle and a binary string, where a 0 represents the particle passing a closed connection and a 1 represents the particle passing an open connection, and the binary string ends when the initial point is revisited. It was previously mentioned that the path of the particle is periodic if and only if the particle revisits its initial point. This implies two properties about a binary string corresponding to a periodic path. Firstly, it implies that the length l of the binary string must be a multiple of 2c, because the number of moves the particle must make in order to revisit its initial point is a multiple of 2c. The second property is as follows.

Given a binary string, let the *r-value* of each digit *d* be $d \times (-1)^m$, where *m* is the number of 0s in the binary string which are to the left of the digit d. If the path of the particle is periodic, then the sum of the *r-values* of all of the digits in the corresponding binary string is 0.

The property above follows from the idea that after a closed connection is encountered, the direction in which the particle moves through open connections (i.e. up or down) gets reversed. Furthermore, the sum of r-values being 0 is equivalent to the particle returning to its initial level. Hence, note that the two properties mentioned above are simply equivalent to the particle revisiting its initial point, hence these two properties are equivalent to the path of the particle being periodic.

However, we must note that there exist binary strings that fulfill the two properties above but don't correspond to any path on the cylinder. This problem may arise due to 2 reasons. Firstly, the initial point may be revisited before the end of the string, contradicting a major condition involved in the bijection. Secondly, the same connection may be treated as both open and closed in a binary string. For example, if c = 3, the binary string 001000100000 corresponds to a path which treats the same connection as both open and closed. Note that the same connection being treated as both closed and open is equivalent to the particle revisiting a particular point which is not the initial point. Hence, we can combine the two reasons mentioned above into one condition which is that the particle cannot revisit any point before the end of the path. To comply with the condition in the previous sentence, we can impose a simple restriction on the binary string which is mentioned below.

The binary string cannot have any proper substring which fulfills both of the properties mentioned below:

(1) Its length is a multiple of 2c
(2) The sum of r-values of its digits is 0

Hence, this restriction prevents the particle from visiting the same point more than once, as it prevents any proper substring from corresponding to a closed path.

Now, we need to count the number of binary strings which satisfy conditions (1) and (2) above in addition to the following condition:

(3) It does not contain any proper substring that satisfies both conditions (1) and (2)

A binary string is said to be *true* if it satisfies the three conditions mentioned above

Proposition 1: The number of *true* binary strings of length 2c is $\binom{2c-1}{c-1}$.

Proof:

Let a binary string be called *good* if the sum of r-values of its digits is 0. If the length of the string is exactly 2c, then the third condition is always satisfied. Hence we just need to count the *good* binary strings of length 2c. It is evident that the number of ones in the binary string must be even for the sum of r-values to be 0. So let the number of ones be 2k. Now define the sequence $a_1, a_2 \ldots a_{2c-2k+1}$ such that a_1 is the number of ones before the first 0, a_2 is the number of ones between the first 0 and the second 0, and so on, and $a_{2c-2k+1}$ is the number of ones after the last 0. For the sum of r-values to be 0, $a_1 + a_3 + \cdots + a_{2c-2k+1}$ must be equal to $a_2 + a_4 + \cdots + a_{2c-2k}$. Since the total number of ones in the binary string is 2k, both of the sums must be equal to k. Using stars and bars, the number of possibilities of $a_1 + a_3 + \cdots + a_{2c-2k+1}$ = k is $\binom{c}{k}$. Similarly, using stars and bars, the number of possibilities of $a_2 + a_4 + \cdots + a_{2c-2k}$ = k is $\binom{c-1}{k}$. Since both of the equations are independent of each other, the total number of *good* binary strings of length 2c having 2k ones is $\binom{c}{k}\binom{c-1}{k}$. Note that k can be any value except c as for the binary string to be *good*, there must be at least one 0 so that the sign of the r-values of the digits changes at least once in the string. Hence, the total number of *good* binary strings of length 2c = $\sum_{k=0}^{c-1}\binom{c}{k}\binom{c-1}{k}$. Using Vandermonde's identity, $\sum_{k=0}^{c-1}\binom{c}{k}\binom{c-1}{k}$ = $\binom{2c-1}{c-1}$, as desired. Q.E.D

Hence we have found the number of *true* strings of length 2c. However, if the length is not 2c, then the third condition applies due to which this method will not work if the length is greater than 2c. Below, we consider the case where the length of the binary string is a multiple of 2c which is strictly greater than 2c.

First, we define a sequence of matrices A_1, A_2 ... such that A_n is a $2^{2n} \times 2^{2n}$ matrix where $(A_n)_{ij}$ = 1 if and only if $j \equiv 2i \ (mod\ 2^{2n})$ or $j \equiv 2i - 1 \ (mod\ 2^{2n})$. Recall that A_n is simply the adjacency matrix for the directed De Bruijn graph which contains all binary strings of length $2n$ as its vertices and the index corresponding to each binary string is simply its value in base 10. For example $(A_n)_{12}$ corresponds to the whether there is an edge from the binary string of length 2n with a denary (base 10) value of 1 to the binary string of length 2n with a denary value of 2. Now we define a sequence of matrices T_1, T_2 ... such that to obtain T_n, you need to take A_n and change $(A_n)_{ij}$ to 0 if i or j correspond to a *good* binary string. Note that T_n is simply the adjacency matrix of the directed De Bruijn graph from which we have removed all vertices corresponding to *good* binary strings. Now, let X_c be the set of all ordered pairs (i, j) such that if we concatenate the first 2c digits of the binary string corresponding to i with the binary string corresponding to j, the resulting binary string is *good.* Lastly, we define the function $f_c(M)$, where M is a matrix, such that $f_c(M)$ = $\sum_{(i,j)\in X_c} M_{ij}$. It is easy to see that $f_c(T_c^{2c})$ gives us the number of strings of length 4c that satisfy the three properties mentioned earlier, as this follows directly from the definitions of T_n and f_c, and the fact that raising the adjacency matrix of a graph to a particular number m gives us the number of paths of length m for each ordered pair of start and end vertices.

Proposition 2: $f_c(T_c^{2c})$ gives us the number of *true* strings of length 4c.

Proof:

Note that $(T_c^{2c})_{ij}$ gives us the number of paths of length 2c from the vertex corresponding to i to the vertex corresponding to j which does not pass through any vertex corresponding to a good binary string. Since each vertex is a binary string of length 2c, each path corresponds to a binary string of length 4c not containing any *good* binary string of length 2c. Hence, $f_c(T_c^{2c})$ gives us the number of *good* binary strings of length 4c which do not have any *good* proper substring of length 2c, as desired. Q.E.D

Now we will create a sequence of matrices $M_{n,c}$ for n $\geq$ 2 such that $f_c\left(M_{n,c}^{2c}\right)$ gives the number of *true* binary strings of length 2nc. By proposition 2, we know that $M_{2,c} = T_c$. Before we find $M_{n,c}$ for all n $\geq$ 2, we must define a particular operation on matrices called as the Skewed product denoted by the symbol $\times^{Tc}$.

Given two $2^{2nc} \times 2^{2nc}$ matrices A and B, consider the elements A_{ij} and B_{ij} such that $(i-1) \cdot 2^{2c}\ (mod\ 2^{2nc}) < j \leq i \cdot 2^{2c}\ (mod\ 2^{2nc})$. If $i \cdot 2^{2c}\ (mod\ 2^{2nc})$ = 0, then change it to 2^{2nc} in the inequality . It is evident that there are $2^{2(n+1)c}$ such ordered pairs (i, j). Call these ordered pairs as *skewed.* Then, C = A $\times^{Tc}$ B is a $2^{2(n+1)c} \times 2^{2(n+1)c}$ matrix such that C_{ij} = $A_{i_1 j_1} \cdot B_{i_2 j_2}$, where (i_1, j_1) is the i^{th} *skewed* ordered pair in dictionary order and (i_2, j_2) is the j^{th} *skewed* ordered pair in dictionary order. The reason behind taking into account only skewed ordered pairs is that these are the only pairs for which it is possible to concatenate the two binary strings. This is because the last 2(n – 1)c digits of the first string must be the same as the first 2(n – 1)c digits of the second string and the length of each string is 2nc. So we must first multiply the base 10 value of the first string with 2^{2c} to shift the digits 2c places to the left, and then we take its mod 2^{2nc} to remove the first 2c digits, so that the last 2(n – 1)c digits of the first binary string become the first 2(n – 1)c digits of the second binary string, as desired. The expression for j is an inequality because the last 2c digits of the second binary string can be anything, hence exactly 2^{2c} values of j are permitted, as can be seen in the inequality above. Note that since we have considered the skewed ordered pairs in dictionary order, they represent the binary strings of length 2(n + 1)c in their normal order (i.e. in the order of their base 10 values) because i is the first string in the concatenation, so if the i values are different, the j values don't play any role in determining which binary string has a larger base 10 value, just as in dictionary order.

Let $M_{n,c}$ be the $2^{2(n-1)c} \times 2^{2(n-1)c}$ adjacency matrix of the directed De Bruijn graph of binary strings of length 2(n – 1)c which are not *good* and do not contain any *good* proper substring whose length is a multiple of 2c (note that $M_{n,c}$ does contain binary strings which are good or which have good proper substrings as vertices, but all the edges connected to these binary strings are removed). Let the binary strings be indexed according to their value in base 10. This definition of $M_{n,c}$ is consistent with $f_c\left(M_{n,c}^{2c}\right)$ giving the number of *true* strings of length 2nc and it is also consistent with $M_{2,c} = T_c$, so all that is left now is to create an equation which gives $M_{n,c}$. Since we know $M_{2,c}$, we can find $M_{n,c}$ for all n > 2 by finding an appropriate recursive formula, which is done below.

Proposition 3: $M_{(n+1),c}$ = $(M_{n,c}^{2c} \times^{Tc} M_{n,c}^{2c}) \circ T_{nc}$, where ∘ denotes the Hadamard product or element-wise product of two matrices.

Proof:

Note that $\left(M_{n,c}^{2c}\right)_{ij}$ gives the number of binary strings of length 2nc which do not have any *good* proper substring whose length is a multiple of 2c, and which starts and ends with particular strings of length 2(n - 1)c, namely i and j respectively. Since n ≥ 2, 2(n – 1)c $\geq nc$. Therefore, there is at most one binary string of length 2nc which starts with a particular binary string of length 2(n - 1)c and ends with a particular binary string of length 2(n - 1)c. This is because both of these strings take up at least half of the binary string of length 2nc as shown by the inequality above, due to which every digit of the binary string of length 2nc will be uniquely determined by these two binary strings. As a result, $\left(M_{n,c}^{2c}\right)_{ij}$ = 0 or 1.

T_{nc} is the adjacency matrix of the directed De Bruijn graph of the binary strings of length 2nc which are not *good*. Hence, to get $M_{(n+1),c}$ from T_{nc} , we need to remove all the binary strings which contain a *good* proper substring whose length is a multiple of 2c from the graph of T_{nc}. Hence, $(T_{nc})_{ij}$ must be changed to 0 if either of the binary strings corresponding to i or j have a *good* proper substring of length 2c. It follows from the definition of the skewed product that each index (i, j) of the skewed product of $M_{n,c}^{2c}$ with itself corresponds to an ordered pair of the ith and jth binary strings of length 2nc, since the skewed ordered pairs in dictionary order correspond to the binary strings of length 2nc in their normal order (i.e. according to their values in base 10). Furthermore, note that $(M_{n,c}^{2c} \times^{Tc} M_{n,c}^{2c})_{ij}$ = 1 if and only if the ith and jth binary strings of length 2nc don't contain any *good* proper substrings whose length is a multiple of 2c. Hence, taking the Hadamard product of T_{nc} with $(M_{n,c}^{2c} \times^{Tc} M_{n,c}^{2c})$ will change $(T_{nc})_{ij}$ to 0 if and only if either of the ith and jth binary strings of length 2nc have a *good* proper substring of length 2c, as desired.

Q.E.D

Now, at first glance, it may seem that we can compute the expected value by simply knowing the number of true strings of lengths which are a multiple of 2c, which is equal to the number of valid paths on the infinite cylinder of the corresponding number of loops. Recall that the corresponding number of loops is the length of the string divided by 2c. However, we must note that paths with the same number of loops may have different probabilities, depending on the number of connections which are traversed twice. For instance, if there exist two paths of the same length with one path passing through 10 distinct connections (both open and closed) but the other passing through 11 distinct connections, the latter path will have a probability which is half of that of the path with 10 distinct connections, since each distinct connection would be either open or closed with a probability of ½ for each choice. As a result, we must count the number of paths of each length in which n distinct connections are traversed, for all permissible values of n. First, we must notice that for each connection that is traversed twice in the path, there exists a proper substring of the corresponding binary string such that the length of the proper substring is a multiple of 2c and its sum of r-values is 1. This is because for the same connection to be crossed twice, the particle must land up on the point exactly one level above or one level below (depending on the orientation of the connection) the point from which the connection was crossed. The sum of r-values must be 1, and not -1 because if the orientation of the connection is downwards, then the particle must land up on the point exactly one level below, so the negative signs due to the orientation and change in level cancel out, leading to a sum of r-values of 1. So the problem mentioned earlier in the paragraph translates to finding the number of paths of each length which have n proper substrings, each of which has a length of 2kc, for some positive integer k, and a sum of r-values of 1.

To solve this problem, we must first define a 2^{2c} dimensional vector such that the ith element of the vector is 1 if the binary string of length 2c having a base 10 value of i has a sum of r-values of 1. Otherwise, the ith element of the vector is 0. Call this vector C_1.

Now let’s define another 2^{2c} dimensional vector such that the ith element of the vector is 1 if the binary string of length 2c having a base 10 value of i ends with a substring (not necessarily a proper substring) whose length is a multiple of 2c and whose sum of r-values of 1. Call this vector B_1. Note that $C_1 = B_1$, but treating them as separate vectors will help us in generalizing our solution.

Use these two vectors to create a $2^{2c} \times 2^{2c}$ matrix called D_1, such that $(D_1)_{ij} = (C_1)_i + (B_1)_j$. For simplicity, call a binary string *interesting* if its length is a multiple of 2c and its sum of r-values is 1. It is easy to see that $(D_1)_{ij}$ gives the number of interesting substrings of length 2c in the binary string of length 2c + 1 formed by concatenating the binary strings corresponding to i and j. However, $(D_1)_{ij}$ gives us this value if and only if it is possible to concatenate the two binary strings in such a manner.

Recall that the adjacency matrix of the De Bruijn graph containing all binary strings of length 2c tells us whether it is possible to concatenate the binary strings corresponding to i and j in the manner described above. Using this insight about the adjacency matrix, we can create a function which when iterated an appropriate number of times will give us a matrix whose elements give the number of interesting substrings of length 2c in each binary string of length 4c. Before we create this function, let’s define a square matrix B_1' such that each row of B_1' is the row vector B_1.

Now, let g be a function which takes four square matrices W, X, Y, Z of the same dimensions as its input, and it outputs one matrix F of the same dimensions, such that:

$$F_{ik} = \sum_j W_{ij} \cdot Y_{jk} \cdot \left(X_{ij} + Z_{jk}\right)$$

If we set W and Y as the adjacency matrix T_c, X as D_1 and Z as B_1', F will be a matrix such that F_{ik} gives us the number of interesting substrings of length 2c in the binary string of length 2c + 2 which results from the appropriate concatenation of the binary string corresponding to i and the binary string corresponding to k, assuming that it is possible to concatenate them so and assuming that they don’t contain any good substrings of length 2c.

If it is not possible to concatenate the binary strings of i and k, or if the resulting binary string of length 2c + 2 contains a good substring of length 2c, then F_{ij} = 0. This is because then the corresponding value in at least one of the adjacency matrices will be 0 for each value of j, resulting in F_{ik} = 0. Furthermore, if the resulting binary string is valid, then we get the number of interesting substrings of length 2c present in it, as mentioned earlier, because we add the number of interesting substrings in the first 2c + 1 digits with 1 if the last substring of length 2c is interesting and with 0 otherwise. At this point, you may have noticed the potential problem that due to the summation over all j, the number of interesting substrings of two different binary strings may be added together. However, note that given a concatenation of two binary strings, the string in the middle (i.e. from the 2nd digit to the (2c + 1)th digit) is uniquely determined by the 2 binary strings, so the problem mentioned in the previous sentence does not arise. Now, recall that the adjacency matrix squared tells us whether it is possible to concatenate two binary strings to get a binary string of length 2c + 2. Since 2c $\geq$ 2, each element of the matrix is either 0 or 1, which tells whether the concatenation is possible. Since the length of the string needs to be less than or equal to 4c for each element of the matrix to be either 0 or 1, we can repeat the process till we get a concatenation of length 4c. We will inherently assume this result later in the paper. Hence if we replace the initial W with T_c^2 and if we replace X with the matrix F which was the output of the function, the function will now give us the number of interesting substrings in the binary string of length 2c + 3 which results from concatenating the binary strings corresponding to i and k. Now, we can repeatedly iterate this process with W being higher powers of the adjacency matrix and X being the output of the previous iteration to get a matrix representing the number of interesting proper substrings in each valid binary string (which does not contain a good substring of length 2c) of length 4c. It is not hard to see that for coordinates not corresponding to valid binary strings, the entry is 0, as then the corresponding entries in at least one of the adjacency matrices is 0. Using this matrix, we can get the number of true binary strings of length 4c with k interesting proper substrings by simply counting the number of entries in the matrix which are equal to k and such that concatenating the coordinates (binary strings corresponding to i and j) gives us a good binary string. Let the functions h_k , for all positive integers k, be responsible for performing the above counting for entries equal to k.

Now, let's generalize this process to binary strings of length 2kc. So let C_n be a sequence of vectors such that the nth vector in the sequence is a 2^{2nc} dimensional vector where the ith element in the vector is 1 if the binary string of length 2nc having a base 10 value of i is an interesting binary string, and it is equal to 0 otherwise. Let B_n be a sequence of vectors such that the nth vector in the sequence is a 2^{2nc} dimensional vector such that its ith element is 1 if the binary string of length 2nc having a base 10 value of i ends with an interesting substring (not necessarily a proper substring) whose length is a multiple of 2c. Note that the binary strings can end with at most only one interesting substring whose length is a multiple of 2c as long as the binary string does not contain any good proper substring whose length is a multiple of 2c. Suppose otherwise. Then the sum of r-values of 2 substrings at the end with lengths 2ac and 2bc would both be equal to 1. However, this implies that there exists a proper substring with length 2|b – a|c whose sum of r-values is 0, a contradiction to the fact that the binary string has no good proper substrings whose length is a multiple of 2c. Note that although $B_n = C_n$ for n = 1, as noted earlier, this does not hold in general. So now, we will use a step of recursion to show how is it possible to get the matrix representing the number of interesting proper substrings for binary strings of length 2(k + 1)c from the corresponding matrix for 2kc for values of $k \geq 2$. Call the corresponding matrix for 2kc as G_{2kc}. Now, we will convert G_{2kc} into a 2^{2kc} dimensional vector which contains $(G_{2kc})_{ij}$ for all skewed ordered pairs (i, j) in dictionary order (we defined skewed ordered pairs in our discussion of the skewed product). Note that the reason behind this is that each skewed ordered pair corresponds to a unique binary string of length 2kc, which is determined by concatenating the binary strings corresponding to i and j. It can also be seen that each binary string of length 2kc is accounted for by the skewed ordered pairs, and that they are in ascending order of their base 10 values since the skewed ordered pairs are in dictionary order. Now, we add C_k to this vector because we must also count the interesting substrings of length 2kc. Call the resulting vector as C. Then, we can define a matrix D_k as:

$$(D_k)_{ij} = C_i + (B_k)_j$$

Note that this is analogous to how we defined D_1. Now, we will define B_k' analogously as well, by letting it be a square matrix such that each of its rows is the row vector B_k. Now we can simply iterate the function g to get the corresponding matrix for 2(k + 1)c and we are done, but before that we must clarify what will the arguments of the function g be in this case. Y must be the adjacency matrix of the binary strings of length 2kc which do not have any good substrings (not necessarily a proper substring) whose length is a multiple of 2c (note that this adjacency matrix does contain binary strings with good substrings whose length is a multiple of 2c as vertices, but the edges connected to these vertices are removed). We have already found a formula for these adjacency matrices earlier in the paper and we had called the sequence of these adjacency matrices as $M_{n,c}$. Hence, Y must be $M_{(k+1),c}$. W is initially $M_{(k+1),c}$ and with each successive iteration W takes on the next higher power of $M_{(k+1),c}$. Concretely, W is initially $M_{(k+1),c}$, then it becomes $M_{(k+1),c}^2$ for the next iteration, then it becomes $M_{(k+1),c}^3$ and so on. Z_{ij} must represent whether the binary string corresponding to j ends with an interesting substring (not necessarily a proper substring) whose length is a multiple of 2c. Recall that a binary string can end with at most one interesting substring whose length is a multiple of 2c provided that the binary string does not contain any good proper substring whose length is a multiple of 2c. If it does contain a good proper substring whose length is a multiple of 2c, then at least one of the corresponding entries in the adjacency matrices would be 0, making the entire term 0. So we just need to consider whether the binary string ends with an interesting substring whose length is a multiple of 2c, which is exactly what Z_{ij} does. It is easy to see that such a matrix is the same as B_k'. So Z will be equal to B_k' for all iterations. Lastly, X is initially D_k and then it is the output of the previous iteration. The explanation for this is the same as that given for the case where k was equal to 1. So after iterating this function 2c – 1 times, we will get the corresponding matrix for 2(k + 1)c.

Now, we are ready to figure out the expression for the expected value of loops for a path with c connections. First, let's call the sequence of matrices obtained by the process above as $P_{n,c}$. So, $P_{k,c}$ represents the $2^{2(k-1)c} \times 2^{2(k-1)c}$ matrix such that $(P_{k,c})_{ij}$ is the number of interesting proper substrings of the binary string of length 2kc which results from concatenating the binary strings corresponding to i and j. If it is not possible to concatenate these binary strings or if the resulting binary string contains a good proper substring, then the entry of the matrix at (i, j) will be 0 because then the corresponding entries in at least one of the adjacency matrices W and Y would be 0.

Theorem 1: The expression for the expected value is:

$$\frac{\binom{2c-1}{c-1}}{2^{2c}} + \sum_{n=2}^{\infty} n \frac{f_c\left(M_{n,c}^{2c}\right) + \sum_k (2^k - 1) h_k\left(P_{n,c}\right)}{2^{2nc}}$$

Where k takes on all positive integer values q such that at least one entry in the matrix $P_{n,c}$ is equal to q.

Proof:

This expression follows from the fact that the expected value of loops can be represented in the form 1 x P(L = 1) + 2 x P(L = 2) … , where L is the number of loops and P(L = r) is the probability that the number of loops is equal to r. The $2^k - 1$ factor follows from the idea that if the number of interesting substrings increases by 1, then the number of distinct connections traversed decreases by 1, so the probability of that path occurring gets doubled. Since each path is accounted for once in $f_c\left(M_{n,c}^{2c}\right)$, the factor is $2^k - 1$ and not 2^k. Notice that in the summation, each term takes into account the probability of each path being periodic and it having n loops, rather than taking the probability of a periodic path having n loops. But, since a path on the cylinder is periodic with probability 1, both of the probabilities mentioned in the previous sentence are equal to each other. Hence, we have found an expression for the expected value of loops for periodic paths on an infinite cylinder, which was the ultimate goal of this paper. Q.E.D

We have added the first few terms of the infinite series for c = 2 and c = 3 using a computer program.

The number of terms in the infinite series that were included in the sum	c = 2	c = 3
One term	0.5	0.3984375
Two terms	0.921875	0.721435546875
Three terms	1.40625	N/A

Upon comparing the values in the table above with the numerical results table on page 6, we get a rough idea of how quickly the infinite series converges. Note that we could not add more terms of the infinite series because doing so is computationally very expensive. For instance, computing the fourth term of the infinite series for c = 2 involves computations with 65536×65536 matrices which are way beyond the capability of the standard PC. Similarly, computing the third term of the infinite series for c = 3 involves computations with 262144×262144 matrices which again supersede the capabilities of the standard PC. Nevertheless, we hope that the values in the table above give the reader some insight into how quickly the infinite series converges.

Conclusion:

In this paper we have considered the deterministic walk taken by a particle on a cylinder with infinitely many levels whose connections are randomly decided to be open or closed. We have found a formula which gives us the expected value of loops a particle takes in one period of a periodic path on this infinite cylinder given the number of connections between two adjacent levels in the underlying graph. We hope that the techniques introduced in this paper to solve this particular problem regarding deterministic walks in random environments in 2 – dimensions helps solve some of the most prominent open problems in this area of research.

References:

[1] Bunimovich, Leonid and Khlabystova, Milena (2003) One-Dimensional Lorentz Gas with Rotating Scatterers: Exact Solutions, Journal of Statistical Physics, - J STATIST PHYS. 112. 1207-1218. 10.1023/A:1024623827182.

[2] Bunimovich, Leonid and Troubetzkoy, S. (1992) Recurrence properties of Lorentz lattice gas cellular automata, Journal of Statistical Physics, 67. 289-302. 10.1007/BF01049035.

[3] Bunimovich, Leonid and Troubetzkoy, S. (1993) Topological dynamics of flipping Lorentz lattice gas models, Journal of Statistical Physics, 72. 297-307. 10.1007/BF01048051.

[4] Bunimovich, Leonid and Yurchenko, Alex (2007) Deterministic walks in Markov environments with constant rigidity, 10.1090/conm/430/08251.

[5] Bunimovich, Leonid and Yurchenko, Alex. (2008). Deterministic walks in rigid environment with aging. Discrete and Continuous Dynamical Systems. Series B. 1. 10.3934/dcdsb.2008.9.37.

[6] Bunimovich, Leonid (2004) Deterministic walks in random environments, Physica D: Nonlinear Phenomena, 187. 20-29. 10.1016/j.physd.2003.09.028.

[7] Bunimovich, Leonid (2000) Walks in rigid environments, Physica A, 279. 169-179. 10.1016/S0378-4371(99)00518-X.

[8] Bunimovich, Leonid (2003) Walks in rigid environments: Symmetry and dynamics. Asterisque.

[9] Ruijgrok, Th and Cohen, Ezechiel (1988) Deterministic lattice gas models, Physics Letters A, 133. 415-418. 10.1016/0375-9601(88)90927-9.

[10] Webb, B. and Cohen, Ezechiel (2015) Self-Limiting Trajectories of a Particle Moving Deterministically in a Random Medium, Journal of Physics A: Mathematical and Theoretical, 48. 10.1088/1751-8113/48/48/485203.

[11] Yurchenko, Aleksey (2020) Some problems in the theory of open dynamical systems and deterministic walks in random environments.

E-mail Address of Author:

tanavchoudhary@gmail.com